MediaStore: World Wide Web access to multimedia database systems

Peter Kutschera
Institute for Parallel and Distributed High-Performance Systems (IPVR)
University of Stuttgart, Germany
E-mail: Peter.Kutschera@informatik.uni-stuttgart.de

1 Introduction

The evolution in technology during the last few years has led to an increase in processor speed and to an improvement in the areas of tertiary storage techniques as well as communication networks. Parallel to the performance increase, the prices for hardware decreased and made the development of new applications feasible. Especially in the area of multimedia computing, developers began to implement prototypes ranging from video-on-demand applications to computer based learning environments. All of these applications had in common that their developers characterized them as the "killer applications" of the forthcoming age of multimedia computing. Independent of this situation, a few years ago started the World Wide Web (WWW) project at the European Center for Nuclear Research (CERN) in Geneva. Its goal was the implementation of a distributed hypermedia system. These research efforts at CERN were not noticed from a broader audience until the National Center for Supercomputing Applications (NCSA) developed the graphical user interface Mosaic, which made the World Wide Web for Unix workstations (connected to the Internet) accessible. After the development of the browser's first version in 1993, the Web gained more and more popularity. In 1995, it made its way even into mass media and is nowadays a household name. Thus, a lot of people today develop different applications for the World Wide Web. The popularity of the World Wide Web, which provides on-line access and graphical presentation to a huge framework of resources world wide, is mainly based on the availability of the software on almost every hardware platform, the easy and appealing graphical interface of the browser, the simplicity to make information accessible world wide and the possibility to provide some basic form of user interaction. The availability of the software on different platforms and the user-friendly graphical interface make the World Wide Web an ideal basis for a database frontend to multimedia database systems.

This paper presents the design and implementation of a World Wide Web based database browser with the capability to access traditional data as well as continuous data like audio - and video clips. In the following sections, the term "traditional data" refers to relations in the relational data model or objects in the object-oriented data model. That kind of browser can be used during editorial work for radio broadcasts or television programs and any other application area like search engines for tele-teaching material, audio- or video libraries, etc. which incorporate multimedia data. The basic idea of the MediaStore browser is to provide the user with a graphical interface that hides the underlying data model as well as the query language of the database system and avoids the need for specialized HTML tags to include database queries into the HTML documents. The objective during the implementation of the browser was to accomplish these ideas without any modification of the World Wide Web software to ensure the browsers' usability in many different environments.

2 The World Wide Web

The World Wide Web is definitely the most popular hypermedia system at the moment. The idea behind hypermedia systems is the management of non-linear information consisting of text, images, video and audio data. The different types of data can be linked to each other by the means of hyperlinks which provide the user with a navigational access to a huge framework of resources. A majority of these resources are hypermedia documents which are formatted according to the hypertext markup language (HTML) and are accessible by the means of a graphical frontend which is called a browser. Some of the most popular browsers today are the Mosaic browser and the Netscape Navigator. The appealing point-and-click interface provides the user with a navigational interface along the hyperlinks which is easy to comprehend. Due to the large number of existing image -, audio - and video formats, the available browsers have to use external viewers or helper applications for the presentation instead of displaying them within the browser. For that reason, the playback of a video using an external viewer is accomplished as follows: The browser waits until the video data has been transmitted entirely, stores them on the local disk and starts an external viewer to display the video clip. An alternative to external viewers is to use Netscape's plug-in facility [7], which is available for the Netscape Navigator. Plug-ins are enhancements of the browser with the capability to make more different types of data accessible. In comparison with the browser, which presents the user interface of the World Wide Web, the server manages the resources of the World Wide Web and makes them accessible from all over the world. The interaction between the browser and the server is based on the stateless hypertext transfer protocol (HTTP), a simple request/response protocol. The access to the different resources world wide on behalf of the browser is accomplished by using their physical address, which is known as their uniform resource locator (URL). After receiving a request containing a resource's URL, the server sends back the requested resource in response. Additionally to the access of resources, which are managed by the World Wide Web servers, the World Wide Web can as well be used by means of the common gateway interface (CGI) to access external resource managers like database systems. The access to external resources can be accomplished by a program, which is called a CGI script and is started by the server on behalf of the browser. As a result of its computation, the CGI script generates a HTML document dynamically and sends the document back to the World Wide Web server. The World Wide Web server on its turn forwards the document to the browser where it will be displayed. The CGI interface and the HTML forms extension, which are part of the HTML 2.0 standard, enable some basic interaction between the user and external resource managers like database systems.

2.1 Database access via World Wide Web

Developing a World Wide Web based interface to database systems, four different approaches have to be considered. These approaches differ with regard to their efficiency and the necessary modifications to the World Wide Web software that have to be made. Basically, the following design alternatives are possible:

1. The interaction between the database system and the browser could be handled by CGI scripts. This is the standard method to interface external data sources. In that case, the CGI script handles the connection and transmits the users' queries to the database system, receives the results and transforms them into HTML documents.

2. The browser could be modified to allow a direct access to the database system. This approach has been used for the JDBC proposal[10] that enables database access from Java programs, which are executed within the browser.

3. Correspondingly, the server could be modified to handle database requests by introducing a special form of database URL that is used to denote references to data in the database. In that case, the World Wide Web server itself connects to the database system, transforms the URL notation into a database query, receives the results, includes them into a HTML document and sends them back to the browser.

4. At last, it is possible to modify the server as well as the browser to provide access to database systems.

The access to database systems using CGI scripts is in fact not as efficient as using a modified version of the World Wide Web software due to the overhead involved in starting up a CGI script for every single database access. Nevertheless, we choose the CGI interface to access the database, because it is the standard interface to external resources and the use of standards ensures the usability of the MediaStore browser in a broad range of different environments. Any modification to the World Wide Web software on the other hand, leads to a limited usability, because its usage requires the installation of special software components prior to its use.

2.2 Related Work

A lot of work has been done in the field of World Wide Web based interfaces for database systems. As a result, a number of commercial products already exist, including solutions for relational database systems like Sybase[15], Oracle[6], and DB2[4], as well as object-oriented database systems like ObjectStore[14] or LINCKS[11] and object-relational systems like Illustra [5]. All of the commercial products as well as the research prototypes [9, 13, 2] have in common that they rely on an external schema description of the database and use vendor-specific extensions to the HTML language to include database queries into HTML documents. Thus, it is the application programmer's task to design HTML pages for each database query as well as result set of the query. Besides, none of these interfaces are, to our best knowledge, able to access continuous data residing in the database. Although, a few research prototypes already propose extensions to the current World Wide Web architecture to handle continuous data as well. One of these approaches tries to save network bandwidth by transmitting script files describing musical scores[8] or animations[1] respectively, instead of digitized audio - or video data. These script files are transmitted to a modified World Wide Web browser that is able to transform them into music or animations respectively. The other approaches are using a modified World Wide Web server which is able to handle continuous data that is transmitted using HTTP[12] or a more sophisticated transmission protocol[3] respectively.

3 Design concepts

3.1 Presentation of textual data

In principle, it is possible to use two different approaches to handle the presentation of data residing in the database. The first possibility is to leave the responsibility for the creation of the HTML documents that are used to query the database completely to the database browser. Thus, the browser has to create the HTML pages during runtime based on the actual database schema description (see figure 1). The second possibility is to assign this task to the application programmer. This approach relies on HTML documents that are created manually by the application programmer. These HTML pages contain HTML form elements like text fields, menu bars, etc. as well as database queries, which are included using vendor-specific extensions to the HTML language. The access to that kind of HTML page is accomplished by a CGI script that has to extract the database query out of the page, submit the query to the database system, receive the result values and create the appropriate HTML page to display the query result to the user. Leaving the document creation to the browser, it is not possible to adapt the document format according to one's views, but not any HTML document as to be created manually. The advantage of this approach is the avoidance of inconsistencies between HTML pages containing database queries and the actual database schema in case of changes of the database schema. Thus, the administration overhead for such an environment is considerable lower. In our opinion this advantage outweighs the deficit that the data presentation cannot be changed. That was the reason why this approach has been chosen for the implementation of the MediaStore browser.

```
create distinct type movie as blob (4 M)

create table tapes (
nr          integer not null,
title       varchar(30) not null,
primary key (nr, title),
foreign key (title) references video
)

create table video (
title       varchar(30) not null,
director    varchar(40),
year        date,
genre       varchar(25),
distributor varchar(30),
film     movie,
primary key (title),
foreign key (distributor) references Distributor,
)
```

Figure 1: A sample database schema

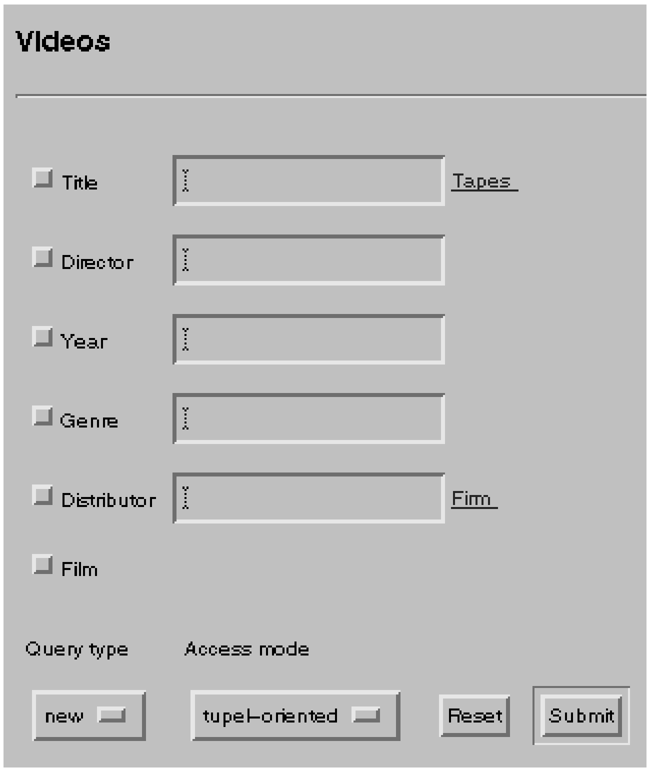

Figure 2: The graphical presentation of the video entity of the database schema that is used in figure 1

3.2 Hypertext-based user interface

One of the basic ideas of the database browser is to use the point-and-click interface of the World Wide Web browser to provide the user with an intuitive, navigational interface to multimedia database systems. An important requirement to achieve this goal, is to provide the user with an abstraction of the underlying data model and the query language of the database system. To simplify the interaction with the database system even further, the same representation for querying the database and the display of the results has been chosen. In this way, the user is able to stepwise refine his queries which are limited to a single entity type (relation or object). Querying the database means selecting the appropriate entity type and activate the attributes that should be part of the result set via mouse clicks with the possibility for further limitation of the result set by specifying certain attribute values. The relationships between different entity types (object references in the object-oriented data model or primary/foreign key relationships in the relational data model) are represented using hyperlinks (see figure 2). Although the database browser provides the user with a tuple-oriented interface to the database with the possibility to stepwise refine his queries, this access method is not feasible for the display of a large result set. Therefore, the user is able to choose between a tuple-oriented and a set-oriented access mode to display the query results. Using the set-oriented access mode, the user is able to select one of the tupels in the result set for further refinement of the database query.

3.3 Real-time presentation of continuous data

As it was already stated in section 2, the World Wide Web browser handles continuous data by means of external viewers. This method is not feasible for longer data streams, because the user has to wait quite a long time before the data will be displayed. Especially in situations where the user just wants to a check the content of a unknown resource, this mode of operation quickly becomes annoying. For that reason, an important feature for the presentation of continuous data is to interleave the receipt and the display of the video - and audio streams. Thus, the database browser environment has to provide presentation services, which are capable of doing this. The consequence of the interleaved receipt and display of continuous data is, that it it necessary to maintain some sort of real-time requirements during playout to avoid degradation of quality in form of "hiccups". But, a guarantee of these real-time requirements for the entire duration of the playout requires a resource reservation protocol, which includes all of the involved resources along the data path beginning with the retrieval of the data from disk to the display of the data by the presentation services. However, todays' hardware and software environments, apart from a few research prototypes, do not provide those capabilities to make reservations in form of memory -, cpu -, I/O bandwidth - or network bandwidth requirements. On the other hand, it is at least questionable if a heterogeneous environment like the Internet, where the World Wide Web is used most, will ever incorporate these capabilities in a suitable and dynamic fashion across all the different hardware - and software platforms. Thus, a lot of systems use a "best effort" strategy for transmitting continuous data over the network. The drawback of this approach is that the presentation quality is dependent on the available network bandwidth. Although, it is not possible to guarantee real-time requirements by all means, with some additional effort, it is possible to guarantee at least a minimum quality of service (frame rate per second, image size, etc.) during playout. With this aim in view, the database environment has been extended by a component that is responsible for monitoring the available I/O bandwidth for the retrieval of the continuous data as well the network bandwidth that is available for the transmission. Thus, prior to the retrieval and transmission of continuous data, this component has to be contacted to decides, if an incoming request on continuous data can be granted or has to be rejected due to bandwidth limitations. A request is handled, if there is enough bandwidth available without violating existing quality of service guarantees of previous requests. If a request cannot be handled due to bandwidth limitations, the user is informed accordingly. Although, this approach prevents quality of service violations at the beginning of the playout, it does not prevent from quality of service violations during playout due to an increased network contention or increased load factor of the cpu where the presentation services reside. Thus, the presentation services use a feedback mechanism to inform the database environment whenever such a situation occurs. An increased load factor of the local cpu is monitored by a significant drop in the frame rate that is achieved during display, whereas a buffer underflow indicates an increased network contention. Based on the feedback provided by the presentation services, the database environment lowers the quality of service in a stepwise manner, until no more quality of service violations are monitored. The basic idea is to provide the user with a smoother playout compared to a "best effort" strategy.

3.4 Interaction between the database system and the World Wide Web

Accessing database systems by means of the CGI interface results in an overhead due to the start-up costs for the CGI script that has to be invoked every time the user wants to access the database. Thus, it is very important to maximize the performance of the CGI script. A very big influence of the performance has the size of the CGI script. A tiny script of only a few kilobytes written in Perl will be loaded much quicker by the operating system than a huge C program with a few hundred kilobytes code. Thus the start-up time can be reduced using a Perl script that implements only some basic functionality. In principle, it is possible to access the database system directly by means of the CGI script, but this would lead to a very inefficient design due to the overhead of opening and closing a connection to the database system for every database access. Thus, the CGI script should only transmit the user's queries to another component that is part of the environment and maintains a durable connection to the database system as long as the user accesses the database. Due to the tuple-oriented interface of the database browser, it is necessary to provide an additional mechanism that prevents the browser from accessing the database system for every tuple access. Thus, another component has to be introduced that implements a result cache and resides on the same host like the World Wide Web server and the CGI script. After the database query has been processed by the database system, the query results are sent back to the component that implements the result cache. After the tuples have been cached, the CGI script is able to fetch successive tuple by means of shared memory access instead of sending a request to the database environment.

4 Implementation

The architecture of the MediaStore browser, which is shown in figure 3, can be derived directly from the design concepts that have been introduced in the previous sections. This architecture is the basis of the prototype implementation of the MediaStore browser that has been developed in a networked environment of heterogeneous Unix workstations and a relational database system. The general idea of the prototype implementation was the creation of a testbed to show the feasibility of the underlying design concepts and to build a basis for further development.

All components of the MediaStore browser can be characterized according to their location as client-side components, which constitute the browser's user interface, server-side components, which are located on the same host as the World Wide Web server, and database-side components, which are responsible the database access.

Following client-side components constitute the user interface of the database browser:

Frontend: Any standard World Wide Web browser like the Netscape Navigator or the Mosaic browser can be used as the frontend. The interaction between the user and the MediaStore browser is based on HTML forms. After the user logs in, using an existing login name or a special guest account, his access rights are checked and a list of accessible relations are presented. Now he has to choose one relation out of the list as a starting point of his database queries.

Presentation services: The current prototype implementation consists of two software-based public-domain player, namely the MPEG audio player V1.1 of the TU Berlin and the MPEG-1 video player V2.1 of the University of Berkeley. The source code of these players has been adapted that they could be used in a networked environment. Now they are able to receive the MPEG data over the network and report quality of service violations to the media servers on the database-side.

The following components reside on the same host as the World Wide Web server and therefore are called server-side components:

World Wide Web server: The World Wide Web server is responsible for the data transfer between the CGI script and the World Wide Web browser. The CGI script itself is started by the World Wide Web server on behalf of an user interaction at the frontend.

CGI script: The CGI script, which is a Perl script of a few kilobytes code, simply has to act as a fast communication channel between the World Wide Web browser and the DB proxy. Since the DB proxy resides on the same host as the CGI script, the data transfer between these components is based on local memory calls. The prototype implementation uses UNIX System V message queues for the communication between the CGI script and the DB proxy. The CGI script transmits the users' HTML forms input to the DB proxy and in its turn receives the HTML formatted query results.

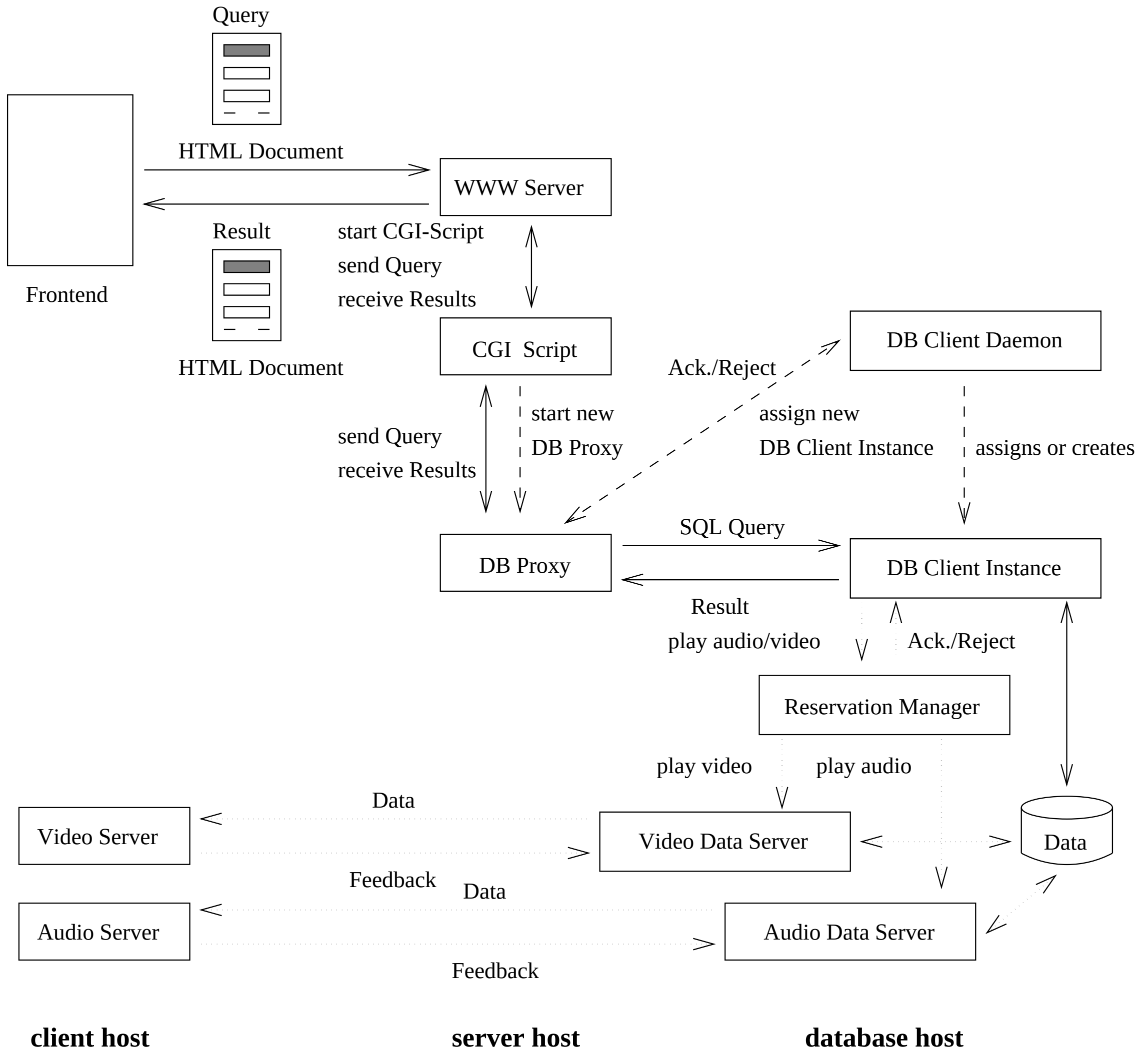

Figure 3: The architecture of the MediaStore browser

DB Proxy: Basically, the DB proxy is responsible for the handling of the different types of user interactions and sending the resulting HTML document back to CGI script. Based on the different types of possible user interactions, the DB proxy has to perform the following three different tasks:

1. Each time the user selects a new relation as the starting point of his database queries, the DB proxy has to check its local cache, if the appropriate HTML document already exists. If the HTML document does reside in its cache, the DB proxy requests the description of the relation which is part of the underlying database schema from its DB client instance on the database-side. Afterwards it uses the description to create the appropriate HTML page and caches it for later use. Finally, the DB proxy sends the resulting HTML page back to the CGI script.
2. Assuming, a new database query has been submitted to the DB proxy, the input of the HTML forms has to be transformed into valid SQL queries. Afterwards, the DB proxy sends the resulting SQL query to its DB client instance on the database-side. For efficiency reasons, the DB proxy maintains a connection to the DB client instance between different interactions. After the query's result set has been received, the DB proxy stores the tuples in its local cache. Based on the user's access mode (tuple-oriented versus set-oriented database access), the DB proxy sends a HTML page, containing all the values of the result set or just a single result value, back to the CGI script respectively.
3. After a query's result set resides in the local cache, each tuple-oriented access to the data results in a cache access instead of contacting the DB client instance.

All components on the database-side belong to the database environment and have following tasks:

DB Client Instance: The DB client instance maintains a connection to the underlying database system and provides access to the traditional data. Because every concurrent user is assigned to a different DB client instance, there are as many active DB client instances as concurrent users. The current implementation is based on IBM's DB2/6000 Version 2.1 as the underlying database system. The tasks of the DB client instance are to access the database as well as to extract the necessary schema information out of the database schema that are needed to create a HTML page for a relation.

DB Client Daemon: The DB client daemon is used to establish a new connection to the database after a successful login of a new user or during a shutdown of an existing database connection. After a new user logs in, a new DB proxy for this user has to be created, which in its turn sends a connection request to the DB client daemon. The DB client daemon that manages a pool of DB client instances, searches for an idle instance and assigns it to the DB proxy. If no more idle DB client instances exist, the daemon has to create a new one.

Media Server: Media servers (video data server and audio data server) are responsible for the access to continuous data residing in the database. For each active presentation service on the client-side exists a media server counterpart on the database-side, which is responsible for the transmission of audio - and video data to the presentation server on the client-side. Although, the presentation services already monitor quality of service violations and send an appropriate feedback to the media servers, the media servers currently do not respond to these violations and no quality of service adaptations are made. The current implementation uses the reliable TCP/IP transport protocol for transmitting continuous data, although a lot of features like re-transmission of lost packets or elimination of duplicate packets are not needed and result in an overhead for the processing of continuous data. Unfortunately, the presentation services that are currently in use, rely on the error-free arrival of continuous data.

Reservation Manager: The reservation manager is contacted every time a database query contains any references to continuous data. Its task is to decide, based on the available bandwidth, if a new request can be handled without violating the quality of service guarantees of previous requests. If there is enough bandwidth available, the reservation manager orders the media server to access the data and transmit them to the appropriate presentation server on the client-side. If a request to continuous data cannot be handled due to bandwidth limitations, it will be rejected and the user will be informed accordingly. In the current prototype implementation, the reservation manager uses a very simple strategy to determine the available bandwidth and to decide if a new request could be handled or should be rejected. The available network bandwidth is determined by the analysis of round-trip times resulting from "ping" requests to the users' host. The maximum I/O bandwidth is expressed by a maximum number of requests that can be handled concurrently. Because this maximum value heavily depends on the number of disks used for the storage of continuous data as well as the partitioning of the continuous data, this maximum value is very system-specific.

5 Conclusion and Future Work

In this paper, the design concepts as well as a prototype implementation of the MediaStore browser, a World Wide Web based user interface for multimedia database systems, has been presented. Although, the main goal could be achieved with our prototype, the following shortcomings have been identified and should be addressed by some future work.

The use of the MediaStore browser requires the installation of additional software, namely the modified MPEG players for video - and audio clips. Preferably, such a browser should incorporate the functionality to reload additional presentation services on demand whenever they are needed. This functionality could be achieved by implementing the presentation services as Java applets, which are loaded on demand when accessing a HTML page containing references to video - or audio data. This approach provides the possibility to extend the MediaStore browser with new video and audio formats in a user transparent way.

Our preliminary performance measurements showed that cacheing of tuples results in an significant improvement of efficiency of the tuple-oriented access. However, if the cache would reside inside of the World Wide Web browser, the performance could be improved even further. This functionality could be achieved by using Java applets for the graphical frontend and the tuple cache.

In the future, we plan to add the necessary enhancements to the MediaStore browser to store other types of data like images or HTML documents in the database and make them accessible as well.

References

[1] K. Ø. Arisland, S. Johansen. *Introducing CandleWeb and Å(awe), Bringing Animation Power to the World Wide Web.* Fourth International World Wide Web Conference 1995.

[2] A. Clausnitzer, P. Vogel, S. Wiesener. *A WWW Interface to the OMNIS/Myriad Literature Retrieval Engine.* Third International Conference on the World Wide Web 1995.

[3] Z. Chan, S.-M. Tan, R. H. Campbell, Y. Li. *Real Time Video and Audio in the World Wide Web.* Fourth International World Wide Web Conference 1995.

[4] *DB2 World Wide Web Connection.* Available at http://www.software.ibm.com/data/db2/db2wfac2.html.

[5] J. Gaffney. *Illustra's Web DataBlade Module.* SIGMOD Record, Vol. 25, No. 1, March 1996.

[6] A. Hunter, R. I. Ferguson, S. Hedges. *SWOOP: An application generator for ORACLE/WWW systems.* Fourth International World Wide Web Conference 1995.

[7] *Netscape Plug-in Guide.* Available at http://home.netscape.com/eng/mozilla/Gold/handbook/plugins/index.html.

[8] J. van Ossenbruggen, A. Eliëns. *Bringing Music to the Web.* Fourth International World Wide Web Conference 1995.

[9] L. Perrochon, R. Fischer. *IDLE: Unified W3-Access to Interactive Information Servers.* Third International Conference on the World Wide Web 1995.

[10] *The JDBC(tm) database access API.* Available at http://splash.javasoft.com/jdbc/.

[11] M. Sjölin. *A WWW Front End to an OODBMS.* Second International Conference on the World Wide Web 1994.

[12] J. C. Soo. *Live Multimedia over HTTP.* Second International Conference on the World Wide Web 1994.

[13] C. S. Varela, C. C. Hayes. *Zelig: Schema-Based Generation of Soft WWW Database Applications.* First International Conference on the World Wide Web 1994.

[14] C. Varela et. al. *DB: Browsing Object-Oriented Databases over the Web.* Fourth International World Wide Web Conference 1995.

[15] *Sybase web.sql.* Available at http://www.sybase.com/products/internet/websql/web_spec.html.