# FREIE UNIVERSITÄT BERLIN

Streaming Databases for Audio and Video

Theory and Praxis

Margarita Esponda

B-07-15
October 2007


**FACHBEREICH MATHEMATIK UND INFORMATIK**
**SERIE B · INFORMATIK**

# Streaming Databases for Audio and Video
# Theory and Praxis

Margarita Esponda

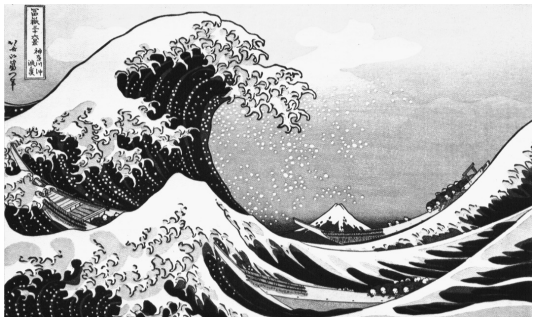
Institut für Informatik
Freie Universität Berlin
Takustr. 9
14195 Berlin

October 2007

## Abstract

This paper offers a review of technological developments regarding massive storage devices and the emergence of new streaming services for audio and video on-demand. While the capacity of storage devices has been increasing exponentially, the price of a stored bit has been falling at an even faster rate. Audio and video files can be stored on-line in personal computers and computer networks. Accessing this information requires new type of databases capable of handling special types of queries: access through annotations and metadata, access by similarity, and access by feature search. Databases with in-built streaming capabilities would be extremely useful for transmitting the information to the end user, while providing at the same time consistency checks, indexing, reporting, and querying features. They should also be fast and scalable. This paper reviews recent academic and industrial projects, and describes an implementation of a streaming video servlet based on the Oracle SQL database, its Binary Large Object storage data type, and the Java Database Connectivity interface.

## 1 Motivation: The Information Tsunami



Recently, there has been an increased interest in developing databases for massive amounts of storage, but specially for audio and video. Broadcasts, that is, music, all kind of audio signals as well as video, constitute the bulk of the information (measured in bits) that we perceive any given day. Traditional services, such as radio and television, produce large amounts of material which is handled

manually in storage rooms. However, most of this information is being digitized and new productions are stored in digital form. In a few more years, even Hollywood movies will be filmed using digital cameras. The traditional movie rental store will disappear and will be substituted by digital streaming services which send the desired movie to our homes. In Germany, T-Online is already offering such a service (vision.t-online.de). In the case of music, digital stores such as Apple's I-Tunes, and other similar companies are rapidly substituting CDs as information carriers. On-line sales of music tripled in 2005 [EETimes 2005]. Many other kinds of streaming services are also appearing, such as real-time stokes quotes, or even streaming of information from sensor networks in agriculture, or in hospitals.

Therefore, the issue is whether traditional databases can accommodate such new applications, or if they should be modified to deal with the new requirements. In a traditional database for handling text, for example, documents are usually retrieved as complete units. The whole document is read or stored, and the application waits until the transaction is complete before attempting to use it. However, in the case of audio and video, a transaction can take a long time – it has a temporal dimension which cannot be ignored. This has led some researchers to postulate the need for "multimedia databases", that is, databases capable of dealing with conventional textual information but also with pictures, audio, and video, possibly synchronized.

## 1.1 Technology Push: the development of mass storage devices

In the field of computer technology, there are usually two sides to consider when new applications emerge. On the one side we have the *technology push* provided by new capturing devices and increased storage availability. Here, technology is providing new tools and opportunities for dreaming and realizing new applications. On the other side, we also have the *demand pull* arising from some "killer applications" ready to consume all the storage and bandwidth which technology can provide. Killer applications for audio are, for example, MP3 stick players, or I-Pods, for which there is a very important mass market. Apple, the pioneering computer company, sells now more I-Pods than Macintosh computers (measured in dollars). In the case of pictures and video, digital cameras have become so cheap that they are integrated in cellular telephones or are small stand-alone devices. The number of bits of data generated by consumers, but also of the bits of data which consumers read at home using their electronic devices is exploding.

The technology push relevant for database technology has been the exponential increase in storage capacity of hard-disks and CD-ROMs. Moore's Law is well known: chip integration (the numbers of transistors on a chip) duplicates every 18 months. However, in the case of hard-disks the pace of development has been even more astonishing. The capacity of a disk drive duplicates every 12 months, for the same price. Half a Terabyte disk drives for PCs are already being sold as of this writing (by Seagate Technologies) and Terabyte drives will appear in the next few months. The Terabyte DVD is also being developed and is around the corner.

Fig. 1 shows data compiled by Edward Grochowski from IBM. The graph shows that the increase in storage density has been exponential (note the logarithmic scale in the *y*-axis). The form factor of the harddisk units has been also falling, so that a microdrive can be included now in MP3 players or digital cameras.

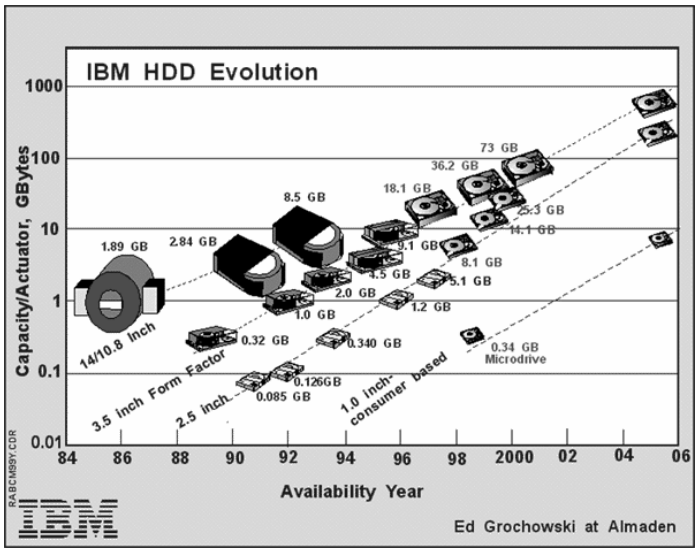



Figure 1: Increase in density of hard-disks for different form factors

As we can see from this data, technology is literally handing us down the possibility of storing all of our data in a single DVD or hard-disk drive. At the same time, technology is putting in our hands devices which produce more data bits as we had done ever before. In a few years, storing all the TV programs we watch will become a matter-of-fact possibility for those willing to store that kind of data.

## 1.2 Demand Pull: New Audio and Video Services

Demand pull refers to those applications which develop a mass market for a certain technology. Demand for music and videos *pulls* the market and the technology further. There is an interplay between technology push and demand pull.

Microsoft's Digital Memories project, started with Gordon Bell's MyLifeBits project is a good example of the kind of data storage which will be needed in the future. Gordon Bell, designer of the PDP-11, is working with Microsoft in order to store all of his data life. This includes all texts he reads or writes, all documents he produces (PowerPoint presentations, annotations, etc.), all e-mails he send or receives, all his telephone calls, all the TV programs or videos he watches, all music he hears, all recordings he makes at his office, etc. Bell thinks that for this kind of information he requires around one Terabyte of storage yearly, which costs now around 300 dollars. Recording all these aspects of his life has become feasible [Gemmell 2002]. In Germany, the European Media Laboratory in Heidelberg is pursuing a similar project which should allow a person to store on-line video of his daily life.

Any middle-class person in an industrial country is slowly catching up with Gordon Bell. The number of digital photographs taken every year is approaching the 100 Billion mark. Digital photography already overtook analog photography, as Fig. 2 shows. In 2003, 57 million analog cameras and 50 million digital cameras were sold. In 2004, 77 million digital cameras were sold, but only 43 million analog ones.

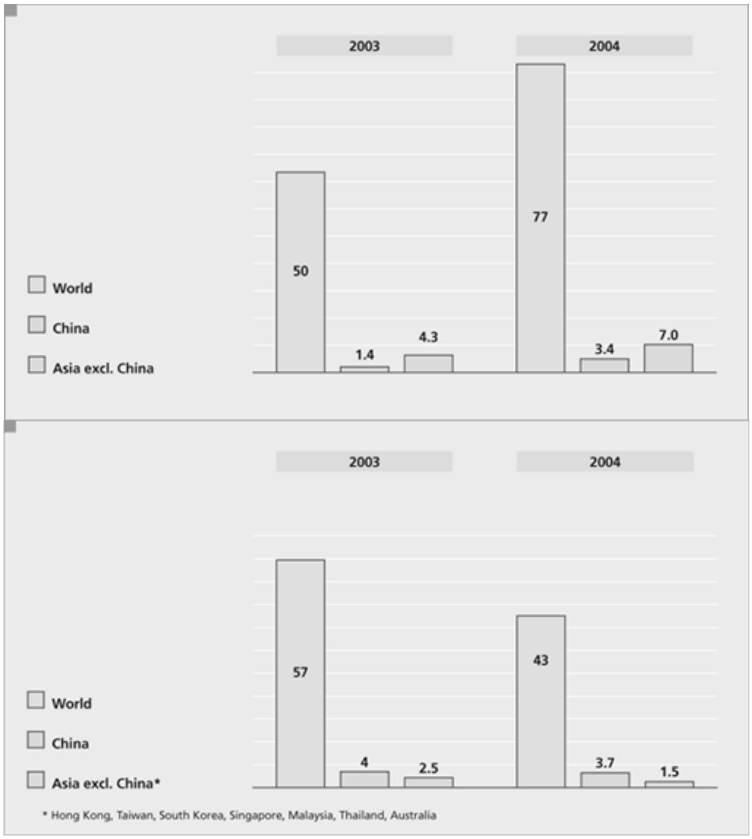



Figure 2:Upper chart: Number of digital cameras sold worldwide in 2003 and 2004. Lower chart: number of analog cameras sold worldwide (million units) in 2003 and 2004.

The most important mobile device fueling the explosion of recordable data is the cellular telephone. Almost all new units include a digital camera, and the most powerful devices can also record video. Fig. 3 shows the exponential increase in the number of subscribers, specially in China. There are now 1500 million cellular subscribers, and it is expected that in July 2006 this number will increase to 2000 million [Esponda 2004a, Adams 2004].

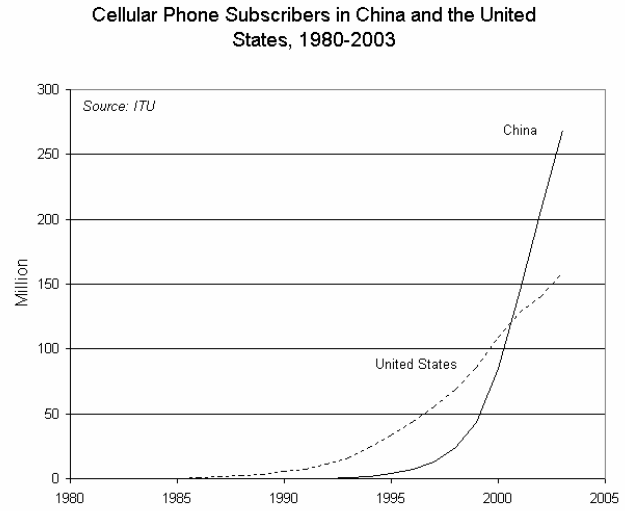



Figure 3:Increase in the number of cellular subscribers in the US and China

In a few years, millions of persons will be carrying a mobile device capable of recording and replaying audio, video, TV. There will be a market for remote servers in which individuals can store their information, paying a small rent for several Terabytes of storage.

## 1.3 The Challenge: Accessing the Information

As Gordon Bell has correctly pointed out, the challenge today is not storing the information, but *accessing* it. It is pointless to store thousands of hours of video, if later on it is not possible to retrieve that single shot in which my daughter is receiving her high-school diploma. TV networks, such as CNN, go to great lengths in order to index their video material, using speech recognition technology and manual annotation. The same kind of data management is needed by the conventional user of the new technologies, who does not yet produce much information, but who will be producing it in a few years. Even today, sometimes the only way to retrieve an old file from the computer is to use an indexing tool such as Google Desktop.

Therefore, new kinds of databases are needed: multimedia databases [Subramanya 2000]. How this databases can be organized and how they can manage queries is still the realm of research projects [Marcus 1997].

Conventional databases offer features which new multimedia databases should preserve as much as possible. These are:

- Data persistence
- Standard operations on data (for example, for relational databases)
- Schema catalog
- User views for the data
- Integrity
- Security and access control
- Synchronization
- Multiuser capable

However, multimedia databases are different in the following respects:

- We would like to preserve all properties mentioned above, but the amount of stored data is several orders of magnitude larger
- The multimedia data types are different: there is a semantic structure and synchronization we would like to preserve
- High bandwidth is needed for accessing the information, possibly in real time
- We want to perform “information (semantic) retrieval” not just “data retrieval”
- Several forms of storage should be managed efficiently (hard disk, DVDs, tapes)
- Different types of playing devices should be serviced
- Interactive queries should be possible
- Fuzzy queries should be serviced iteratively

An audio and video database cannot be accessed as a textual database. In textual databases every word in the database can be counted and can be used as index. In audio and video databases we would like to know if a certain fragment of music is in the database, or if a certain image is in a

video. We need to store metadata describing the data, but we also need to extract *features* from the material which would allow us to retrieve the information when we need it. One good example are music databases when we want to retrieve the name of a song of which only a portion has been recorded. This is useful for copyright management and monitoring radio stations. A few seconds of the music are analyzed and a "sound fingerprint" is computed. This fingerprint is then compared with the fingerprints of the music stored in the database. Several companies offer now this service in the US and Europe, even for cellular telephone users.

Apart from finding the information, there is the challenge of delivering the information as a stream, so that it can be replayed immediately. In the next section we consider this problem, and in section 3 a research prototype of a streaming video database is described.

## 2 Streaming Technology and Quality of Service

There are two basically different ways of reading data from to storage: read the complete data file, and then display or use it, or read the data byte for byte, starting to display or use the data as soon as it is received by the application. In the latter case, we have a continuous stream of bytes – i.e. data streaming. The same applies for data being written: a database can capture the data complete in main memory, and then record it, or can start writing as soon as the first bytes arrive.

Streaming is the only sensible alternative when the data is retrieved through a network and consists of sequential information stored in huge files, as is the case of music and video. If the harddisk [Özden 1996], the server, the network, and the client are all fast enough, playing music or video is not so difficult. However, there is usually a "weak link" in this data retrieval path. The Internet, for example, does not provide any guarantees on the travel time of packets. Delays in the network are unavoidable and this can be heard, or seen, when data packets do not arrive timely.

In the ideal case, all components in the data retrieval path should provide a "Quality of Service" (QoS) guarantee. We would know what is the worst-case access time for a file from storage, what is the maximum processing time in the server, which is the maximum delay for a packet in the network, and what is maximum display time in the client. QoS bounds have to be provided for each single link and also for the software operating the hardware [Steinmetz, Nahrstedt 2004]. In the extreme case, streaming is not possible if the operating system is not a real time operating system. The harddisk can be fast enough, for example, but if the operating system decides to do garbage collection at the wrong time, the temporal deadlines would be missed.

There are many intricate algorithms for computing QoS bounds for streaming data. However, most of the data being streamed today is sent using a "best effort" approach. Hardware and software are adjusted to work as fast as the receiver can play the data, and are tuned to minimize delays, but no effective time guarantee can be provided. The whole system and the network does its best in order to provide timely delivery of data packets – without any further assurances.

## 2.1 Stream Buffering

The conventional approach for delivering audio and video streams is to apply the best effort approach and use a buffer for counteracting the network delay jitter [Hunter 1997]. A FIFO queue is filled on the server side before starting to transmit (for example, several seconds of audio data) and another FIFO queue is used on the client side. Whenever the network stalls from the server point of view, the server buffer fills. When the network packets arrive faster than the player needs them, the client buffer fills. If the network delay stays within certain bounds, the user does not notice the packet delays and network jitter is ironed out.

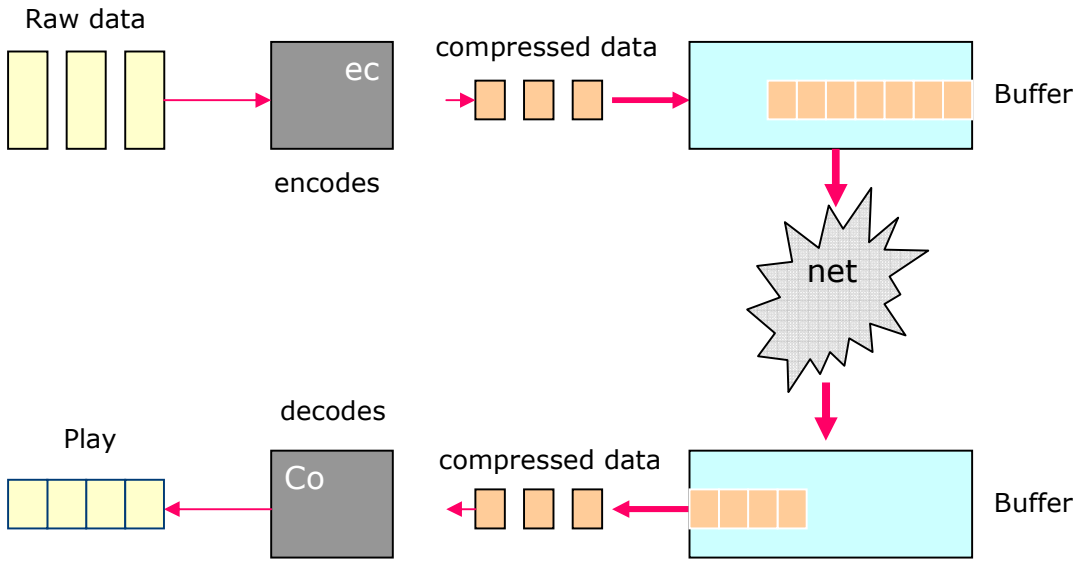



Figure 4: Streaming data over a network

Fig. 4 shows the usual approach: raw data from a recorder or video camera is processed and a Codec (coder and decoder) reduces the necessary bandwidth by compressing the data. In the case of video some kind of standard format could be used. The server buffer fills for a few seconds and then transmission starts. On the client side, the packets are captured first in a buffer and then the compressed packets are given to the decoder part of a Codec, which send the reconstructed data for replay. In this case, the information is being coded on the fly. If the information has been stored in a database, it can be stored in the compressed form (for example in MP3 format). The middleware (not shown in the Figure) takes care of error correction and of handling all the data transfer protocols. If, for example, a packet does not arrive to the client, a retransmission could be needed. On the Internet, both the TCP and UDP protocol can be used for packet transmission. TCP operates with an acknowledge, which makes necessary a retransmission when packets are lost. UDP operates on a best effort basis, sends packets and does not wait for an acknowledgement.

## 2.2 Streaming Techniques

There are several possibilities for streaming audio and video over a computer network: a point-to-point connection can be opened, a multicast transmission can be started, or the data can be broadcasted. A point-to-point connection is the usual case when downloading data from the Internet. In that case the client is almost fully absorbed by the retrieval operation, while the server side can employ a single machine, or a network of dedicated servers to provide the data. The multicast approach requires the definition of a private virtual subnetwork on top of the real network, as for example the MBone, which is a virtual network on top of the Internet. Multicast packets are sent to all nodes on the virtual network and the nodes retrieve them according to need. Broadcasting data is only done on dedicated channels, such as digital or cable TV.

The most popular commercial audio and video players (Virtual Media Player and Real Networks) use point-to-point connections. Distributing data can be made more efficient, if in a network of servers, the computer nearest to the end-user can be used to transmit the data. This is the approach used by Akamai Technologies, the largest content distributor in the world. Akamai has an installed base of 14,000 servers, distributed around the world. Companies can distribute content through Akamai. When a user connects to the company's server, the request is serviced by Akamai, which locates the best server in the world for handling the transaction. That server receives the information through high-speed lines, and caches the data for eventual new requests. Since traffic on the Internet tends to be bursty, this caching approach saves bandwidth and distributes the needed files on Akamai's network. In some public domain projects, point-to-point connections have been used to distribute content by building an ad-hoc network of peers which pass down the needed data to the end user.

Multicasting on the MBone had some popularity in the 1990s, but has not been developed further mainly due to the rapid increase in the available bandwidth for point-to-point connections and the efficiency of peer-to-peer and network caching techniques.

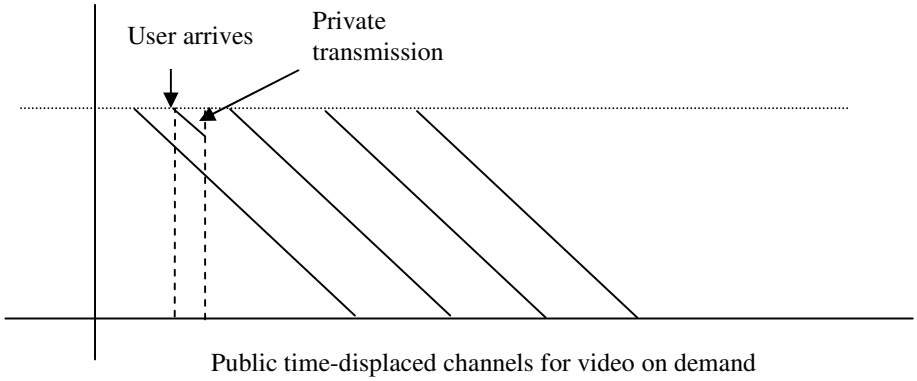



Figure 5: Video-On-Demand: The same video starts periodically on four channels, each channel is displaced on time relative to each other. A user which arrives late, can obtain the beginning of the video through a point-to-point connection and can cache the temporally nearest channel on a setup box.

For video-on-demand over broadcasting channels, some transmission strategies have been developed over the years. The simplest one is using several cable channels, transmitting the same video in every channel, with a fixed delay of, for example, five minutes. Someone willing to watch a video needs to wait a maximum time of five minutes to start receiving the video. If the TV setup box is caching data continuously (always the first five minutes of a video), even this delay can be suppressed. This is the near-video-on-demand strategy. For real video-on-demand several techniques have been patented. One technique consists in starting a point-to-point transmission whenever a user arrives, caching at the same time one of the delayed channels described above. If the user arrives, for example, two minutes after the video has started (Fig. 5), this two minutes are sent point-to-point to her, while the setup box stores the public channel, starting with minute three. Once the two minutes from the point-to-point connection have been played, the setup box takes over and delivers the rest of the program, while it continues to store the delayed channel in advance, as a kind of buffer.

On the Internet none of these conventional video-on-demand techniques are used. The Akamai approach is the dominant one, and is probably the one used by T-Online for its online-video store.

Streaming data from one server to many users can require a careful organization of the storage in the hard-disk. Several hard-disks or a RAID unit could be needed in order to service all streaming requests. Special compresson methods could be also needed, in order to minimize the bandwidth. Polycom video-conference cameras, for example, perform coding and decoding of video in hardware and can be used as video-telephones.

The choice of operating system is also an issue, in the case of the server. High-performance real time operating systems could be needed. Reducing the network delay can be also done only by using the Akamai approach. In summary, there are many problems associated with streaming data, which touch all levels of the computational pyramid. A database system cannot solve all of them, it can only help.

# 3 Theory: Streaming Databases for audio and video

There are no widely available commercial streaming databases for audio and video. There are some research prototypes, one of which will be discussed in this section, and there are streaming databases for other purposes. StreamBase is a company which offers a streaming database for stocks quotations. The stream of data is captured from on-line stocks services, is processed on-the-fly by the database, and the streams are stored, in indexed form, while at the same time the stream is delivered to the end user. StreamBase also offers its streaming engine for storing data from sensor networks and for the military. There is no StreamBase product for audio or video.

In this section we look at the structure of a prototype of a video database management system, with a streaming engine, developed at the University of Purdue in the USA.

### 3.1 The VDBMS System

VDBMS (Video Database Management System) was developed with the intention of providing a database which could be searched according to different criteria, so that videos could be retrieved by content and could be streamed to the viewer [VDBMS 2005, Walid 2000]. VDBM was written using open source code from the Predator/Shore relational database community. VDBMS includes a real time stream manager for servicing requests [Aref 2002]. This differentiates it from other systems, in which no streaming engine is provided. Fig. 6 shows a diagram of the different components of VDBMS.

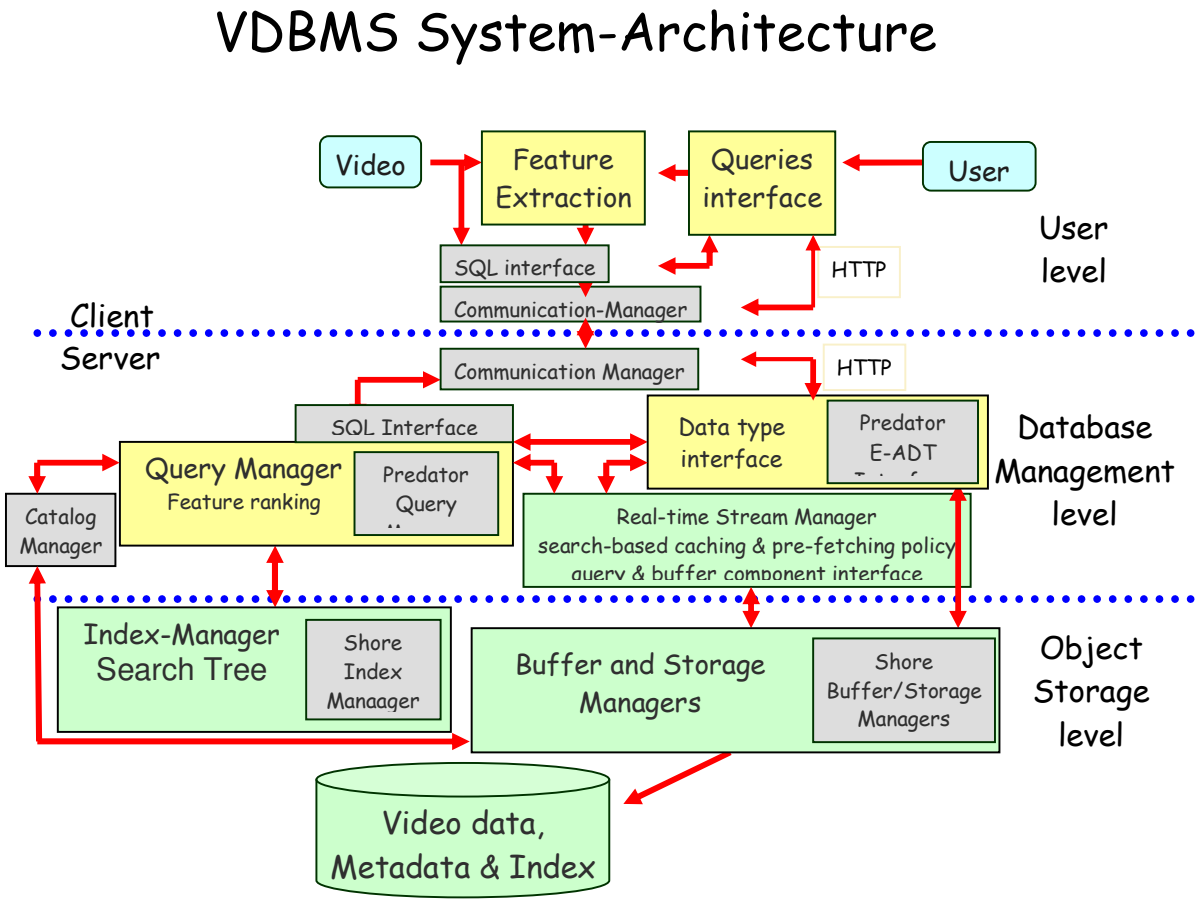



Figure 6: The architecture of Purdue's VDBMS system [Aref 2002]

In VDBMS a video is stored as raw data, but when the video is captured a set of features is computed in real-time. One important feature is the beginning and end of a shot. Another set of features is the color histogram of each frame. The color of the maximum blob is also recorded, and so on. The set of features is extensible and can be extended at any time. A recorded video is passed down to the storage manager which stores it in a harddisk.

More interesting is what happens when a user requests a video. A video can be retrieved by name, or by providing keywords which are compared against the metadata fields. But the user

can also request to see the video in which a specific frame is contained. The user query is an image, which is preprocessed by the system. An SQL query is generated and is passed down to the Query Manager. This starts a search using the indexed data, and the result set of candidates is ranked by relevance (as in Google searches). The buffer and storage manager retrieves the data and starts the stream manager which hands down the data to the data type interface and communications manager. The data type interface decodes images, or videos.

As can be seen in the figure, there are three levels: user, database management, and object storage level. The system does not provide QoS time bounds, but makes a best effort to provide the data in real time.

Fig. 7 shows an example of the user interface for the system. The query is represented by an image (in the middle, to the right). The system shows the ranked result set sorted by relevance (images in the lower left). The user can then request to see any one of the ranked videos (lower right). Several menus and buttons allow the user to select the kind of color space, some filters for texture, etc. The SQL query is visible in a text window (upper left).

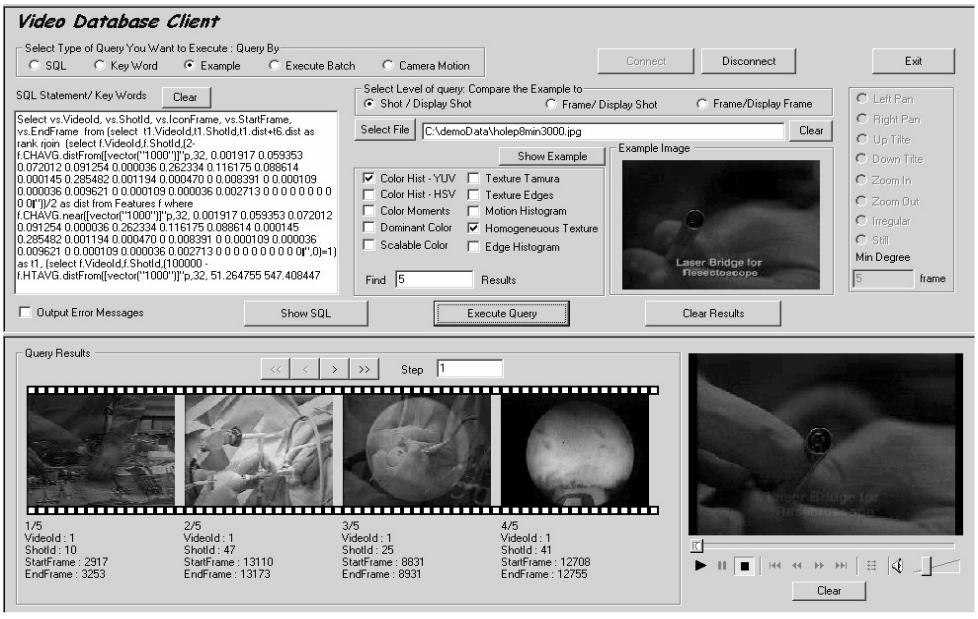



Figure 7: The user interface for VDBMS queries

Already this simple example shows that the main problem is indexing the data (that is, defining the features to be computed), and providing a streaming engine for viewing the data. A case could be made that the streaming engine should be provided by the operating system, but in the case of VDBMS, caching and prefetching of data can be made more efficiently since we know that the data is video data.

VDBMS was written to be compatible with MPEG7. Meta-annotations can be linked with the video data using XML. The features extracted by VDBMS are organized in a search tree in a high dimensional space. When a user provides an image, its features are computed on-line and a

search is done to localize the nearest neighbor in the database. The search strategy is therefore somewhat naïve and relies entirely in a good set of extracted features. The first version of VDBMS used such features as the color histogram, the texture of each frame, and parameters for motion. Query by example is thus supported, and the user can refine the query adjusting parameters in the user interface.

Regarding streaming, the database supports multiple users. Individual viewers are handled by a multithreaded system. The stream manager handles all streams and caching of common portions of videos. When several viewers are receiving the same video, it is not necessary to go back to the harddisk storage every time that a video shot is needed.

VDBMS and similar experiments, remain as a research prototypes and have not transformed into commercial products.

## 3.2 Streaming Databases in General

Streaming databases have been developed in recent years, not for audio and video, but for handling streams of numerical data. Examples are data from sensor networks and stock quotes. Streaming databases store and deliver the information as soon as it is received.

One can think of a streaming database as a filter in which the queries have been stored in advance. For example, if one user would like to receive a notification every time the IBM stock falls below a certain threshold, and another wants also a notification as soon as Apple stocks goes above another threshold, both queries would be stored in the streaming database and the incoming data will be filtered using the queries. As soon as an event happens, the users are notified, according to their query.

Streams require “streaming algorithms”, which are defined as algorithms which can handle a long sequence of data using polylogarithmic time in the length of the stream.

StreamBase, the company mentioned before, is the only current commercial streaming database. One of StreamBase’s founders is Michael Stonbreaker, who has started other successful database companies. StreamBase claims: “The StreamBase stream-processing engine is designed to (handle)… real-time data management problems with a new infrastructure platform for capturing, integrating, understanding, and reacting to business events as they occur. With the StreamBase platform, financial institutions can quickly build state-of-the-art applications that process and analyze data as fast as the data arrives and without sacrificing the richness or the business value of the analysis.” (www.streambase.com).

For audio and video, there is a need to extract features “on-the-fly”, when the data is arriving to the server, but there is no need to transmit the audio or video feeds immediately. Therefore, streaming databases for audio and video are essentially different from streaming databases for sensor networks. Semantic queries are the main feature of video and audio databases, and also the main problem which has to be solved.

# 4 Practice: Streaming with conventional databases

Although much research is being done on multimedia databases and specially, on databases for audio and video, the reality of the market, at this moment, is that conventional databases have been enhanced with capabilities that allow the users to stream audio and video from relational databases such as Oracle or MySQL [Marcus 1996]. Some standards have been defined for this purpose. This section describes one such approach, and our implementation of a streaming architecture based on JDBC for the Oracle database system. First, I describe the E-Chalk system and its three streams, and then I proceed to explain how the system was coupled to Oracle. This example is representative for the practical side of streaming from relational databases using a best effort approach.

## 4.1 The E-Chalk system

The main idea of the E-Chalk system is to provide the functionality of the traditional chalkboard using a large contact sensitive computer screen, but enhanced with all the capabilities of a digital system [Friedland 02, 03]. An electronic blackboard should be as easy to use as a traditional one. The only interface to the electronic board should be a stylus, instead of a piece of chalk.

When an E-Chalk session is started, the server computer starts storing and sending three streams: the board events, the audio channel, and an optional video channel. The three streams can be accessed from a Web page, by starting the E-Chalk client, which is a collection of three client Applets, one for each stream. The streams are synchronized by the audio time stamp. Therefore, an E-Chalk session can be recorded completely, but the quality of the reproduction of the board on the client side is much higher, since the board is repainted with the full resolution of the computer screen. Fig. 8 below shows schematically a teaching scenario: a lecturer teaches to a live audience; the E-Chalk server transmits the audio, video and board streams and stores an archival copy. A remote viewer watches the class using an Internet browser.

Fig. 9 is a screenshot of an actual lecture, as seen by a remote viewer in his browser. The look and feel of the screen is that of a lecture on a good blackboard. The use of color helps to emphasize some important aspects of the lecture. The screenshot shows that a computer connected to the contact sensitive screen starts three servers: a board, an audio and a video server (which are Java applications). The servers connect with their respective three clients on the user computer, which are Java Applets. Three streams are sent from the lecture room to the viewer. Each stream is handled by a server-client connection. The audio stream provides the timing signal for the synchronization among the streams.

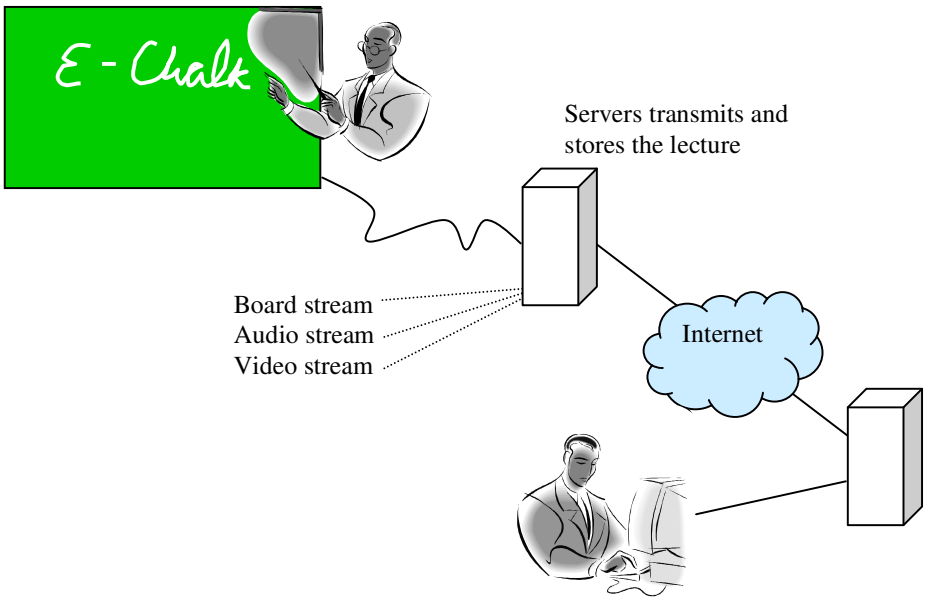



Figure 8: The E-Chalk system. A lecturer writes on an electronic blackboard. Audio, video, and board contents are stored and are streamed through the Internet. Remote viewers watch a lecture using a Java enabled browser.

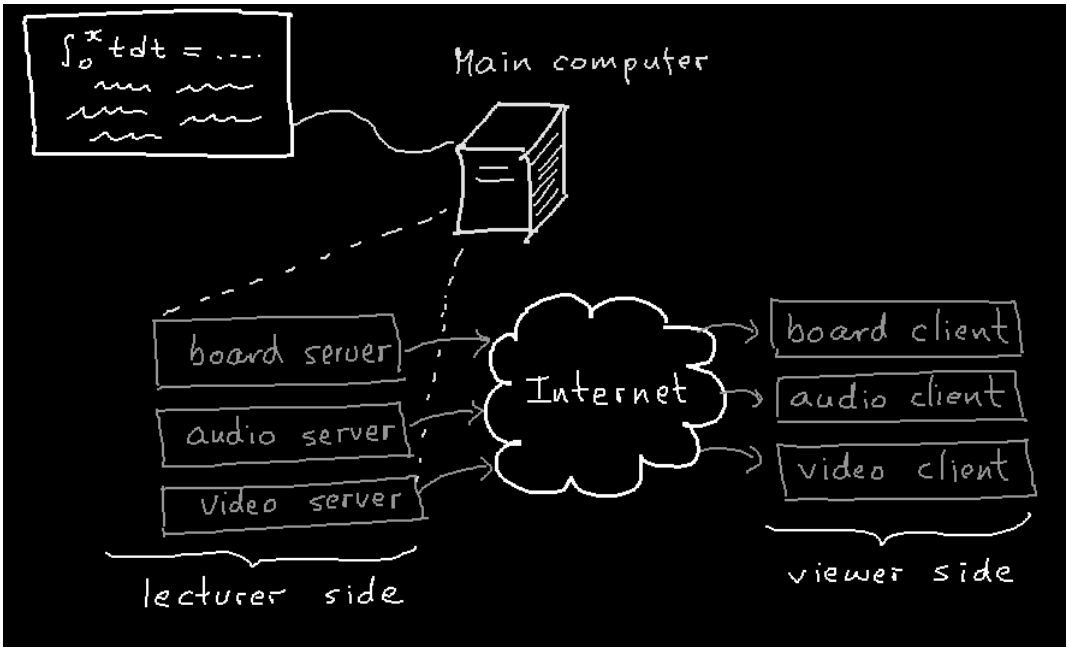



Figure 9: A real E-Chalk lecture about the client-server architecture of E-Chalk

The main feature of E-Chalk is to go beyond the original blackboard metaphor and provide "intelligence and information on demand". This means, that a series of special programs is running in parallel with E-Chalk and is watching the user interacting with the screen. Certain programs can then become active when certain conditions are met. For example, a program can observe the handwriting of the user and if a mathematical formula is entered, and if the user

writes a special stop symbol, the program can interpret the formula. If it is an equation, it can solve it. This capability has been already implemented in E-Chalk, using Mathematica as the mathematical equation solver [Tapia 03].

An electronic classroom has been installed at FU Berlin. The classroom features a four screen rear-projection system, which provides a 6 meter long electronic blackboard, as shown in Fig. 10. E-Chalk runs on this system, which among other things, can animate algorithms using handwritten input data [Esponda 2004b, 2004c]. A camera (not visible in Fig. 10) captures the image of the lecturer and streams this data to remote viewers.

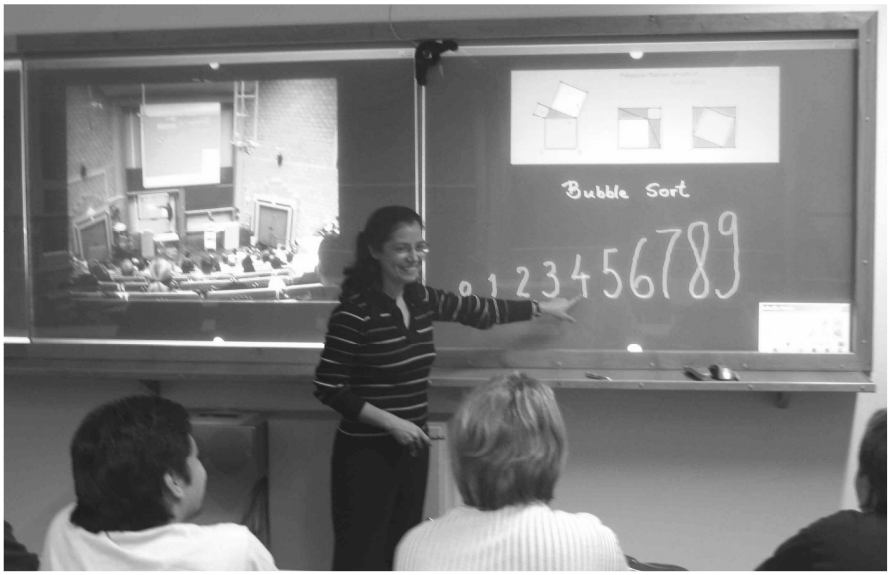



Figure 10: The electronic classroom at the FU Berlin. Algorithmic animations can be started by the lecturer using handwritten input.

## 4.2 The Oracle Server and Java Interface

When E-Chalk started to be used for many different lectures, we suddenly had the problem that storing the streaming files in the file system was not enough. The lecturers wanted to have an automatic storage mechanism which would make sure that students could find the lectures easily in our web site. Also, the Windows file system has no provisions for doing fast forwarding or rewind of computer files (as needed, when streams are being played). We decided to couple E-Chalk with a relational database, in our case with the Oracle server at our department.

A plug-in for E-Chalk was written, which allows E-Chalk to store streams as Binary Large Objects (BLOBs). Metadata associated with the lecture (name of the course, lecturer, topic, duration, etc.) can be stored together with the BLOBs. It is then easy to access the information by referring to the metadata. Oracle BLOBs can also be read sequentially, or starting from a certain byte offset. It is then trivial to extend the E-Chalk client to keep track of the bytes being extracted from the BLOBs, so that fast-forward or rewind operations can be implemented efficiently.

There exists an standard for interfacing relational databases with Java, it is the JDBC interface (whose name is usually interpreted as Java Database Connectivity, although JDBC is a

trademark). JDBC allows a Java programmer to open a relational database and send SQL queries. The results are transformed into Java data types, and are returned to the calling program. Java applications and Applets can then exploit the full functionality of databases such as Oracle and MySQL.

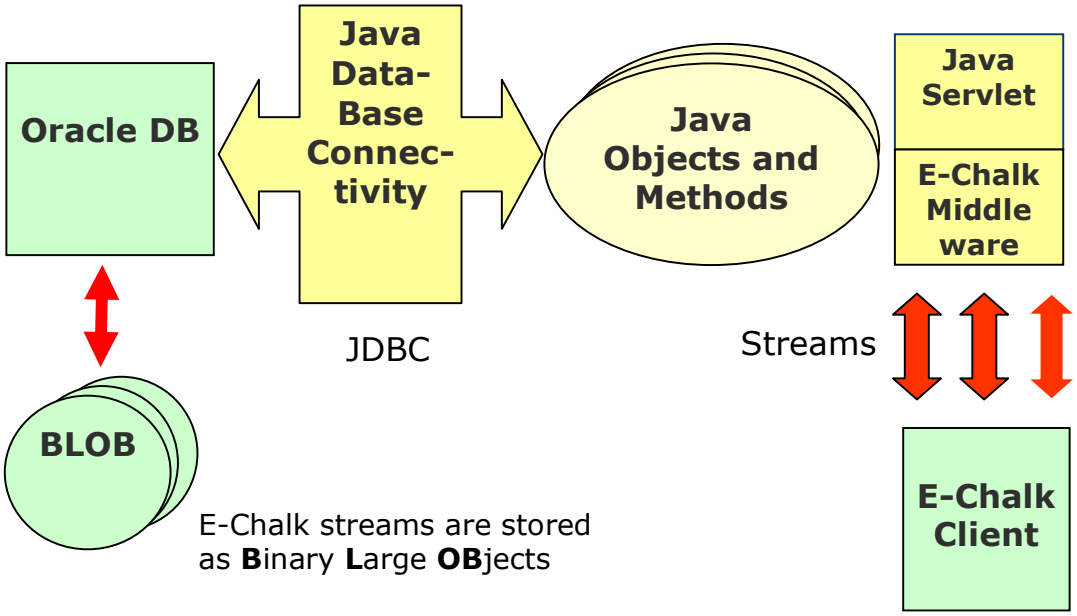



Figure 11: The JDBC API mediating between Oracle and E-Chalk.

Fig. 11 shows how the E-Chalk system was interfaced with the Oracle database. Oracle manages the metadata and BLOBs containing the streams. The JDBC API provides a set of Java objects and methods which can be used by Java programs to send SQL queries and receive result sets. A servlet was written, which using such data objects, mediates between Oracle and the E-Chalk code which implements streaming from the server to the E-Chalk client. The client is an Applet in the viewer's machine. Server and client connect through sockets. The JDBC API lets the Java servelet see the Oracle database as a source of streams. The BLOBs are read byte for byte, and are transferred sequentially to the client. The client can request that the streams start at any desired byte offset, so that fast-forwarding and rewinding are possible.

In the next section, we take an inside look at the JDBC interface and how to implement streaming using it.

## 4.3 The Java Database Connectivity API

JDBC is the Java implementation of the ODBC (Open Database Connectivity) standard. ODBC is an interface for sending commands to database systems. A driver is needed, and must be provided by the database vendor, for example [Oracle 2005]. The driver is written in native code.

When using JDBC, first the driver must be loaded using the Class.forName() method:

```
Class.forName("oracle.jdbc.driver.OracleDriver")
```

Once the driver is loaded, an instance of a connection object “con” is created, for example:

```
Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@server:port:serverinstance",
username, passwd);
```

One  a connection object has been returned, the methods associated with “con” can be used to send SQL statements to an Oracle database. With “executeUpdate” the open database is asked to execute SQL commands passed as strings, for example:

```
Statement stmt = con.createStatement();

stmt.executeUpdate("CREATE TABLE Sells " +
      "(bar VARCHAR2(40), beer VARCHAR2(40), price REAL)" );
stmt.executeUpdate("INSERT INTO Sells " +
      "VALUES ('Bar Of Chocolate', 'Becks', 2.00)" );
```

The statements above are enough to define a new table “Sells”, containing the fields bars, beer, and price. An Insert statement can add a row of concrete data to the table.

As can be seen, the JDBC interface is a direct way to open a textual channel to the database, which can be used to send commands and receive results. An example of a query providing data back to Java is:

```
String bar, beer ;
float price ;

ResultSet rs = stmt.executeQuery("SELECT * FROM Sells");
  while ( rs.next() ) {
    bar = rs.getString("bar");
    beer = rs.getString("beer");
    price = rs.getFloat("price");
    System.out.println(bar + " sells " + beer + " for " + price + "
Dollars.");
                        }
```

Here, a SELECT command returns a ResultSet “rs”, which can be analyzed using the “getString” and “getFloat” methods. The method “next” checks if there are still records in the result set.

These simple examples shows that accessing a Database from Java is straightforward, when a JDBC driver is provided by the vendor. JDBC allows the user to define atomic transactions and handle errors too.

In our case, we are interested in storing the three E-Chalk streams as BLOBs. A database is defined in which each record includes the three BLOBs (for audio, video, and board events), as well as metadata, such as course, date, time, length, and keywords. Lectures can then be retrieved by date or course. Moreover, the database can provide the BLOBs as streams, so that a lecture can be streamed to the viewer once the transmission buffers in the server have been filled.

The Oracle JDBC driver supports the streaming of BLOBs. In E-Chalk, the board events are

stored in ASCII (coordinates of board strokes and other events), the audio and video files in a proprietary binary format. In order to read BLOBs the extension classes in `oracle.sql.BLOB` and `oracle.sql.CLOB` (character BLOB) are used.

Once a result set has been generated (by sending the appropriate SQL command to the database) the ResultSet.getObject() method returns an oracle.sql.BLOB or oracle.sql.CLOB object. The BLOB contains raw data which can be streamed as binary raw data. To do this, the getBinaryStream() method of the oracle.sql.BLOB object is used. The BLOB can be read byte by byte, and if a connection has been opened, it can be streamed byte by byte to a remote computer.

In our implementation, board, audio and video have been defined as entities, with the following structure:

```
Video {[ id : Int, name : String, size : Int, object : BLOB, x : Int, y : Int,
resolution_x : Int, resolution_y : Int, bandwidth : Int, length : Int, format
: String]}

Board {[ id : Int, name : String, size : Int, object : BLOB, x : Int, y : Int,
bandwidth : Int]}

Audio {[ id : Int, name : String, size : Int, object : BLOB, bandwidth : Int,
format : String]}
```

A video, for example, has an ID, a name, a total byte size, a resolution, the minimal bandwith to transmit it without interruptions, and some other parameters.

## 4.4 Client and Servlet interaction

Now, we can describe in more detail how the coupling between E-Chalk and Oracle was implemented by a student working with the E-Chalk team, so that the database provides the streams needed for replay of a lecture. Fig. 12 shows a diagram of the control and data paths between the E-Chalk software and the Database.

E-Chalk lectures are played by starting a browser. The browser can send a query, which is transformed into an SQL query by the Java Servlet written with JDBC. The SQL query returns a result set. The servlet passes the result set object, and the connection to the E-Chalk middleware in charge of streaming. The offset is the point from which the user wants to start replaying the lecture. A lecture can be replayed from the beginning (offset 0) or from any point in the middle. The offset is selected using a slider in the Applets started by the browser as clients. The E-Chalk middleware can then read the appropriate board, audio and video BLOBs from the database, just by accessing the result set. The middleware opens streaming channels to the E-Chalk client, and the transmission starts.

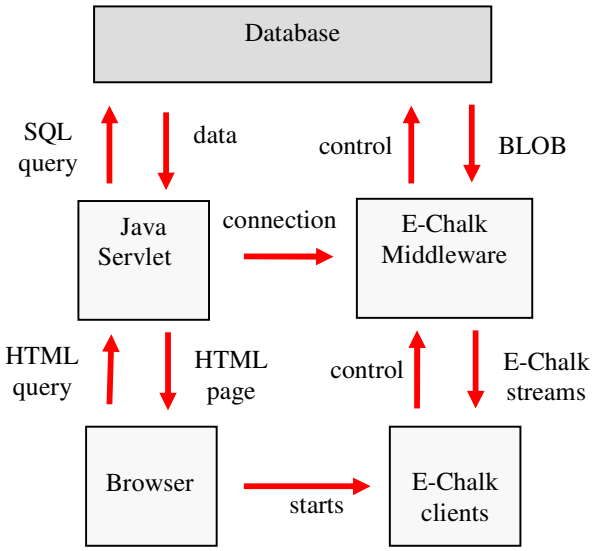



Figure 12: Servlet and E-Chalk interaction with the database

Notice that fast forwarding and rewind are efficient, since a BLOB can be read starting from a byte offset. If the user rewinds, the client notifies the middleware, which in turn just restarts the input streams from the needed offset in the BLOBs. The byte offset is proportional to the time mark used in E-Chalk, so that it is possible to jump to a given time in the lecture using the client slider.

## 4.4 Fusing the video and board channels

In future versions of E-Chalk, the board and video channels will be fused, so that only two streams will be transmitted: audio and board/lecturer channels. E-Chalk does not just transmit video of a lecture because the board contents is difficult to read due to the compression used. E-Chalk transmits a vector graphics image of the board which can be reproduced with very good quality on the client computer. However, there arises a problem of divided attention: the lecturer is visible on the video, while the board is visible on another window. It would be better if the lecturer could be segmented, cut from the video, and could be superimposed on the board vector graphics.

This is what the newest version of E-Chalk achieves [Friedland 2006]. The video channel is segmented. The lecturer is identified by its movement against the background. This allows the computer to detect the colors of its cloths, face and arms. Doing color segmentation, together with some additional postprocessing steps, it is then possible to cut the lecturer from the original video. The lecturer is transmitted (at 20 frames per second), and is superimposed on the board. The lecturer image is transmitted as a semitransparent object, so that the board contents are not occluded by his or her body. In future versions of E-Chalk, the viewer will be able to dim the lecturer in or out, as desired.

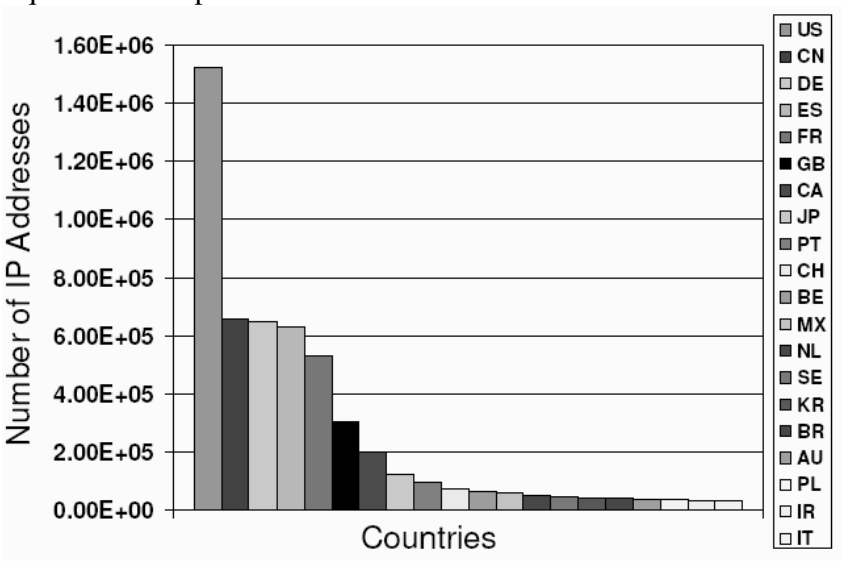
Note that we will still send three streams, since the lecturer image is a video stream superimposed on the board stream. The database interface remains the same.

## 4.4 Live Streaming today

The most comprehensive study of live streaming from a dedicated network was done by Sripanidkulchai et al [2004]. They analyzed the streaming traffic in the Akamai network and reached the following conclusions:

- Most of the streaming traffic is for audio. Only 7% of the measured streams were video.
- Few live streams concentrate most of the traffic, most of the data is seldom streamed
- Demand for streams is concentrated in time ("flash crowds").
- Half of the domains use TCP as transport protocol. UDP is not the dominant protocol.
- Most of streams reach more than ten countries.
- The diversity of the software used for receiving streams is large.

The popularity of streaming is also evident from the number of hosts accessing the top 40 streams during the period of the study (Fig. 13). Millions of computers in the US, China, and Germany, among others, requested the top 40 streams.




In the Akamai network there is not an underlying database. Streaming is done by streaming engines (Real Networks, Windows Media, etc.). It can be safely said that most of the streaming data transmitted today has a flat structure, as files in the file system.

# 5 Conclusions and Outlook

I can summarize this review of database architectures for streaming audio and video with a list of the main ideas discussed in the text:

1 The price of a stored bit of data has been falling exponentially in the last years. Hard-disks double their density every year, while the price of the unit falls by 50%.
2 Individual users already produce many Gigabytes of data every year, which in the future will be handled digitally and will be transmitted using computer networks. There is a need

to provide a storage structure for all this data Tsunami.

3 Traditional databases were designed under assumptions which are valid for storing and accessing mostly textual information. Support for other kinds of data formats has been usually sparse, and has been limited to Large Binary Objects.

4 Streaming audio and music on the Internet is done today mainly through point to point connections. Streaming requires buffering, compression, and decompression of data.

5 Streaming databases for audio and video have been investigated as research projects. The main idea is to provide semantic access to the data, so that the user can request a video in which a certain person appears, or which contains a certain image, or which corresponds to a textual description. The challenge today is how to access the information, not how to store the data.

6 Streaming databases for audio and video have not yet matured as commercial products. Current streaming databases are geared towards numerical data, such as that coming from stock markets or sensor networks. The first commercial products have appeared.

7 Conventional databases can be extended to support streaming. The ODBC standard provides an interface between programming languages and databases. JDBC is the Java variation of ODBC. With JDBC it is possible to search for audio and video, using metadata annotations and retrieve the streams stored as BLOBs.

8 In the E-Chalk project an Oracle database server has been used as repository for the E-Chalk streams. A servlet was written, which allows the user to search for lectures according to metadata, and then stream the BLOBs to the viewer. The Oracle database supports BLOB streaming.

9 In the near future, additional extensions to conventional databases will allow them to handle most of the streaming audio and video requirements of low volume servers. The servers will be embedded in a private network of servers.

10 Streaming databases for audio and video, which allow semantic querying, will provide the solution to the data access problems of the future.

## References


[Adams 2004] David Adams, "1.5 Billion Mobile Phone Subscribers", New Mobile Computing, http://www.newmobilecomputing.com/story.php?news_id=4124.

[Aref 2002] W.G. Aref, A.C. Catlin, A.K. Elmagarmid, J. Fan, J. Guo, M. Hammad, I.F. Ilyas, M.S. Marzouk, S. Prabhakar, A. Rezgui, S. Teoh, E. Terzi, Y. Tu, A. Vakali, X.Q. Zhu, "A Distributed Database Server for Continuous Media", *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, 2002.

[EETimes 2005] EETimes Press Release: 07/11/2005.

[Esponda 2004a] Margarita Esponda, "Trends in Hardware Architecture for Mobile Devices", Technical Report B-04-17, Freie Universität Berlin, Fachbereich Mathematik und Informatik, November 2004.

[Esponda 2004b] Margarita Esponda, R. Rojas, „Learning Algorithms with an Electronic Chalkboard over the Web", in W. Liu, Y. Shi, Q. Li (Hrsg.), *3rd International Conference on Web-based Learning 2004*, Tsinghua University, Beijing, China, Springer-Verlag, Berlin, 2004.

[Esponda 2004c] Margarita Esponda, „Algorithmic Animation in Computer Science Education with the Flashdance System", *2. Workshop Grundlagen Multimedialen Lehrens und Lernens*, Berlin, 15.-18. März 2004.

[Friedland et al. 2002] Friedland G., Knipping L. and Rojas R., "E-Chalk Technical Description," Technical Report B-02-11, Faculty of Computer Science, Freie Universität Berlin, May 2002.

[Friedland et al. 2003] Friedland G., Knipping L., Rojas R. and E. Tapia, "Das E-Chalk System: Stand der Entwicklung," Technical Report B-03-03, Department of Mathematics and Computer Science, Freie Universität Berlin, February 2003.

[Friedland 2006] Friedland, G., *Adaptive Audio and Video Processing forElectronic Chalkboard Lectures*, PhD Thesis, Freie Universität Berlin, October 2006.

[Gemell 2002] J. Gemmell, G. Bell, R. Lueder, R. Drucker, C. Wong, „MyLifeBits: Fulfilling the Memex Vision", *ACM Multimedia* 2002.

[Ghandeharizadeh 1996] Shahram Ghandeharizadeh, "Stream-based Versus Structured Video Objects: Issues, Solutions, and Challenges", in [Subrahmanian 1996], pp. 215-235.

[Marcus 1997] Sherry Marcus and V.S. Subrahmanian, "Foundations of Multimedia Database Systems",*Journal of the ACM*, Vol. 43, No. 3, May 1996, pp. 474-523.

[Hunter 1997] Jane Hunter, Varuni Witana and Mark Antoniades, "A Review of Video Streaming over the Internet", DSTC Technical Report TR97-10, August 1997, http://archive.dstc.edu.au/RDU/staff/jane-hunter/video-streaming.html.

[Marcus 1996] Sherry Marcus, "Querying Multimedia Databases in SQL", in [Subrahmanian 1996], pp-263-277.

[Oracle 2005] Oracle 7 JDBC Drivers, http://www-wnt.gsi.de/oragsi/Docs/jdbc.htm

[Özden 1996] Banu Özden, Rajeev Rastogi, and Avi Silberschatz, The Storage and Retrieval of Continuous Media Data, in [Subrahmanian 1996], pp. 237-261.

[Sripanidkulchai 2004] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang, "An Analysis of Live Streaming Workloads on the Internet", IMC`04, October 25-27, 2004, Taormina, Sicily, Italy.

[Steinmetz 2004] R. Steinmetz, L. Nahrstedt, *Multimedia Systems*, Springer-Verlag, Berlin, 2004.

[Subrahmanian 1996] V. S. Subrahmanian, Sushil Jajodia (Eds.): *Multimedia Database System: Issues and Research Directions*. Springer 1996, Springer-Verlag, Heidelberg.

[Subramanian 2000] S.R. Subramanian, "Multimedia Databases - Issues and Challenges", IEEE Computer, December 1999/January 2000, pp. 16-18.

[Tapia et al. 2003] Tapia E. and Rojas R., "Recognition of On-line Handwritten Mathematical Formulas in the E-Chalk System," *Proceedings of the Seventh International Conference on Document Analysis and Recognition* (ICDAR), Edinburgh, Scotland, August 3-6, 2003.

[VDBMS 2005] "The vdbms Project – Advancing Multimedia Database Technology", http://www.cs.purdue.edu/vdbms/.

[Walid 2000]Walid G. Aref, Ann C. Catlin, Jianping Fan, Ahmed K. Elmagarmid, Moustafa A. Hammad, Ihab F. Ilyas, Mirette S. Marzouk, Xingquan Zhu, "A Video Database Management System for Advancing Video Database Research", Medical Information Systems 2000.