Information integration among Heterogeneous and Autonomous Applications

Benabdelkader, A.

Citation for published version (APA):
Chapter 4

MegaStore: Advanced Web Databases for Music Industry

4.1 Introduction

The MegaStore\textsuperscript{1} system described in this chapter, aims at the design and set-up of the necessary database structure and platform architecture for advanced e-commerce applications, and in specific addresses the CD and music industry. Unlike most existing systems, the database design of MegaStore is general enough to include additional information about the music. Furthermore, the search engine can benefit from the additional stored information about the music composer/performer and lyrics for the titles, among others. Similarly, the storage of complete audio/video clips can serve for a future extension to this system towards what is known as on-line Music from the Wall, with a small effort and reduced cost.

\textsuperscript{1}From the usage point of view, the Internet-based CD shopping system, called the Virtual MegaStore, consists of a front-end system with two main components. The first component being the Internet-Shop, that can be accessed by all Internet users and the second component constituting a so-called Shop-in-a-Shop. The Shop-in-a-Shop interface can be installed inside an existing physical music store and it offers the store keepers the unique and strong ability to immediately and at the run-time respond to the requests of customers visiting the music shop, through downloading the raw music data and producing CDs tailored to the customers requests.

\textsuperscript{1}The MegaStore front-end system is based on a back-end component that includes a distributed object-oriented database and a high performance server architecture. The database supports geographically distributed multi-media information and the designed extensible server architecture assures the required high data transfer rate and the short response time for on-line requests.

The proposed system architecture best suits the e-commerce application, by separating the public and general information from the private information, and supports the large data sets that need to be securely kept at predefined Internet sites. Furthermore, the necessary inter-stores communication requirements are studied and supported based on each identified activity within the system.

\textsuperscript{1}The Virtual MegaStore project has been supported by the Dutch HPCN foundation whose partners are the University of Amsterdam, the Frame Holding BV, and the International Music consortium BV.
4.1.1 E-Commerce Applications: Attempts and Aims

To support necessary requirements and flexibility to the buyers of different goods, advanced and efficient internet-based Electronic Commerce (E-Commerce) services must be designed and developed. In addition to the traditional user requirements for every application environment, the new developed system must properly address several efficiency and organization related issues, among which the data catalogues and information classification, short response time for on-line requests, high system performance, and high data transfer rates must be considered.

One major technical problem hampering the realization of suitable implementation for electronic shopping is the lack of possibilities to integrate a wide variety of data in a single coherent environment, which bases on a comprehensive system architecture and advanced database technologies [BAH 00, BAH 99, Atz 99, Atz 98, BBH+99]. In the context of web-based systems, several approaches have been investigated and worked out during the last few years. These approaches cover different domains of interest and address several application requirements. As such, these applications address different domains ranging from simple search engines that allow users to find information of their interest using a user friendly interfaces [NK 97, MMR 97] to advanced systems that manipulate multimedia information taking into account the emerging Internet technologies [BBO+99a, AAC+99, MH 97]. Nowadays, more e-commerce applications embracing electronic shopping via virtual stores are also emerging (e.g. CD Now\(^2\) and Amazon\(^3\)). However, Web-related systems are still involving the development of a large number of tools for data manipulation [BEM+98, FGN 98, Frt 99], which require a lot of efforts for their internal maintenance and operation. The work described in this chapter overcomes some of these problems and addresses the need to provide the user of electronic shopping with an environment through which he can experience as sufficiently close to real life shopping experience.

The structure of this chapter is organized as follow. In section 4.2, the music industry application is studied and analyzed, and the necessary requirements are identified. The section also describes the design of complex music library information, and accomplishes the MegaStore base schema structure using the UML notation and the ODL definition. Section 4.3 focuses on the general design of the server architecture for the MegaStore system, and mainly outlines the necessary requirements for: the Internet-Shop interface, the Shop-in-a-Shop interface, and the server architecture extension. In section 4.4, a brief descriptions of different audio/video music formats supported by the MegaStore system are presented, while section 4.5 addresses the manipulation of music data contents and their storage mechanisms. Section 4.6 describes the MegaStore advanced features and outlines the current implementation status of the system. Section 4.7 presents the Luisterpaal interface and the Music Sheet application: two applications derived from the MegaStore framework. And finally, section 4.8 concludes the chapter and introduces some possible extensions to the system.

4.2 Problem Analysis and Required High Level Architecture

The analysis of the music data to be stored and transferred between music shops and the users plays an important role in defining the MegaStore server architecture. The MegaStore

\(^2\)http://www.cdnow.com

\(^3\)http://www.amazon.com
network must be designed and build in such a way that it provides high bandwidth for huge amount of data transfer in a very short response time while taking into consideration the information visibility rights and security of access. Due to the music data specification and copyrights, a main characteristic of the MegaStore application is that the real music data is protected by certain music label centers, and neither can it be centralized in one common database, nor can it be freely or randomly replicated at different sites.

To properly support the requirements of the MegaStore environment, the designed system architecture involves the following components: [BAH 00]

1. The back-end system, including the database engine and the predefined networking connection between the MegaStore system components.

2. The front-end system, including (1) the Internet-Shop interface, where a user from home (or work place) can search for music, listen/watch to the audio/video clips, and order CDs, and (2) the Shop-in-a-Shop interface, where the music storekeeper can fetch on-line the real music data from its original source in order to burn at run-time the requested music CDs.

Under the specifications of the MegaStore system enumerated above, we have identified the need to design and build a dedicated networking infrastructure for this application, where the following aspects are studied:

- The music data is geographically distributed over the network
- Information about music is classified into two main categories: the general information stored at the Directory Services that can be accessed by any user connected via the Internet-Shop Interface and the raw music data that can only be accessed by the music storekeepers at music centers or burning towers.
- Depending on the user profile and authorization, only a part of the information can be accessed, and users need not to know about the data distribution.
- The real music data is securely transferred through a dedicated Network connection among music centers.
- The system must benefit from intelligent caching mechanisms, which is being further investigated at the University of Amsterdam, in order to improve the performance of the system [BA 98a, BA 98b].
- High bandwidth connection is necessary to handle raw music data that needs to be passed between the real music storage centers and the burning towers.
- Low latency network connection for the Internet-Shop interface is necessary to support the huge number of users expected to connect to the system.

4.2.1 Database Design

The database design for MegaStore is achieved in collaboration with the experts in the music industry domain. For design of the database schema, mnemonic names are chosen, taken from the music context, and thus objects are named for what they represent. This choice helps for instance the storekeepers to easily understand the elements of the database schema and use that in formulating their requests.
Based on the study and analysis of the data to be managed (i.e. stored/transfered) within the music industry, we have identified two main categories of information:

- The general information needed for the Internet-Shop interface, which represents the complete information about each song, artist, album, customer, order, etc. (except for the raw music data itself).
- The raw music data (real tracks) for the Shop-in-a-Shop interface that represents the real data used for the on-line CD burning.

The dynamism and flexibility of the Virtual MegaStore system mostly depends on its database design and how open it is in supporting several application domains with different structures and different size [BAH 99a]. Different pieces of information about the MegaStore application domain are defined and stored as a set of inter-linked objects of different kinds, grouped by their domain of interests, e.g. artists, songs, CDs, consumers, stores, burning towers, etc. We have also taken into consideration the support for the following needs:

- To enrich the database, in order to support data of different types (text, html, images, audio, video, postscript, etc.) [BAH 99a].
- To capture the inter-relationship semantics among the objects, through the storage of a large set of relationships among different pieces of information.
- To build a distributed environment where data must be stored at different music centers, without replication or redundancy at different sites [BAH 99, BBO 99a].
- To extend the database in a way that the Graphical User Interface application requirements can be supported through the Web server.

The schema represented in Figure 4.1 shows the static view of the MegaStore database catalogue (also called Directory Services) in terms of classes and relationships among them. The Database catalogue defines the general music information as described within the section. The real music data however, is stored at the secure parallel/distributed database server described in section 4.3.3. The name of a class is derived from the problem domain and must be as unambiguous as possible. The attributes define the characteristics of the class and capture the information that describes and identifies the class; every attribute has a type, which specifies what kind of data it represents. The relationship association between classes is drawn as a solid line and has a name and a multiplicity range [UML 98].

The part of the database schema design for the MegaStore system that is presented in Figure 4.1 describes the detailed structure of its Directory Service. For instance, a customer can order some albums, where each album consists of a set of Songs, and it is possible that one or more artists sing every Song. A song may also have a link to its music composer, music performer and/or some instruments. Such a specification may help in satisfying users through many points of views. For instance, a user may be more interested in his/her search, in the music performer, the music composer, or the used instruments, rather than being interested in a search based on the artist name or the song title.

Following is a brief definition of the variety of information within the schema defined for the MegaStore database catalogue:

- The class Song represents the main entity in the MegaStore system. Within this class, the information about each song is defined. The richness of MegaStore system strongly depends on the availability of such information in the database. Since the
Internet-Shop is a dynamic Web interface, for which Web pages are created on the fly depending on the user request, the system automatically checks the database and provides the user with the most complete information that it finds in the database. The class Song has three links to the classes Album, Artist, and Instrument via the defined relationships “Song Of Album”, “Sung By”, and “Uses”.

- The Album entity represents the class for CDs, tapes, and other means of music titles collection. Mainly, an album has some characteristics, consists of a set of songs, and one or several artists sing the songs in the album. The class Album links to other classes such as Artist and Song, through the specified relationships “Album Artists” and “Album Songs”.

- The class Artist is the entity that holds all the information about each artist. Under the normal consideration some attributes such as the artist name, artist photo, and a short biography, are enough for the artist description.

- The class Customer keeps the necessary information about the customers. Each time a user makes an order, the system automatically checks the user’s identity based on the information available in the database, and decides whether to directly access the information about him/her from the database, if it exists, or requests it from the user, if it is not.

![Figure 4.1: Base Schema Definition for the MegaStore System](image_url)

### 4.2.2 ODL Schema definition

The ODL schema definition of the MegaStore Database presented below is a conversion of the music library database as presented in Table 4.1 which, corresponds to the Unified Modeling Language (UML) into an Object Definition Language (ODL). The ODL schema presents the big advantage that it can be automatically loaded into the Matisse database [Mt 01] or any other ODMG compliant database [CBB+00]. In addition to the standard ODL definition, Matisse ODL supports the following extensions:
Multimedia types (e.g. audio, video, and image) are implemented in Matisse DBMS as a list of 8/16 bit Unsigned Short.

The `entry_point.dictionary` function serves as an entry point for the object to which the attribute belongs. Entry points are associated with the attributes by using the keyword `entry_point.of` and also serve as an efficient index.

The cardinality is specified in between square brackets after the name of the relationship. The first digit in between the square brackets specifies the minimum number of successors, and the second digit specifies the maximum number (-1 means no limit). The default cardinality for a relationship association is [0, -1].

The empty initialization for some attributes declares a null default value for these attributes.

```java
interface SONG : persistent {
    attribute String SongTitle;
    entry_point_dictionary epSongTitle
    entry_point_of SongTitle;
    attribute AudioSongStream;
    attribute String SongLabel = "";
    attribute String SongGenre;
    attribute String SongLyrics = "";
    attribute AudioSongSample;
    attribute String SongType;
    attribute Date ReleaseDate;
    attribute Float SongPrice;
    attribute String StatusCode;
    attribute String SongLanguage = "En";
    attribute Short SongDuration;
    attribute Video SongVideo;
    relationship List<Album> SongOfAlbum
    inverse Album::SongOfAlbum;
    relationship List<ARTIST> SungBy
    inverse ARTIST::SungBy;
}

interface Album : persistent {
    attribute String AlbumBarCode;
    attribute String AlbumTitle;
    entry_point_dictionary eAlbumTitle
    entry_point_of AlbumTitle;
    attribute Image AlbumCover;
    attribute Float AlbumPrice;
    attribute String AlbumPublisher;
    attribute Date AlbumPubYear;
    relationship List<SONG> AlbumSongs[1,-1]
    inverse SONG::SongOfAlbum;
    relationship List<ARTIST> A_Artists[1,-1]
    inverse ARTIST::ArtistAlbums;
    relationship List<ORDER> AlbumOrder
    inverse ORDER::OrderAlbums;
}

interface ARTIST : persistent {
    attribute Image ArtistImage;
    attribute String ArtistName;
    attribute String ArtistBio = "";
    entry_point_dictionary epArtistName
    entry_point_of ArtistName;
    relationship List<CDAlbum> ArtistAlbums
    inverse Album::AlbumArtists;
    relationship List<SONG> ArtistSongs
    inverse SONG::SungBy;
}

interface ORDER : persistent {
    attribute Date OrderDate;
    attribute Float OrderPrice = 0.0;
    relationship List<Album> OrderAlbums[1,-1]
    inverse Album::OrderAlbum;
    relationship List<Customer> OrderedBy
    inverse Customer::OrderedBy;
}

interface Customer : persistent {
    attribute String CustomerName;
    attribute Short CustomerFaithful;
    attribute String CustomerCompany;
    attribute String CustomerAddress;
    attribute String CustomerCountry;
    attribute String CustomerEmail;
    relationship List<ORDER> CustomerOrders
    inverse ORDER::OrderedBy;
}

interface User : persistent {
    attribute String UserLogin;
    attribute String UserPassword = "";
    attribute String UserDescription = "";
    attribute short UserLevel = 1;
}
```

Table 4.1: ODL Schema for the MegaStore Database

Furthermore, the Matisse ODL supports other object-relational extensions that are not illustrated in the example in Table 4.1. Such extension includes the definitions of indices and methods for the attributes as well as the specification of check functions and triggers for both attributes and relationships.
4.3 The MegaStore System Architecture

This section focuses on the design of the server architecture and its extension to support the Virtual MegaStore system. The choice of the server architecture extensions are due to the specific needs of this application for supporting its information management and the data transfer requirements.

As depicted in Figure 4.2, the MegaStore system consists of two components for data storage: the database catalogue at the Directory Services, and the parallel/distributed database server at the back-end system. All the music information including the short clips of the converted streaming audio/video that do not require high security protection will be placed at, and accessible through, the database catalogue, to be made available to the Internet users. However, the real raw music data that serves for CDs burning is securely kept at different distributed music centers linked with each other via a secure predefined network connection, so that only authorized users can access and manipulate the raw music data. The two sections below respectively describe in more details (1) the Internet-Shop interface to which ordinary users can connect in order to access the general music information and place their orders, and (2) the Shop-in-a-Shop interface to which only authorized users from the music stores can be connected.

![Diagram](https://example.com/diagram.png)

Figure 4.2: MegaStore Server Architecture Description

There will be three kinds of connections that need to be established between the components of the MegaStore system, namely:

1. High bandwidth connection, which is required for the case of transferring a considerable amount of data. This is usually the case, for transferring raw music data between geographically distributed music centers where the studies show that a minimum of 1 Gigabyte per second Internet connection is required [BAH 00].

2. Medium bandwidth with a medium latency connection, which is required for transfer of medium size data between the MegaStore system components. This is usually the
case, for updating the directory services when new music albums are produced (e.g. the connection from the Shop-in-a-Shop interface to the directory services).

3. Low latency connection, which is required to support the huge number of high end-users who access the MegaStore system and require the transfer of small amount of data (e.g. the connection between the Internet-shop interface and the Directory Services). The analysis of this application domain, shows that a huge number of users must be supported [BAH 99], the data to be transferred between the Internet user and MegaStore server however, will be in the range of medium to small size.

The design of the innovative architecture and technology for the MegaStore distributed environment although applied to the music industry application, is general enough to be applied to other similar complex application environments. Such complexities are found in many other e-commerce applications. Fore example, for the complex applications where the research and market findings need to be combined with the commercial advertising data, as it opens the collaboration among different enterprises based on common interest in system architecture platform and database content and technology.

4.3.1 The Internet-Shop Interface

The Internet-Shop attempts to give the user the same feeling as when he or she is visiting a real music store. The music store, as we know it today, is a place where items can be touched, where intuition plays a bigger role than knowledge, and where music albums are usually bought in an unplanned fashion. This is how most people behave during shopping and this is also what makes the shopping itself fun. The real shop is not limited to a search engine, which finds products in a faultless but also emotionless manner. Even for the more expensive products in general, the final decision is often not objective, but is based on the emotions caused by the color, form, brand, and price. Therefore, the user interface of the Internet Shop needs to invoke the same kind of emotion as in a real shop.

This means that in an ideal case, a consumer from home wishes to request tracks/titles that he/she wants to be included in one CD of his/her own compilation, and if he decides to buy this tailored CD, he wishes to pay, in a secure way, by means of electronic payment. If the payment is accepted, the system must automatically allocate this customer's order to the closest music shop, in relation to the location of the customer. At that shop the titles are eventually burned and delivered to the user.

4.3.2 The Shop-in-a-Shop Interface

The Shop-in-a-Shop interface adds a new dimension to music shopping. Normal music shops only have a limited stock. If a customer is looking for a specific piece of music that is not in the stock, he or she cannot be served. To avoid this problem, the Shop-in-a-Shop system places a customer terminal either in the music shops or other shops, e.g. photo developing shops, or even supermarkets, that supports the browsing and selection of music that is stored in a database server. If the customer has decided to purchase a piece of music, a CD is produced by downloading the raw music data from the database server. Also, additional information like booklet, the CD label, and the CD cover is downloaded and produced. Naturally, the music store must have a very high bandwidth Internet connection to be able to retrieve the large raw CD data in a few minutes time. The advanced internet technology although perhaps lucking behind the support for vastly increasing number of customers for
4.3. The MegaStore System Architecture

e-commerce, due to the costs and required efficiency, dedicated fast connections are expected to be available in the near future to properly support the music industry application. Also, the production equipment like CD burners and high quality printers are expected to be sufficiently fast in the near future.

4.3.3 Server Architecture Extension

This section addresses the server architecture extension to support both the information management and data transfer requirements for the MegaStore system. The main requirements to take into consideration includes:

1. Design and implement the extensions needed for the existing parallel server system [PH 98], in order to support all identified Virtual MegaStore database functionality. These extensions support:
   (a) The functionality needed by the HTTP daemons (Web server) front-end, in terms of support for the Web user interface, including the streaming of audio and video data.
   (b) Easy database administration.

2. Develop and implement a mechanism that supports the entry of music and associating data into the database system [BAH 99b].

4.3.3.1 Distributed Parallel Server Extension:

To provide the MegaStore web server with efficient access to the raw music data, a parallel/distributed database framework is designed and developed [PH 98]. With this implementation, the nodes (music stores) of the distributed MegaStore server are inter-connected, making it possible for specific users to connect to any node in the distributed server and to request an object, without the need to know where that object actually resides.

Due to the music data specification and copyrights that do not allow data replication or redundancy, this data must be securely kept at the site where it belongs. The distributed database supports the following required functionalities:

- Provides a way for managing huge amount of data
- Data is securely kept at geographically distributed music centers
- Data is stored only at the point(s) where it belongs
- Data is visible from any node (music center) within the cooperation community
- Data is efficiently transferred between the nodes in short response time

4.3.3.2 Data Storage and Manipulation:

The MegaStore data manipulation concerns two components: the database catalogue at the Directory Services and the parallel-distributed database server at the back-end system. All the music information including the short clips of the converted streaming audio/video that do not require high security protection will be placed at and accessible through the Directory Services, to be made available to the Internet users. However, the real raw music data that serves for CDs burning is securely kept at different distributed music centers linked
to each other via a secure network connection, so that only authorized users can access and manipulate the raw music data.

The music data loading for MegaStore is a two-fold process. On one hand it stores the raw music data at the secure distributed server, and on the other hand it updates the Directory Services with the general information concerning newly acquired albums and titles. The storage of the music data is provided locally at each site (music centers) by the music producer framework that can be in some cases integrated with the Shop-in-a-Shop interface. At this level, in order to keep the system up-to-date and more consistent, not only the music data entry mechanisms are provided, but also the music data conversion and data formatting are considered [BAH 99b].

4.4 Music Audio and Video content

This section addresses some issues related to the music data conversion and briefly describes the variety of different music formats supported by the MegaStore system, as well as it provides information concerning the music encoders.

Music Audio and Video clips consist of previously captured digital audio or video files, which can also be recorded from many types of media device. Currently, the MegaStore System supports most of the existing audio and video formats including Real Audio, MPEG, CD Tracks, Waveform, QuickTime, etc.

In addition, the MegaStore system is open to support other emerging standard formats, such as the Secure Digital Music Initiative (SDMI)\(^4\). However, most efforts investigated on music data conversion for the MegaStore system are mainly focussed on the Real Audio [RA Inc] and the MPEG (MP3) formats, due to the various advantages of these two technologies over the others. The RealAudio has an advantage of producing both audio streaming and video clips, and the generated files are of smaller size. The MP3 however presents the advantage that it produces near CD-quality music and it is widely used over the world.

4.4.1 Bandwidth and Encoding Algorithm

Bandwidth, also known as bitrate, is the amount of data that can be sent through an Internet or network connection during a set period of time. Bandwidth is measured in kilobits per second (Kbps). Standard modems are commonly referred to by the bitrate they are able to receive, for example, 14.4, 28.8, and 56 Kbps. In addition to the standard bandwidths, music clips can be recorded for bitrates up to 100 Kbps, 200 Kbps, or more. These higher bandwidths, however, are generally more typical of corporate LANs or entertainment-based Web sites.

When audio files need to be processed or digitized, an encoder and an encoding algorithm must be selected. Most Encoders can encode using different algorithms. Each encoding algorithm is optimized for a particular type of audio and connection bandwidth. Such a dynamic selection of audio/video clips allows the system to provide the Internet-user with the best quality connection his/her system can handle, without the user having to explicitly choose from separate clips recorded for different speeds. In the MegaStore system, for the digitized music clips we mainly use RealAudio and MP3 formats. RealAudio 56K ISDN. Music - Mono and Stereo template, best suits the Internet-Shop needs since we expect Internet users via ISDN or similar connection. While. MP3 200kps - CD quality music. is

\(^4\)http://www.sdmi.org/
required for the Shop-in-Shop burning system. This process is transparent to users, and the MegaStore system is configured to automatically serve the appropriate streaming file. As such, we can reach the widest possible audience, while still providing the best listening experience to users with a high bandwidth connection.

4.4.1.1 MP3 Encoders:

The MP3 audio format has become the standard for this main reason that it produces near CD-quality music and it is widely used over the world. MPEG Layer 3 files can fit up to a minute of CD-quality (44.1 KHz, 16-bit stereo) audio in a single megabyte. In comparison to an audio CD that contain a maximum of 74 minutes of music, filling a 650MB CD-R disc with MP3 files would result in more than ten hours of music.

The default bandwidth encoding in the programs is 320kbps, however a 128 kbps is recommended because it provides near CD quality music and smaller outputs. Any bandwidth higher that 320 kbps causes overkill, and will make the resulted files much larger which also implies longer download time and increases transfer costs.

MusicMatch Jukebox\(^5\) allows creating MP3s and WMAs from cassette or microphone with line-in recording. Ultimate Encoder\(^6\) is a high quality MP3 Audio encoder & decoder that supports MPEG Layer 1, 2 and 3. The XingMP3 Encoder\(^7\) processes files up to 8 times faster than other encoders. Faster encoding means you get to spend your time listening to your music, not waiting for it to encode.

4.4.1.2 RA Encoder:

The RealAudio Producer\(^8\) can encode using several different templates. Each encoding template is optimized for a particular type of audio and connection bandwidth. The RealAudio has an advantage over the MP3 in the sense that it can produce both audio streaming and video clips. and the generated files are of smaller size.

In the MegaStore system, for the digitized music clips we used RealAudio 56K ISDN, Music – Mono and Stereo. These templates best suit our needs since we expect Internet users via ISDN and similar connections. Using these algorithms to encode songs of 5 minutes duration for instance, produce digital audio clips of 1.37 MB per Song (4.9 KB * 300 s = 1.37 MB) which, may result to up to 40 hours of music per CD.

4.4.2 Data Volume Estimation

This section gives some estimation about the data volume that needs to be handled within the Internet-Shop and the Shop-in-a-Shop interfaces.

For the Internet-Shop interface, the average disk space requested per CD will be around 8 MB (a CD album contains about 15 titles: 550 KB * 15 = 8.25 MB). Thus, a prototype system of 5000 albums requires about 40 Gigabytes of disk space.

For the Shop-in-a-Shop interface we should add to the total disk space needed for the Internet-Shop an average of 65 MB per CD representing the complete raw data (in MP3 format), which is around 350 Gigabytes for a system dealing with 5000 albums.

\(^5\)http://www.musicmatch.com/jukebox/
\(^7\)http://www.xingtech.com/mp3/encoder/index.html
\(^8\)http://www.realnetworks.com/products/producer/
The MegaStore system is designed in a flexible way that supports different implementation strategies. For instance, if the system needs to be extended to support more albums and titles, one strategy that preserves the system performance is to store the audio/video clips together with the raw music data at the distributed/parallel database server and not at the Directory Services. This approach not only extends the system in terms of supporting a considerable amount of music data, but it also preserves the performance of the system including short response time and data security, by keeping the directory services as efficient as possible. The Directory Services play a major role in defining the general MegaStore information and specifying where the related raw data for each song or album is located within the distributed system.

4.5 Music Data Manipulation

Due to the huge number of instances to be loaded into the database and the relationships to be maintained among those objects, the data loading process into the database is an error-prone task to be done manually. Some automatic mechanisms and tools have been developed in order to ease the data loading process. The developed tools expect the data to be available in specific format in order to be automatically scanned to a standard format and loaded into the database.

4.5.1 Objects Loading Strategies

For the data translation process among heterogeneous data sources, a number of approaches are described in chapter 2 (section 2.2.2.2). In this section, we describe two similar approaches, which serve the general data storage mechanism in the MegaStore projects.

1. The first approach builds specific interface to directly store the data from its available format into the database. This approach requires the development of two-side dependant interface for each input format. If we consider N different input formats to be loaded into M databases, the number of interfaces to be developed will be \(N \times M\) interfaces (Figure 4.3-a). Thus, the number of interfaces to be developed increase as more input/output formats are considered. The mapping of ten different data inputs into five different databases, for instance, requires the development of fifty specific interfaces (10 * 5).

2. The second approach uses an intermediate step first by storing the data into an intermediate standard format (e.g. OIF, XML), and second by loading this data into the database system. The advantage of using intermediate standard format is that, it can be loaded into any DBMS that is compliant to standards. As depicted in Figure 4.3-b, the number of interfaces to be developed for N different input formats will be reduced to N+1 (i.e. one-dependant side interface for each input format plus one standard interface (the standard interface is usually supported by the DBMS itself)). In this case, the mapping of ten different data inputs requires the development of ten specific interfaces regardless the number of target database systems (outputs).

The intermediate format through standard only requires the development of a one-sided-dependent interface for each input that needs to be stored into the database. It has the advantage over the first approach of being ready to communicate with any other database system or application program that are compliant to standards. Thus, this approach, based
on intermediate standard format, is certainly preferred for most cases. However, in order to cover all possible cases, we also foresee the necessity for supporting the first approach in some cases that are simple and not expected to change.

![Diagram](image)

**Figure 4.3: Data Storage Mechanisms**

For the music data storage into the database, depending on the input data set structure, several algorithms are available and can be used to fill the database with the existing data that is available in different format.

Regarding the first approach we have developed a specific interface to directly read the data from a Dbase file "*.DBF"\(^9\) and store it into the Matisse Database [Mt 01]. The DBF file was provided by a partner from the music domain and it contains more than 50,000 record about song titles and artist names. However, the provided data does not include any audio/video streaming or images.

For the second approach, we have developed three algorithms for data loading. Each of those algorithms generates an OIF standard format of different levels of complexity for which a generic OIF loader is used to fill the database. The three data sets entry algorithms supported by the system are described within this section, namely:

**Format A:** expects a set of subdirectories, where each subdirectory is named same as artist name and contains a list of audio/video files (digitized songs) for that artist stored locally. The format of each file\(^{10}\) consists of the complete song name with the proper extension that reflects the audio/video type, and the file size that can be used in estimating the duration time for each song. From the description of the input data set, as presented in Figure 4.4, the developed tools allow the creation of the Artist objects and for each artist, the complete set of songs is created with the proper links between the two classes *Artist* and *Song* via the relationships *ArtistSongs* and *SungBy.*

---

\(^9\) A dBASE file, a format originated by Ashton-Tate, but understood by Act!, Clipper, FoxPro, Arago, Wordtech, xBase, and similar database or database-related products.

\(^{10}\) In order to preserve music labels and copyrights, the real name and type of the titles are not shown in the example.
Chapter 4. MegaStore: Advanced Web Databases for Music Industry

Eric Clapton

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Song 1:ae</td>
<td>1.069KB</td>
</tr>
<tr>
<td>Song 2:ae</td>
<td>1.102KB</td>
</tr>
<tr>
<td>Song 3:ae</td>
<td>1.572KB</td>
</tr>
<tr>
<td>Song 4:ae</td>
<td>1.434KB</td>
</tr>
<tr>
<td>Song 5:ae</td>
<td>1.636KB</td>
</tr>
<tr>
<td>Song 6:ae</td>
<td>1.028KB</td>
</tr>
<tr>
<td>Song 7:ae</td>
<td>1.086KB</td>
</tr>
<tr>
<td>Song 8:ae</td>
<td>2.332KB</td>
</tr>
<tr>
<td>Song 9:ae</td>
<td>1.132KB</td>
</tr>
<tr>
<td>Song 10:ae</td>
<td>1.901KB</td>
</tr>
</tbody>
</table>

(a) Artists

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Song 1:ae</td>
<td>1.069KB</td>
</tr>
<tr>
<td>Song 2:ae</td>
<td>1.102KB</td>
</tr>
<tr>
<td>Song 3:ae</td>
<td>1.572KB</td>
</tr>
<tr>
<td>Song 4:ae</td>
<td>1.434KB</td>
</tr>
<tr>
<td>Song 5:ae</td>
<td>1.636KB</td>
</tr>
<tr>
<td>Song 6:ae</td>
<td>1.028KB</td>
</tr>
<tr>
<td>Song 7:ae</td>
<td>1.086KB</td>
</tr>
<tr>
<td>Song 8:ae</td>
<td>2.332KB</td>
</tr>
<tr>
<td>Song 9:ae</td>
<td>1.132KB</td>
</tr>
<tr>
<td>Song 10:ae</td>
<td>1.901KB</td>
</tr>
</tbody>
</table>

(b) Artist Songs

Figure 4.4: Music Input - Format A

Format B: As depicted in Figure 4.5, Format B expects a random list of audio/video files stored locally, the format of each file consists of the artist name between parentheses followed by the complete song name and its extension. The system, for instance, can extract from the “(Dire Straits) Money for Nothing.ra” entry, the name of the artist “Dire Straits”, the title of the song “Money for Nothing”, the type of the audio format “Real Audio”, and the audio streaming itself. This information will be stored in the database together with other generated set of relationships such as “Artist Songs” from artist “Dire Straits” to song “Money for Nothing”, and “Sung By” from song “Money for Nothing” to artist “Dire Straits”. If the file format does not contain information about the Artist, only objects of the class Song will be created with the necessary information and stored into the database.

English Songs

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Artist Name) Song Title</td>
<td>5.130KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>4.353KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>2.446KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>1.639KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>1.586KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>1.335KB</td>
</tr>
<tr>
<td>(Dire Straits) Money for Nothing.ra</td>
<td>787KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>1.106KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>1.056KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>1.023KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>940KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>884KB</td>
</tr>
<tr>
<td>(Artist Name) Song Title</td>
<td>854KB</td>
</tr>
</tbody>
</table>

Figure 4.5: Music Input - Format B

Format C: can be considered as an extension of formats A and B. In addition to the description given within format B, the Directory structure will be divided into several subdirectories and each subdirectory is named same as the Album name and contains the list of the Album titles (see Figure 4.6). Same as it happened in real, an album consists of a set of songs and one or several artists are singing each song. This format
is the most suitable, since it allows the creation of three set of class objects (Album, Artist, Song) with the necessary relationships among them. In addition to the specification supported in Format A and format B, the system will automatically extract the Album name and creates the necessary links to the classes Artist (via AlbumArtists and ArtistOfAlbum) and Song (via AlbumSongs and SongOfAlbum).

4.5.2 Extensions

From the three formats presented above, it is clear that some information can be easily extracted from the music data set such as the artist name, song title, streaming size, etc. However, some other information can also be automatically derived from the music data set as presented in the three formats above.

- The type of music can be extracted from the extension of the streaming file (e.g. *.rm, *.ra, and *.ram extensions are Real Audio specification; *.mp3, and *.mp2 are MPEG specification, etc.). The music type helps in invoking the proper music player for the selected audio/video streaming. During a web session for instance, the proper plug-in will be automatically activated by the system based on the type of the music stream.

- In addition, since the music encoder for audio/video tracks is known or can be extracted. Then, the song duration can be calculated and provided to the database system. If we consider a Real Audio encoder using a 56K Dial-Up, Music – Stereo connection that provides a 32 kps streaming audio. We can easily estimate the song duration of Money for Nothing.ra to 3 min and 44 seconds, since the file size is 787 KB and the used algorithm require 3.5 kilo byte disk space per second.

An Object Interchange Format (OIF) example is presented in Table 4.2, in this example information about Songs, Artists, Albums are specified and the different links between these.
entities are defined. This example depicts a part of a simple standard exchange format generated by the algorithm described in format C.

The reader of this document may notice that, within the example presented above, there are additional information such as the ArtistImage and the CDCover that are not present in any of the three formats presented above. For that we consider a set of images that are already loaded into the database, the images are supposed to be titled same as artist name and album title. During the process of music data loading into the database, the system automatically checks if there exists a corresponding image for the artist or the album, and if so a link to that image will be created and maintained for future use.

| Song{SongTitle "Beatiful Girl", SongStream "Beatiful Girl.ra", SongDuration "2:29"} |
|---------------------------------|-------------------------------------|
| SungBy{Artist{ArtistName "CharlieRich", ArtistImage "CharlieRich.jpg"}} |
| SongOfAlbum{Album{AlbumTitle "All Time Classics", AlbumCover "All Time Classics.jpg"}} |
| Song{SongTitle "How 'bout Us", SongStream "How 'bout Us.ra", SongDuration "4:30"} |
| SungBy{Artist{ArtistName "Champaign", ArtistImage "Champaign.jpg"}} |
| SongOfAlbum{Album{AlbumTitle "All Time Classics", AlbumCover "All Time Classics.jpg"}} |
| Song{SongTitle "When I see your Smile", SongStream "Smile.ra", SongDuration "4:00"} |
| SungBy{Artist{ArtistName "Beat English", ArtistImage "Beat English.jpg"}} |
| SongOfAlbum{Album{AlbumTitle "All Time Classics", AlbumCover "All Time Classics.jpg"}} |

Table 4.2: An OIF Example

### 4.5.3 Database Administration

Some database administration facilities for the management and maintenance of the MegaStore back-end system are provided. Among other features, the database administrator interface (DBA), implemented in C++ and in Windows NT environment:

- Allows the creation of the database schema based on the data model defined for the MegaStore system.
- Eases the generation of ready for input data files via the scanning of storage disks and reformatting the data to fit the standard Object Interchange format (OIF).
- Supports the automatic loading of the music data from standard (non standard) input format and the creation of the necessary links between inter-related pieces of information, and
- Provides a mean for dynamic Web pages generation based on specific user requests.

In order to preserve the data security and confidentiality, users of the system need at first to connect to a running database server, where they must provide the system by the database name, the host of the machine running the database, and eventually the user name and password.

Figure 4.7 shows the main menu of the DBA interface for the music data objects management. The modules presented on that interface are the OIF files generator for each of the mechanisms described in section 4.5.1, and the OIF objects loader where the user do not need to do more than selecting an existing OIF file and validates it to be loaded into the database.
4.6 MegaStore Interfaces - Advanced Features

Some major benefits for a Web application to deploy a database management system are: the dynamism, flexibility, cataloguing, and searching facilities. In the domain of e-commerce, customers want to be able to find and view different products, make comparison between those products, and select the products that best suit their needs in an efficient manner. Another challenging feature in such applications is the fact that customers may have different preferences based on their needs, their cultures, or depending on the occasion. This section briefly describes the Internet-Shop user interface for music titles and CD album provisions, more details about this interface can found in [BAH 99b]. The Internet-Shop interface is a Web server for music titles and CD albums, it allows a user from home (or at work) to search for music in an efficient way, listen to partial/complete music tracks, and order the music he likes.

The adopted MegaStore web server architecture, described in section 4.3, provides the Internet-Shop interface with an efficient access to the raw music data where, the distributed/parallel database framework is adapted and extended to handle the huge amount of raw music data required for burning Compact Discs [BAH 99b].

Figure 4.8 illustrates an activity diagram for the MegaStore Web interface using UML notation [UML 98]. The activity diagram for MegaStore has a starting point and several end points. A starting point (also called initial state) is shown as a solid filled circle, and an end point (also called final state) is shown as a circle surrounding a smaller solid circle. A state in a diagram is shown as a rectangle with rounded corners. Between the states are states transitions, shown as a line with an arrow from one state to another. In this diagram, for simplicity and clarity reasons only the “Show Help” state shows the possibility to end the web session or move to another entry point (state in the diagram). However, the system implementation strategy allows the possibility of ending a session or moving to another point at/from any state presented in the diagram.

This diagram helps the reader of this document to get a global overview concerning the MegaStore Web Interface implementation and shows how the different modules inside this
interface are connected to each other. Namely, level 1 corresponds to the modules that are accessible from the “Main Menu”, level 2 matches the interfaces invoked from the above level and displayed on the “First browsing Area”, and level 3 presents the modules showed on the “Second Browsing Area”.

![Activity Diagram for the Internet-Shop Interface](image)

Figure 4.8: An Activity Diagram for the Internet-Shop Interface

Figure 4.9 presents a screen shot of the MegaStore Internet user Interface. As depicted on this figure the MegaStore system interface consists of three main areas of information presentation:

1. The Menu (top left) presents the starting entry from which the user can explore the MegaStore system for music titles, CD albums, artists, and gets help when requested.

2. The main browsing area (bottom left): at the starting of the system (the fist time a user connect to the MegaStore system), this area will be the target for top music hits, best selling and advertisements for which we want to notify the user and bring his/her attention. Later on, this will be the area where results of the user requests are presented in an organized way. If the user for instance request all the available albums (from the menu by clicking the album’s button and then submit the Search Album button) the result of his request will be presented on this main browsing area. In this case, the result consists of a sequence of web pages representing information about each album.

3. The Second Browsing Area (to the right): At first, on this area a short description of the MegaStore system will be provided to the user. During the user exploration of the system, this will be the browsing area where goes the additional results to the user request. For instance, if the user requests a CD album from the main menu (1) by simply specifying the album name, the response to his request will be organized and presented in the main browsing area (2). From this result the user can click the album cover and gets more information about the album. Mainly, he can check the titles within the Album and order it, he can also listen to the music of each song, etc.
The MegaStore Web interface allows users to navigate through the system and find the information of their interest in a very efficient way. This section presents a brief description of some innovative features supported by the MegaStore Internet-Shop interface, such features include: the system security, ordering mechanisms, dynamic browsing, embedded plugging, etc.

### 4.6.1 Dynamic Browsing

Using the MegaStore Internet interface, user queries are immediately sent to the database and the result of each query is merged on the fly with the corresponding predefined html formatting, in order to be viewed by the user who is clicking a way the MegaStore Web Interface. Thus, web pages are created on the fly based on the user request, and information availability in the database.

Figure 4.10 presents the result of a query submitted by a user requesting all the available information for the Album “She is the Best 1”. The system for instance, provides the more relevant information and makes the embedded links for audio/video clips based on the availability/non-availability of this information. Moreover, the system is built in such a way that the user will not feel data incompleteness in the interface. In this case for instance, in addition to Album name, Album cover Image, and the list titles within the album, the MegaStore system provides the following functionalities:

- The image button at the bottom of the album image, titled ‘Add Album to your Cart’, allows the user to add the current CD album to his/her standard order. In this case the user makes standard order by selecting existing albums,

- the optional speaker icon (●) after each song, indicates the audio stream availability for the corresponding title. This option gives the Internet user the possibility to listen to single partial/complete music tracks from the album. In addition, the speaker icon on the top-right, after ‘Album Songs’, gives the possibility to listen to all album’s songs by a single mousse click on that icon,
- The optional camera icon ( ), after each song, indicates the video clip availability for the corresponding title,
- The check-in box marks the tracks licensed for customs compilation, and
- The button at the bottom, titled ‘Add to Cart’, adds the checked titles to the user customized order. In this case the user can make his own compilation of titles to be included within an Album (see section 4.6.2). Please notice that the user can always add new titles to his custom compilation from different screen sessions. A tailored order will not be taken into account until being validated by the user.

![Figure 4.10: Album Songs Interface](image)

### 4.6.2 Ordering System

The MegaStore Internet Shop do not only allows users to search for artists, titles, albums, and listen to audio/video clips, but it also allows a user from home or at work to make his own orders and place them into the system.

Figure 4.11 presents a state diagram for orders using the UML notation [UML 98].

![Figure 4.11: State Diagram for Orders](image)

The MegaStore ordering system has the capability to produce both standard Album orders and custom compilation orders.

- **Standard Orders:** A user can create an order based on existing albums. Each time he/she adds a new item to his/her CD shopping cart, a general overview about the current order status will be shown including: the list of ordered items, the price of
each item, and the total charge for the order including the handling and shipment costs. The user will always have the possibility to add/remove items to/from the order, and the order will not be taken into consideration unless it is validated (checked out). The order validation process includes gathering the ordered items, the customer information, and the payment procedure.

- **Customized Orders:** As depicted on Figure 4.12, the user customized Albums feature allows users to select the set of titles to be included in the album. While the user navigates through the Internet-Shop interface he/she can select the songs to be added to his/her tailored album compilation. The number of titles to be included in the compilation mainly depends on the total space/time available for the songs support (CD, tape, etc.). The total price of the customized album is gradually calculated based on the price of each song and other handling, shipment, and compilation costs.

<table>
<thead>
<tr>
<th>Titles in your Custom CD: 1</th>
<th>Price</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>€0.7</td>
<td>Remove</td>
</tr>
<tr>
<td>Where Does My Heart Beat Now</td>
<td>€0.6</td>
<td>Remove</td>
</tr>
<tr>
<td>La Vie En Rose</td>
<td>€0.8</td>
<td>Remove</td>
</tr>
<tr>
<td>Don't Worry</td>
<td>€0.7</td>
<td>Remove</td>
</tr>
<tr>
<td>Miss You Nights</td>
<td>€0.7</td>
<td>Remove</td>
</tr>
<tr>
<td>I Don't Want To Lose You</td>
<td>€1</td>
<td>Remove</td>
</tr>
<tr>
<td>Against All Odds</td>
<td>€0.6</td>
<td>Remove</td>
</tr>
<tr>
<td>Dance Into the Light</td>
<td>€0.9</td>
<td>Remove</td>
</tr>
<tr>
<td>True Colors</td>
<td>€1.1</td>
<td>Remove</td>
</tr>
<tr>
<td>CD+S+H</td>
<td>€6.50</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>€13.60</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.12: Custom Order

In order to preserve music labels and artist rights, only music tracks specifically licensed by music labels and artists for custom compilation can be included in tailored orders. Currently, most major music labels only provide their music for full Album sales.

### 4.6.3 System Security

The MegaStore System is a multi-level security system. Several kind of users are expected to use the system including (three roles defined for accessing the MegaStore system):

- The ordinary user (access level 1),
- The storekeeper (access level 2), and
- The system administrator (access level 3).

Within each session a user must identify himself (herself) to the system using a user name and a password, the system will automatically checks the user authentication against the information available in the database and allocate an accessibility level to the user. This level will be valid within the same session and the user can only see the parts of the information
(system) that corresponds to his/her visibility/access level. In addition, authorized users can always re-identify themselves with a different access level for the same session.

4.6.4 Current Implementation Status

For the implementation purpose of the MegaStore Internet-Shop interface, we combined different technologies to support the application requirements as described in section 4.2. Based on the detailed study of the MegaStore application functionality needs, the appropriate approach to apply and the technologies to use are identified [BAH 99]:

- The designed database is being implemented on top of the Arches computers at the University of Amsterdam. The Arches system is currently composed of 20 nodes containing each a dual Pentium II with 512 MB ram and 9 GB disk, and it supports several network communications.

- The Matisse database system [Mt 01] is used as the object-oriented distributed database system (ODBMS). Among other features Matisse supports transaction management, concurrency control, historical versioning, indexing mechanisms, high speed for data access, multi-media streaming, and standard programming interfaces using C/C++, Java, ODBC, etc.

- Different Internet infrastructures are deployed depending on each functionality of the MegaStore system (a high bandwidth communication between the music stores for huge amount of data transfer and a low latency communication to the Directory Services for high end-users access).

Some experiments are performed using an NT front-end machine to run a DBA interface and an Internet-Shop server, that is in turn connected to the Matisse database running on the Arches machines, using the ODBC driver. The database administrator (DBA) interface, implemented in C++ and Windows NT environment, provides some administration facilities including the automatic loading of the music data and the creation of the necessary links between inter-related pieces of information. And The Internet-Shop server is implemented using a combination of the most recent and relevant software technologies including JAVA Script and Visual Basic Script for tips programming, Active Database Objects (ADO) for database connection, and HTML for text formatting. The implementation of the server is made possible, using the Active Server Pages programming environment that allows the combination of all these different software technologies in one single environment.
4.7 Derived Applications

This section describes the implementation prototypes of two innovative interfaces related to the MegaStore system, namely, the LuisterPaal\textsuperscript{11} interface and the Music Sheet\textsuperscript{12} application. The LuisterPaal interface provides a simplified version of the MegaStore system, while the Music Sheet application provides an extended implementation of the MegaStore general concept. Figure 4.13 presents a conceptual model for e-MegaStore applications. This model shows the applicability of the comprehensive MegaStore framework to the design and implementation of similar systems in the area of manipulating multimedia large data sets. Thus, the two applications proof the general implementation approach for the MegaStore system, from which other applications that share the same characteristics can also benefit.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{conceptual_model.png}
\caption{Conceptual Model for an e-MegaStore Application}
\end{figure}

4.7.1 LuisterPaal Interface

The LuisterPaal interface [BAR+01] is a listening booth facility, which enable users visiting a music store to listen to music. In order to facilitate interactions with the system, the self-service LuisterPaal interface uses a scanner (barcode reader) and a touchscreen\textsuperscript{13}. The barcode reader facilitates the searching mechanism for the user, while the touchscreen is a finger selection technology that makes the interaction with the system more reliable. Music selection is simplified to the scanning of the barcode of a given album, therefore, the entry to the database is achieved via the unique barcodes of the albums.

From the implementation point of view the LuisterPaal does not differ much from the MegaStore concept. Its user-friendly interface is based on the use of (1) a database catalog holding the Album’s information (e.g. titles, duration, artists) and (2) a multimedia database server holding the real audio/video clips for each album.

\textsuperscript{11}The LuisterPaal interface is a joined project between the University of Amsterdam, Power Computing and Communication PCC - UvA, Free RecordShop NL, and Siemens Nixdorf.

\textsuperscript{12}The Music Sheet application is a joined project between the University of Amsterdam, Power Computing and Communication PCC - UvA, MondriCom b.v, and Attitude NL.

\textsuperscript{13}Touchscreens are specialized hardware products aimed at assisting interaction with computer learning devices. Touchmonitors emulate a mouse, giving the user "Free-Rein" in selecting appropriate items on the screen, by touching the requested features.
The LuisterPaal system is simple to use and very advanced regarding the functionalities it provides. It consists of a very friendly user interface, where a user can simply scan the barcode of a CD and follow a real interaction with the system. The barcode scanning is realized via a fixed barcode reader (scanner) attached to the touchscreen. The barcode reader and touchscreen are mandatory for simplicity reason; the system also runs on a PC with a normal keyboard and mouse.

Within the LuisterPaal system, the user in a music store can grab a case of a CD and scans its barcode to the system. At the music stores, instead of using the case of a CD to scan the barcode, the availability of simple catalogs for Albums including barcode’s specification provides a more convenient solution. When a user enters the barcode of an Album, the system connects to the database catalog, extracts the necessary information to be browsed to the user on the screen, and launches the audio/video streams to be played on-line. At any time during the listening process, the user can introduce the barcode of a new Album or freely interact within the audio/video clips of the album. Figure 4.14 shows three screens of the LuisterPaal interface. As depicted in the figures, the user can do the most audio/video operating functions (e.g. play, stop, pause, play next, play previous, play first, and play last). The LuisterPaal interface provides the following benefits and advantages:

- Very simple interface in which, a user has only to scan the barcode of an album, the rest is automatically done by the system.
- Flexible interface based on a large collection of music data stored in a database catalog,
- Support for video clips to be played on the screen,
- Audio/video data streaming are fetched on-line from the multimedia database server,
- Interactive system in which, a user can freely navigate across albums and titles, using specialized hardware devices (barcode reader and touchscreen),
- Unlike existing kiosks, in which a listing point is requested for each album, a single LuisterPaal interface is linked to the totality of albums within the database catalog.
- Most users in music stores are not aware on how to make complex queries based on keywords. Thus, The use of a barcode reader and a touchscreen facilitate the user interaction with the system,
4.7. Derived Applications

- The LuisterPaal approaches reduce the costs and efforts in making such facilities for ordinary users. The estimated costs and efforts are limited to the development of the database catalog and the set-up of the interface.
- The use of database standards and middleware solutions during the implementation of such a system eases its extension and makes it reusable.

4.7.2 Music Sheet Application

Music Sheet is a new emerging application in the domain of music notes and their performing issues. It consists of the design and development of a complete system, which model the complete life cycle of music production in one general concept. The development phases of the global concept rises from the lyrics writing phase, to the performing phase, to copyrights and publishing phase. At the same time it addresses the degree of difficulty, the used instrumentation, and Intellectual Property Rights (IPR) related to music production.

The Music Sheet server [BAR+01], briefly described in this section, provides a catalog that lets retailers form different music bookstores to search for music books, retrieve music notes, and place their orders into the system.

Among the highlighted development characteristics of the Music Sheet application are the extended data model and the flexible search interface. The complete data model allows the proper structuring of the application and links together related real world entities, while the flexible search interface provides a mean to navigate through a complex set of data inter-linked to each other via many relationship associations.

Figure 4.15 illustrates the data model for the Music Sheet application and shows the added entities to the data definition concepts, among which music notes, different instrumentation types, and instrumentation elements are supported. Within this model, music publications are well defined and classified into several categories of music books and folders. Each of these publications is composed of a set of titles; a title in turn is composed of several parts.

The richness of the Music Sheet data model has led to the development of a very flexible interface for the sheet music application. The search mechanism of the interface uses and combines several entities as entries to the database system. Search for publication titles and music notes is based on a combination of author name, title, instrument type, music genre, keyword, etc. A user for instance, can search for a piece of music notes of genre ‘classic’, composed by ‘Bach’, and played with both ‘organ’ and ‘string’ instruments.
Figure 4.15: Database Model for the Music Sheet Application
Figure 4.16 shows four screens from the Music Sheet interface, those interfaces illustrate few samples among many other interfaces\textsuperscript{14} dedicated to explore and navigate through the complete structure of the database model. Following are brief clarifications concerning the four samples presented in Figure 4.16:

1. \textit{Subscription} interface is mandatory when a user decides to order some music books or music titles.

2. \textit{Music search} interface allows a very flexible entry to the system, in which a user can specify and combine different keywords related to his/her search criterions (e.g. title, composer/author, number, genre, and instrument).

3. \textit{Composer search} interface explores in more detail information about authors. This interface is accessed via the main list of authors or via direct links from publications and publication titles.

4. \textit{Music notes} example provides a sample of the music notes available for the publication titles. Usually, those samples are free to download and to print.

\textsuperscript{14}Among them, we enumerate interfaces for publisher, suppliers, reviews, copyrights, detailed titles descriptions, and interfaces for music books ordering.
Advantages gained within the development of the sheet music application:

- The Music Sheet application benefits from the MegaStore concept and adopts its full ordering system.
- Publications and publication titles are well defined and more descriptive elements are introduced to their definition. Among these elements, we enumerate music note samples, degree of difficulty, reviewers' recommendations, and support for several types of keyword.
- A clear distinction is made between the different authors participating in the production of a given piece of music. The authors range from text writer or translator, to the music arranger, to the music composer.
- The system is open for specialists, reviewers, and system visitors to give their comments and recommendation for the different part of the music.
- The designed database model is comprehensive enough to express the intellectual property rights (IPR) such as text originality, copyrights, and publishing organizations.
- The Music Sheet interface provides a very flexible entry to the database based on the combination of keywords from different entities.
- The interface is more dedicated to music bookstore retailers, but it is also simplified and can be used by ordinary users.
- The system is extended with a component, which give users the possibility to describe in more details a desired item that it could not be found using the search interface. User requests will be treated by a specialist in the field and results (if any) are sent back to requesting user.

4.8 Conclusion and Discussion

This chapter addressed the innovative design methodology of an open architecture for the MegaStore application. The chapter first described in details the application requirement analysis and the database design, and then it addressed the general design and the server architecture for the Internet-Shop and the Shop-in-a-Shop interfaces. Further discussions are focussed on the issues of music conversion mechanisms, extensions to the distributed multimedia server, and the adaptation of the MegaStore framework to other e-commerce applications within the music industry. A distributed/parallel multimedia database server is adapted and extended to handle the huge amount of raw music data required for burning Compact Discs.

The development of MegaStore system and its extension with further sub-projects, such as the LuisterPaal interface, and the Music Sheet server; have proven the strength and generality of both the database design and the system architecture developed for these applications. The system analysis and the database design for the applications, developed for MegaStore, are achieved in collaboration with the experts from the music industry. Thus, the database schema description and names chosen for the schema components were taken directly from the music context, and also the object naming strategy in the e-MegaStore framework is mnemonic.
4.8.1 Major Characteristics and Benefits provided to MegaStore Application

As depicted in Figure 4.17, within the MegaStore framework, that is adjusted to support more e-commerce application, we have designed and developed an e-MegaStore architecture that seamlessly fits the current Music Store models of operations. In addition, in supporting their main goals, the e-MegaStore framework offers the following additional advantages:

- Value added information can be gathered from different major suppliers/retailers and gradually added to the database catalog. The database catalog can make references to multimedia data stored either at the multimedia server or detained by external suppliers. At the same time, it is possible that external catalogs link their information systems to the audio/video data stored within the multimedia database server, developed for the e-MegaStore applications.

- The use of Matisse object-oriented database system [Mt 01] made it possible to fulfill several requirements for MegaStore database development and its extensions, in specific to support multimedia data types, large objects handling, indexing, versioning, and high performance. The use of Matisse object-oriented database system together with a parallel/distributed database server for the development of the e-MegaStore applications provide the following advantages:
  - Allows flexible navigation through complex Web objects.
  - Supports scalability as necessary for multimedia large objects.
  - Provides high performance as required by multi-users applications.
  - Supports manipulation of new data types (e.g. for author biography, song lyrics, images/photos data, and audio/video streams).

- The security and data encryption issues for audio/video data streams are fully addressed and considered within different applications. The parallel/distributed database server for MegaStore provides three levels for security and encryption:
Short audio/video clips (data) are securely stored but not encrypted.
Medium audio/video tracks are encrypted and securely stored.
High quality audio/video streams are highly encrypted and securely stored.

4.8.2 Contribution of the MegaStore's Information Management Approach to GFI\textsubscript{2}S

The main idea behind the design and set-up of the necessary database structure and system architecture for the MegaStore application was to develop a comprehensive system that supports applications with two specific characteristics: (1) to facilitate the storage and manipulation of large data sets and (2) to provide a flexible information classification and clear separation between public and proprietary data. The experience gained from the approach followed in MegaStore contributes to the following three features in the design and development of the GFI\textsubscript{2}S:

- **System Reusability**, through the deployment of database standards and Internet middleware, for data definition and information access. Thus, making optimum use of the developed components, unifying the access to data, and reducing the development efforts.

- **System Efficiency**, through the development of parallel/distributed database servers and through the deployment of good strategies for the storage and management of large and complex data. These strategies have proven their strength and flexibility through different applications within current MegaStore-like models of operation.

- **User Assistance**, through the development of user friendly interfaces. In MegaStore, these interfaces are addressing the needs of ordinary end-users in E-commerce applications. While, in GFI\textsubscript{2}S, they are planned to also assist advanced users in creating federated schemas, specifying the access rights on the shared information, defining the schema derivation mappings, and so on.