VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

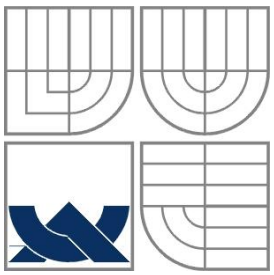
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DICOM MEDICAL IMAGE AND METADATA MANAGEMENT IN ORACLE DATABASE

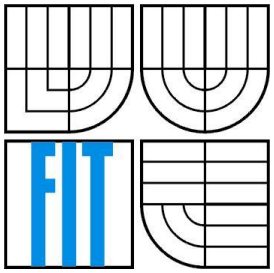
DIPLOMOVÁ PRÁCE
MASTERS´S THESIS

AUTOR PRÁCE SAMUEL GARCÍA BLANCO
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DICOM MEDICAL IMAGE AND METADATA MANAGEMENT IN ORACLE DATABASE

DIPLOMOVÁ PRÁCE
MASTERS´S THESIS

AUTOR PRÁCE
AUTHOR

SAMUEL GARCÍA BLANCO

VEDOUCÍ PRÁCE
SUPERVISOR

PETR CHMELAŘ

BRNO 2007

Abstract

This MSc. thesis deals with multimedia databases and is focused on medical images which follow the DICOM (Digital Imaging and Communications in Medicine) standard. This useful and widespread standard, along with storage, retrieval and processing of this kind of images are studied deeply in a general way and centered in Oracle databases. A sample application has been developed with the aim of knowing and testing the last versions of interMedia technology provided by Oracle, in which a developer can work with DICOM images. The application called "DICOM Manager", can help the doctors in hospitals keeping a register of visits with their related DICOM images and easily retrieving data from previous visits.

Keywords

image, multimedia, DICOM, medical, medicine, database, Oracle, interMedia, query, retrieval, insertion, update, standard, patient, visit, hospital

Abstrakt

Práce se zabývá správou medicínských snímků DICOM a přiřazených metadat. Demonstruje podporu pro tato multimediální data prostřednictvím Javy v databázi Oracle.

Klíčová slova

DICOM, Oracle, medicínské snímky.

Citation

GARCÍA BLANCO, Samuel. *DICOM Medical Image And Metadata Management In Oracle Database*. Master thesis. Brno University of Technology, Brno, 2007. 83p.

DICOM MEDICAL IMAGE AND METADATA MANAGEMENT IN ORACLE DATABASE

Declaration

I declare that I have solved this Master Thesis by myself.

I have mentioned all information resources used in the thesis.

……

Name

Date

Acknowledgement

I would like to thank my supervisor **Ing. Petr Chmelař**, for his great supervision, help and patience with this MSc. Thesis.

Thanks to everybody in Czech Republic who made this experience unforgettable for me, specially to my new friends which made me enjoy every minute of this 6 months, to **Ing. Pepe Gené**, **Ing. Javi Feria** and **Mgr. Monika Bandurova** the girl who gave me all her support and help. Thanks also to my friends from Spain which always let me know they didn't forget me.

Finally, I will always be grateful to **Ing. Jesús García** and **Ing. Ana Blanco** for their great work developed in the difficult field of the real life, during the last 25 years, with me and my little sister **Alejandra**.

Brno, May 2007

Content Index

Figures Index

Code Index

Chapter 1. Introduction

The healthcare industry is one of the most regulated, resource constrained and scrutinized industries in the world. It is under intense pressure to deliver the highest quality of service as efficiently as possible while addressing the needs of an increasingly demanding public.

For many decades, the healthcare system operated on paper and film. Registration and billing were paper based and diagnostic imaging was film based. Film was the norm in the diagnostic arena, and light boxes were the "browser" of choice. Information technology was first introduced in administrative systems, not in diagnostic or medical records systems. Over the past decade, medical records systems were introduced to manage patient records electronically, and Picture Archive and Communications Systems (PACS) were introduced to manage the new digital radiology modalities: X-Rays, Computed Tomography (CT), Magnetic Resonance (MR), Ultrasound, and so on. All of these first generation information technologies and digital techniques have been widely adopted by hospitals, medical centers, and managed care organizations, typically at the departmental level [2].

PACS solutions are becoming very large, increasing rapidly, for these different reasons:

1. Clinics and hospitals are purchasing more and more devices that can collect digital images.

2. More kinds of digital modalities are being offered by vendors and purchased by healthcare enterprises.

3. Newer machines produce images with higher resolution and thus, larger image data size.

4. Enterprise, regional, and even national archives of medical images and other health information are being designed and deployed.

For all of these factors, image storage needs are growing at unprecedented rates.

At the same time, healthcare enterprises are under tremendous pressure to extract additional efficiencies from radiologists and clinicians as well as to deliver better quality care. Thus, image archives must deliver images almost instantaneously anywhere in the enterprise, and the cost of developing and maintaining the archive software must remain as low as possible.

Providing researchers with quick access to anonymized medical records and images could produce significant breakthroughs in medicines, treatments, and efficiencies to improve future health care delivery and outcomes.

The delivery of healthcare in rural areas could be improved using regional networks and archives of medical images. Remote delivery and storage of medical

images means that the images produced at rural clinics could be stored, secured, and professionally managed in a central repository where experienced radiologists could analyze them with minimal delay. Higher network speeds and lower network costs could enable radiologists and local clinicians to collaborate on patient diagnoses even when large distances separate them.

This thesis is compound of several chapters that will describe the up to date technology of multimedia databases pointing at medical images which follows the DICOM standard. This Chapter 1 reviews the advantages of the computerization in the healthcare fields. Chapters from 2 to 5 covers the theoretical background in which this study is based on; databases, metadata, the DICOM standard and the way of working with all of them using Oracle interMedia. Chapter 6 is the technical part; it's about how to manage DICOM images using J2EE and Oracle interMedia. Chapter 7 covers all the process in the development of the sample application, which shows the functionalities of manipulation with this kind of medical images. In the Chapter 8 we can find the conclusions extracted from the previous and the future work. Chapter 9 contains some appends about Oracle interMedia, DICOM, the application manual and a guide about the CD content.

Chapter 2. Databases

A database is an organized collection of data. One possible definition is that a database is a collection of records stored in a computer in a systematic way, so that a computer program can consult it to answer questions. For better retrieval and sorting, each record is usually organized as a set of data elements (facts). The items retrieved in answer to queries become information that can be used to make decisions. The computer program used to manage and query a database is known as a database management system (DBMS). The properties and design of database systems are included in the study of computer science [3]. For a deeply study in databases see [6].

The central concept of a database is that of a collection of records, or pieces of knowledge. Typically, for a given database, there is a structural description of the type of facts held in that database: this description is known as a schema. The schema describes the objects that are represented in the database, and the relationships among them. There are a number of different ways of organizing a schema, that is, of modeling the database structure: these are known as database models (or data models). The model in most common use today is the relational model, which can be understand as a collection of tables each consisting of rows and columns with relations between them.

Strictly speaking, the term database refers to the collection of related records, and the software should be referred to as the database management system or DBMS. When the context is unambiguous, however, many database administrators and programmers use the term database to cover both meanings.

Database management systems are usually categorized according to the data model that they support: relational, object-relational, network, and so on. The data model will tend to determine the query languages that are available to access the database. A great deal of the internal engineering of a DBMS, however, is independent of the data model, and is concerned with managing factors such as performance, concurrency, integrity, and recovery from hardware failures. In these areas there are large differences between products.

2.1. Relational database model

“The relational model was introduced in an academic paper by E. F. Codd in 1970 as a way to make database management systems more independent of any particular application. It is a mathematical model defined in terms of predicate logic and set theory.” Extracted from [3].

The products that are generally referred to as relational databases in fact implement a model that is only an approximation to the mathematical model defined by Codd. The data structures in these products are tables, rather than relations that is how it must be if we follow Codd's model: the main differences being that tables can contain duplicate rows, and that the rows (and columns) can be treated as being ordered. SQL

language which is the primary interface to these products treats Codd relations as tables as well. There has been considerable controversy, mainly due to Codd himself, as to whether it is correct to describe SQL implementations as "relational": but the fact is that the world does so, and the following description uses the term in its popular sense.

A relational database contains multiple tables, each similar to the one in the "flat" database model. Relationships between tables are not defined explicitly; instead, *keys* are used to match up rows of data in different tables. A key is a collection of one or more columns in one table whose values match corresponding columns in other tables: for example, an *Employee* table may contain a column named *Location* which contains a value that matches the key of a *Location* table. Any column can be a key, or multiple columns can be grouped together into a single key.

It is not necessary to define all the keys in advance; a column can be used as a key even if it was not originally intended to be one if it was the correct properties like be unique for example. This property obeys to all values of this column to be different so it is possible to identify a row with the value to this column. This row is called *unique key.* If we use this column to refer to row it is called *primary key.*

A key that has an external, real-world meaning (such as a person's name, a book's ISBN, or a car's serial number), is sometimes called a "natural" key. If no natural key is suitable (think of the many people named *Brown*), an arbitrary key can be assigned (such as by giving employees ID numbers). In practice, most databases have both generated and natural keys, because generated keys can be used internally to create links between rows that cannot break, while natural keys can be used, less reliably, for searches and for integration with other databases. (For example, records in two independently developed databases could be matched up by social security number, except when the social security numbers are incorrect, missing, or have changed.)

2.2. Object database model

Since 80`s, the object-oriented paradigm has been applied to database technology, creating a new programming model known as object databases. These databases attempt to bring the database world and the application programming world closer together, in particular by ensuring that the database uses the same type system as the application program. This aims to avoid the overhead (sometimes referred to as the *impedance mismatch*) of converting information between its representation in the database (for example as rows in tables) and its representation in the application program (typically as objects). At the same time object databases attempt to introduce the key ideas of object programming, such as encapsulation and polymorphism, into the world of databases.

A variety of these ways have been tried for storing objects in a database. Some products have approached the problem from the application programming end, by making the objects manipulated by the program persistent. This also typically requires the addition of some kind of query language, since conventional programming languages do not have the ability to find objects based on their information content. Others have attacked the problem from the database end, by defining an object-oriented data model

for the database, and defining a database programming language that allows full programming capabilities as well as traditional query facilities.

Object databases suffered because of a lack of standardization: although standards were defined by ODMG, they were never implemented well enough to ensure interoperability between products. Nevertheless, object databases have been used successfully in many applications: usually specialized applications such as engineering databases or molecular biology databases rather than mainstream commercial data processing.

2.3. Object-Relational database model

An object-relational database (ORD) or object-relational database management system (ORDBMS) is a relational database management system that allows developers to integrate the database with their own custom data types and methods. The term object-relational database is sometimes used to describe external software products running over traditional DBMSs to provide similar features; these systems are more correctly referred to as object-relational mapping systems.

Whereas only relational products focused on the efficient management of data drawn from a limited set of data types (defined by the relevant language standards), an object-relational DBMS allows software developers to integrate their own types and the methods that apply to them into the DBMS. The goal of ORDBMS technology is to allow developers to raise the level of abstraction at which they view the problem domain.

Then we can define “Object-Relational Mapping” (ORM) as a programming technique for converting data between incompatible type systems in databases and Object-oriented programming languages. This creates, in effect, a "virtual object database" which can be used from within the programming language.

2.3.1. Problem Description

Data management tasks in object-oriented (OO) programming are typically implemented by manipulating objects, which are almost always non-scalar values.

Consider the example of an address book entry, which represents a single person along with zero or more phone numbers and zero or more addresses. This could be modeled in an object-oriented implementation by a "person object" with "slots" to hold the data that comprise the entry: the person's name, a list (or array) of phone numbers, and a list of addresses. The list of phone numbers would itself contain "phone number objects" and so on. The address book entry is treated as a single value by the programming language (it can be referenced by a single variable, for instance). Various methods can be associated with the object, such as a method to return the preferred phone number, the home address, and so on.

Many popular database products, however, such as SQL DBMS products, can only store and manipulate scalar values such as integers and strings, organized within tables.

The programmer must either convert the object values into groups of simpler values for storage in the database (and convert them back upon retrieval), or only use simple scalar values within the program. Object-relational mapping is used to implement the first approach. The difficulty of the problem is translating those objects to forms which can be stored in the database, and which can later be retrieved easily, while preserving the properties of the objects and their relationships; these objects are then said to be persistent.

2.3.2. Implementations

The most common type of database used is the SQL database, which predates the rise of object-oriented programming in the 1990s. SQL databases use a series of tables to organize data. Data in different tables is associated through the use of declarative constraints, rather than explicit pointers or links. The same data that can be stored in a single object value, would need to be stored across several of these tables.

According to [12], an object-relational mapping implementation would need to systematically and predictably choose which tables to use and generate the necessary SQL. In overview, the impedance mismatch between the architectural approach of the object oriented application such as built in Java and where the data is stored in a relational database management system (RDBMS) such as Oracle or IBM's DB2 creates a rather complex set of challenges to deal with in order to accomplish tasks such as; performance, linear scalability, manage complex operations without slowing down, make maintenance and future application changes simple requiring little or no effort, etc. The real values in using an ORM tool is to save time, simplify development (i.e. the ORM tool handles the complexity for the developer), increase performance or scalability, and minimize architectural challenges related to inability of the ORM tool or developer's experience.

Many packages have been developed to reduce the tedium of developing object-relational mapping systems by providing libraries of classes which are able to perform mappings automatically. Given a list of tables in the database, and objects in the program, they will automatically map requests from one to the other. Asking a person object for its phone numbers will result in the proper query being created and sent, and the results being translated directly into phone number objects inside the program.

From a programmer's perspective, the system should look like a persistent object store. One can create objects and work with them as one would normally, and they automatically end up in the database.

2.4. Multimedia Databases

For the development of this thesis we will focus on the multimedia databases and above all on some aspects about images and metadata which will be useful for the future work. In this section the reader can find some of this concepts like the content based retrieval or the metadata embedded in some multimedia formats.

2.4.1. Definition

Multimedia is the use of several different media (e.g. text, audio, graphics, animation, video, and interactivity) to convey information. *Multimedia* also refers to the use of computer technology to create, store, and experience multimedia content.

Applications that use databases are quite extended since they appeared at the end of 80's. And these kinds of databases have been hardly studied. Typically these databases store numbers and text but nowadays there are many kind of information different to numbers and text that must be stored and retrieved. This is the reason because of a new kind of databases appeared multimedia databases.

These databases store multimedia data like documents, images, audio and video that actually has a great relevance in many areas.

One example of these areas where multimedia databases are used is e-learning: multimedia presentations and courseware are stored in open database architecture and system, which is shared by instructors, students, and administrators of distance learning programs. Similarly, e-commerce system requires large database entries to store customer and product information. Behavior of the customers can also be recorded such that recommendation of new products can be delivered. Video-on-Demand is a new entertainment dimension. Unlike traditional video store, movies and music can be delivered to customer whenever and wherever necessary. [4]

2.4.2. Purpose

In multimedia databases it is possible to store images, audio, large amount of documents and video that adjust to the necessities of the new application. But, what do these databases different to a normal storage in a hard disk? The answer to this question is simple, as well as storing the information, they store some relevant features of the media that make easy the retrieval of the information. Moreover, this "extra" information is used to expand the capability of databases. For example, we can search for images with a specific name but we can try to search for some similar the one we provide to the system. This diffuse characteristic makes too interesting multimedia databases.

Talking about e-learning, image and video information retrieval is one of interesting research areas. How to select two pictures of similar features effectively is still an unsolved problem. Current technologies rely on the comparison of colour, texture, shape, and spatial features of objects in the pictures. However, there exists a gap between human perception and efficient computing. Yet, the automatic segmentation of video clips into shots and scenes in order to realize automatic indexing and searching not only involves image processing techniques, but the computation of temporal properties of video is essentially important. Also, since video clips require higher capacity of storage, the efficient allocation of video data on the disk and memory is a key issue to fast retrieval.

Querying a multimedia database requires new querying strategies. Multimedia queries are hard to formulate explicitly. For example, try to explain what music you like. Often, it is far easier to show an example document and have the system identify similar documents. This querying strategy is better known as query-by-example.

Unlike traditional database systems, which allow query specification based on keywords and numerical comparisons, image database system requires a sophisticated retrieval mechanism. However, the most difficult issue of image content-based retrieval is the investigation of friendly visual specification techniques. How to visually describe the need of a user is a very difficult problem. On the other hand, content-based retrieval of video records not only involves the objects in a video, the timing of object movement is also considered. Scene identification and object tracing are basic techniques, which only solve part of the problem. Yet, tools for semantic analysis of video contents are still underdevelopment. Content-based video retrieval may rely for example on speech detection and recognition, which are also used in the automatic retrieval of audio information.

2.4.3. Content Based Retrieval

Inexpensive image-capture and storage technologies have allowed massive collections of digital images to be created. However, as an image database grows, the difficulty of finding relevant images increases. Two general approaches to this problem have been developed. Both use metadata for image retrieval:

- Using information manually entered or included in the table design, such as titles, descriptive keywords from a limited vocabulary, and predetermined classification schemes. This approach follows the classical perspective, based on normal queries about text or numerical data. This, it not always the best way to manipulate the data included on a image, moreover, this textual description must be introduced by the user so it is not a good idea if we are working with large amount of images.

- Using content-Based retrieval (CBIR) first mentioned in [14], it is based on automated image feature extraction and object recognition to classify image content, that is possible using capabilities unique to content-based retrieval. In this approach we have two advantages, the extraction of the information is quickly and automated and more information about the image it is used to make the queries.

There are several content that we can looking for as color, shape or texture, we can use either of them depending on what are we looking for. At once that we decided to work with the content of an image the next point is to decide which measure use to decide that two images match. There are several possibilities and the reader can take a deeper look at [6][15].

2.4.4. Metadata in Multimedia

As we mentioned before a multimedia database is something more than storage and an interface to access the data.

First difference appears when new data is inserted in the database. At this moment, some data is extracted. For example, if a new image is loaded in a database, information about its size or codification is extracted and stored. This data is formally called metadata.

This process can be manual or automatic. It depends on the DBMS and the data that we collect. For example, the encoding type of an image is automatically collected by most of the systems. With this features a description of the data is built and stored for later use as an XML schema or like binary data for transport.

A user may use this features to make a query the database for browsing the data for manipulate it or retrieving the described content.

Chapter 3. Metadata

3.1. Introduction

Metadata is structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information. The term metadata is used differently in different communities. Some use it to refer to machine understandable information, while others use it only for records that describe electronic resources. In the library environment, metadata is commonly used for any formal scheme of resource description, applying to any type of object, digital or non-digital.

In [16] is said that "Metadata is key to ensuring that resources will survive and continue to be accessible into the future."

3.2. Types of metadata

Metadata can take several forms, some of which will be visible to the user of a digital library system, while others operate behind the scenes. The Digital Library Foundation (DLF), a coalition of 15 major research libraries in the USA, defines three types of metadata which can apply to objects in a digital library:

- **descriptive metadata**: information describing the intellectual content of the object, such as MARC cataloguing records, finding aids or similar schemes

- **administrative metadata:** information necessary to allow a repository to manage the object: this can include information on how it was scanned, its storage format etc (often called technical metadata), copyright and licensing information, and information necessary for the long-term preservation of the digital objects (preservation metadata)

- **structural metadata:** information that ties each object to others to make up logical units (for example, information that relates individual images of pages from a book to the others that make up the book itself)

In general, only descriptive metadata is visible to the users of a system, who search and browse it to find and assess the value of items in the collection. Administrative metadata is usually only used by those who maintain the collection, and structural metadata is generally used by the interface which compiles individual digital objects into more meaningful units (such a journal volumes) for the user.

An important reason for creating descriptive metadata is to facilitate discovery of relevant information. In addition to resource discovery, metadata can help organize electronic resources, facilitate interoperability and legacy resource integration, provide digital identification, and support archiving and preservation.

3.3. Structuring metadata

Metadata schemes (also called schema) are sets of metadata elements designed for a specific purpose, such as describing a particular type of information resource. The definition or meaning of the elements themselves is known as the semantics of the scheme. The values given to metadata elements are the content.

Metadata schemes generally specify names of elements and their semantics. Optionally, they may specify content rules for how content must be formulated (for example, how to identify the main title), representation rules for content (for example, capitalization rules), and allowable content values (for example, terms must be used from a specified controlled vocabulary). There may also be syntax rules for how the elements and their content should be encoded.

A metadata scheme with no prescribed syntax rules is called syntax independent. Metadata can be encoded in any definable syntax. Many current metadata schemes use SGML (Standard Generalized Mark-up Language) or XML (Extensible Mark-up Language). XML, developed by the World Wide Web Consortium (W3C), is an extended form of HTML that allows for locally defined tag sets and the easy exchange of structured

3.3.1. Metadata Scheme example: MPEG

The ISO/IEC Moving Picture Experts Group (MPEG) has developed a suite of standards for coded representation of digital audio and video. Two of the standards address metadata: MPEG-7, Multimedia Content Description Interface (ISO/IEC 15938), and MPEG-21, Multimedia Framework (ISO/IEC 21000), as is explained in [16].

MPEG-7 defines the metadata elements, structure, and relationships that are used to describe audiovisual objects including still pictures, graphics, 3D models, music, audio, speech, video, or multimedia collections. It is a multipart standard that addresses:

- Description Tools including Descriptors that define the syntax and the semantics of each metadata element and Description Schemes that specify the structure and semantics of the relationships between the elements.

- A Description Definition Language to define the syntax of the Description Tools, allow the creation of new Description Schemes, and allow the extension and modification of existing Description Schemes.

- System tools, to support storage and transmission, synchronization of descriptions with content, and management and protection of intellectual property.

Descriptors for visual and audio are defined separately using a hierarchy of elements and subelements. For visual objects there are descriptors for Basic Structure, Color, Texture, Shape, Motion, Localization, and Face Recognition. Audio descriptors are divided into two categories: low-level descriptors that are common to audio objects across most applications, and high-level descriptors that are specific to particular applications of audio. The cross-application low-level descriptors cover Structures and Features (temporal and spectral). The domain-specific high-level descriptors include such elements as Musical Instrument Timbre, Melody Description, and Spoken Content Description.

The Description Schemes are based on XML, and can be expressed in textual form suitable for editing, searching, filtering, and human readability; or in a binary form for storage, transmission, and streaming delivery. Since the full description of a multimedia object can be quite complex, the standard provides for a Summary Description Scheme geared to browsing and navigation.

The standard envisions that search engines could use MPEG-7 metadata descriptions to identify audiovisual objects in entirely new ways, such as digitizing a musical phrase played on a keyboard and then retrieving a list of musical pieces that contain the sequence of notes; drawing some lines on an electronic drawing tablet and retrieving images with similar graphics; or using a voice phrase to retrieve related speech files, photographs, video clips, and biographical information of the speaker. These retrieval mechanisms are outside the scope of MPEG-7, but the standards developers wanted to accommodate these futuristic capabilities and have included many interoperability requirements beyond the typical metadata elements.

MPEG-21 was developed to address the need for an overarching framework to ensure interoperability of digital multimedia objects.

3.4. XML: the language for metadata systems

A decision was reached very early in the planning of metadata for the ODL that it should be expressed in the eXtensible Markup Language (XML). This is a language designed initially for marking up electronic text, but which has since then been used for a wide variety of metadata applications. “Its advantages for metadata encoding are many: they include its robustness, its software independence and hence its ready interchangeability between systems, and the way in which its structure maps neatly to that of many digital objects.” [17]

An XML system can be expressed in two ways: the first, and longer established, system is the *Document Type Definition (DTD)*, which lists what tags may be employed within an XML document, and also their content and relationships to each other. A much newer method of encoding an XML system is *XML Schema*, which expresses the rules an XML document has to follow in a further, separate XML document. XML Schema is much more powerful than a DTD, but because it is so new only limited software is currently available to handle it (although the number of such packages is increasing very quickly).

Chapter 4. DICOM Standard

4.1. Introduction

The initial goal in developing a standard for the transmission of digital images was to enable users to retrieve images and associated information from digital imaging equipment in a standard format that would be the same across multiple manufacturers. As we can observe in [8], the first result was the American College of Radiology (ACR)-National Electrical Manufacturers' Association (NEMA) standard, which specified a point-to-point connection. However, the rapid evolution of computer networking and of picture archiving and communication systems meant that this point-to-point standard would be of limited use. Consequently, a major effort was undertaken to redesign the ACR-NEMA standard by taking into account existing standards for networks and current concepts in the handling of information on such networks. The Digital Imaging and Communications in Medicine (DICOM) standard was the result of this effort. Its popularity has made discussion, if not implementation, of the standard common whenever digital imaging systems are specified or purchased.

The DICOM standard is extremely adaptable, a planned feature that has led to the adoption of DICOM by other specialties that generate images (eg, pathology, endoscopy, dentistry). The fact that many of the medical imaging equipment manufacturers are global corporations has sparked considerable international interest in DICOM. The European standards organization, the Comitâ Europâen de Normalisation, uses DICOM as the basis for the fully compatible MEDICOM standard. In Japan, the Japanese Industry Association of Radiation Apparatus and the Medical Information Systems Development Center have adopted the portions of DICOM that pertain to exchange of images on removable media and are considering DICOM for future versions of the Medical Image Processing Standard. The DICOM standard is now being maintained and extended by an international, multispecialty committee.

The DICOM standard has become the predominant standard for the communication of medical images. However, even though the standard is widely available from manufacturers and is rapidly expanding to include non-radiologic imaging, most radiologists' understanding of it is limited. In part, this is because DICOM has a "steep learning curve" and most introductory material has been written either for the engineer and is highly technical, or for the administrator and is rather superficial.

Why all the interest in what would seem to be a simple task? The answer is that it is not as simple as it first appears. Most radiologists are familiar primarily with film images, and film can be viewed anywhere there is a light source. It is the transition from film images to digital images and the need to communicate, display, and store these images that has made DICOM necessary. With film, slight differences in exposure, processing, and viewing will have little effect in these areas. In digital imaging, however, the difference of a few bytes can make it impossible to transfer an image from one system to another.

The DICOM standard consists of multiple documents; there are 16 published parts. Each DICOM document is identified by title and standard number, which takes the form "PS 3.X-YYYY," where "X" is commonly called the part number and "YYYY" is the year of publication. For example, DICOM Part 2 has a title of "Conformance" and document number PS 3.2-1996. In informal usage, the year is often dropped.

4.2. History

In an effort to develop a standard means by which users of digital medical imaging equipment (such as computed tomography, magnetic resonance imaging, nuclear medicine, and ultrasound) could interface display or other devices to these machines, the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) formed a joint committee early in 1983. The mission of this group, the ACR-NEMA Digital Imaging and Communications Standards Committee, was to find or develop an interface between imaging equipment and whatever the user wanted to connect. In addition to specifications for the hardware connection, the standard to be developed was to include a dictionary of the data elements needed for proper image display and interpretation. For a more detailed view of this standard history the reader can consult [9].

The committee surveyed many existing interface standards, but none was found that was entirely satisfactory. Some, however, were found to contain useful ideas. The American Association of Physicists in Medicine (AAPM) had, about a year before, developed a standard format for recording images on magnetic tape. The header portion would contain a description of the image along with the data elements (such as patient name) identifying it. A concept of using data elements of variable length identified with a tag or key (the name of the element) was thought to be particularly important and was adopted by the committee.

After 2 years of work, the first version of the standard, ACR-NEMA 300-1985 (also called ACR-NEMA Version 1.0) was distributed at the 1985 RSNA annual meeting and published by NEMA. As with many first versions, errors were found and improvements were suggested. The committee had empowered Working Group (WG) VI to follow up on the standard after it was published. This WG answered many questions from potential developers and began working on changes to improve the standard. In 1988, ACR-NEMA 300-1988 (or ACR-NEMA Version 2.0) was published. It used substantially the same hardware specification as Version 1.0, but it added new data elements and fixed a number of errors and inconsistencies.

The problem was that by 1988 many users wanted an interface between imaging devices and a network. While this could be accomplished with Version 2.0, the standard lacked the parts necessary for robust network communication. For example, one could send a device a message that contained header information and an image, but one would not necessarily know what the device would do with the data. Since ACR-NEMA Version 2.0 was not designed to connect equipment directly to a network, solving these problems meant major changes to the standard. The committee had very early adopted the idea that future versions of the ACR-NEMA Standard would retain compatibility with the earlier versions, and this placed some constraints on WG VI.

In a decision of major importance for the standard, it was decided that developing an interface for network support would require more than just adding patches to Version 2.0. The entire design process had to be re-engineered, and the method adopted was that of object-oriented design. Later sections will describe this process briefly.

In addition, a thorough examination of the types of services needed to communicate over different networks showed that defining a basic service would allow the top layer of the communications process (the application layer) to talk to a number of different network protocols. These protocols are modeled as a series of layers, often referred to as "stacks." The existing Version 2.0 stack that defined a point-to-point connection was one. Two others were chosen based on popularity and future expansion: the Transmission Control Protocol/Internet Protocol (TCP/IP) and the International Standards Organization Open Systems Interconnection (ISO-OSI). Figure 1 shows a diagram of the communication model developed. The basic design philosophy was that a given medical imaging application (which is outside of the scope of the standard) could communicate over any of the stacks to another device that used the same stack. With adherence to the standard, it would be possible to switch the communications stacks without having to rewrite the computer programs of the application.

4.3. Goals of DICOM standard

The DICOM Standard facilitates interoperability of devices claiming conformance. In particular, following [10] this standard:

- Addresses the semantics of Commands and associated data. For devices to interact, there must be standards on how devices are expected to react to Commands and associated data, not just the information which is to be moved between devices;

- Addresses the semantics of file services, file formats and information directories necessary for off-line communication;

- Is explicit in defining the conformance requirements of implementations of the Standard. In particular, a conformance statement must specify enough information to determine the functions for which interoperability can be expected with another device claiming conformance.

- Facilitates operation in a networked environment.

- Is structured to accommodate the introduction of new services, thus facilitating support for future medical imaging applications.

- Makes use of existing international standards wherever applicable, and itself conforms to established documentation guidelines for international standards.

Even though the DICOM Standard has the potential to facilitate implementations of PACS solutions, use of the Standard alone does not guarantee that all the goals of a PACS will be met. This Standard facilitates interoperability of systems claiming

conformance in a multi-vendor environment, but does not, by itself, guarantee interoperability.

This Standard has been developed with an emphasis on diagnostic medical imaging as practiced in radiology, cardiology and related disciplines; however, it is also applicable to a wide range of image and non-image related information exchanged in clinical and other medical environments.

Figure 4.1 presents the general communication model of the Standard which spans both network (on-line) and media storage interchange (off-line) communication. Applications may relay on either on of the following boundaries:

- The Upper Layer Service, which provides independence from specific physical networking communication support and protocols such as TCP/IP.
- The Basic DICOM File Service, which provides access to Storage Media independently from specific media storage formats and file structures.

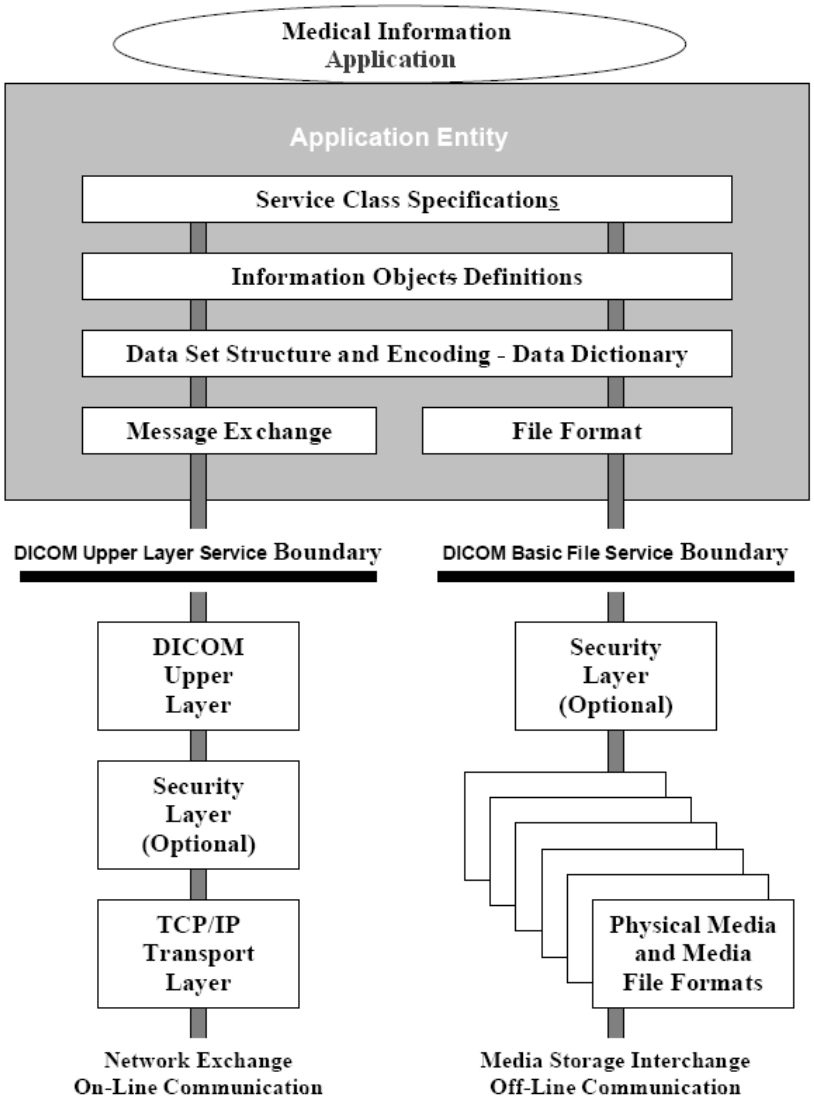


Figure 4.1: The DICOM general communication model [10]

4.4. Parts of DICOM Standard

Unlike ACR-NEMA Versions 1.0 and 2.0, DICOM divides much of the specification into parts. This was done so that parts could be expanded (e.g., new information object definitions added) without having to republish the whole standard. Within the parts, those sections subject to addition or modification are in annexes, further reducing the editing required when updating parts. The current version of DICOM consists of sixteen parts. The interrelationships of the DICOM parts are not always readily apparent. Figure 4.2 is a diagram showing how the parts are related. This figure also shows parts 10 and 11, which address the way DICOM can use files on removable media (e.g., disk and tape) for exchange of information. [10]

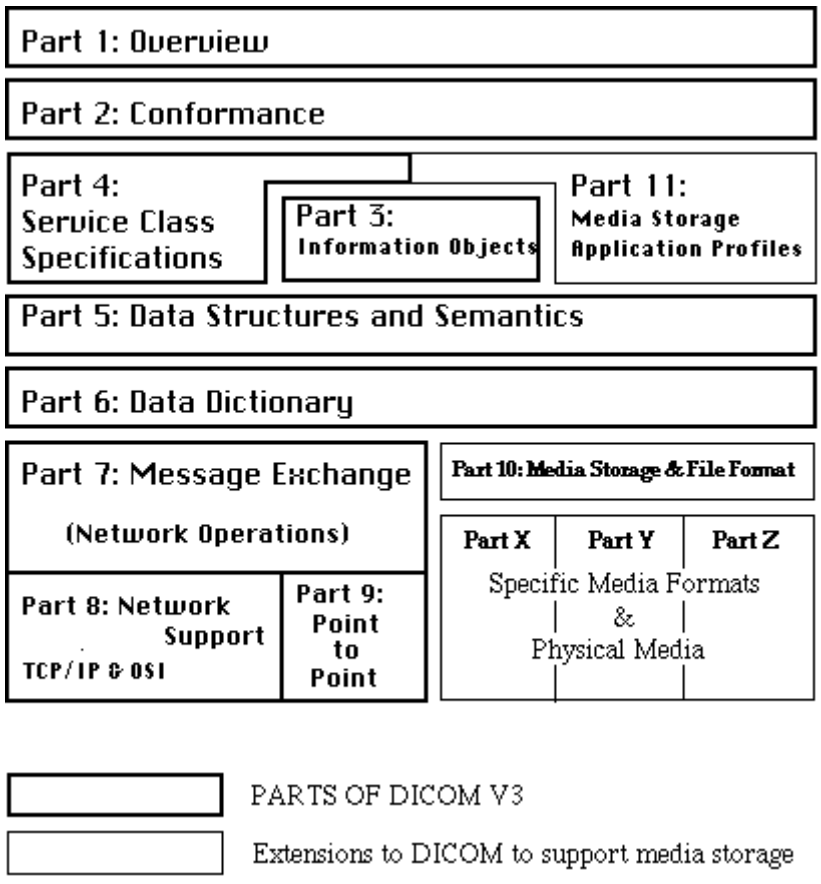


Figure 4.2: Some present parts and proposed extension of DICOM [9]

This figure is not a layered model. The left hand portion represents the parts that define network and point-to-point DICOM communications. The right hand portion shows the parts that support communication using removable storage media. Note that some parts (parts 1, 2, 3, 5, and 6) are used in both environments while others are particular to the specific communications domain.

The new specification of 2007 includes a total of sixteen parts:

- PS 3.1: Introduction and Overview (this document)
- PS 3.2: Conformance
- PS 3.3: Information Object Definitions
- PS 3.4: Service Class Specifications
- PS 3.5: Data Structure and Encoding
- PS 3.6: Data Dictionary
- PS 3.7: Message Exchange
- PS 3.8: Network Communication Support for Message Exchange
- PS 3.9: Retired
- PS 3.10: Media Storage and File Format for Data Interchange
- PS 3.11: Media Storage Application Profiles
- PS 3.12: Media Formats and Physical Media for Data Interchange
- PS 3.13: Retired
- PS 3.14: Grayscale Standard Display Function
- PS 3.15: Security Profiles
- PS 3.16: Content Mapping Resource

These parts of the Standard are related but independent documents.

4.5. DICOM single-file format

PS 3.10 of the standard describes a file format for the distribution of images. This format is an extension of the older NEMA standard. Most people refer to image files which are compliant with PS 3.10 of the DICOM standard as DICOM format files.

A single DICOM file contains both a header (which stores information about the patient's name, the type of scan, image dimensions, etc), as well as all of the image data (which can contain information in three dimensions). This is different from the popular Analyze format, which stores the image data in one file (*.img) and the header data in another file (*.hdr). Another difference between DICOM and Analyze is that the DICOM image data can be compressed (encapsulated) to reduce the image size. Files can be compressed using lossy or lossless variants of the JPEG format, as well as a lossless Run-Length Encoding format (which is identical to the packed-bits compression found in some TIFF format images).

DICOM is the most common standard for receiving scans from a hospital. Neuroimagers and neuropsychologists who wish to use SPM to normalize scans to stereotaxic space will need to convert these files to Analyze format.

4.5.1. The DICOM header

In the next page The Figure 4.3 shows a hypothetical DICOM image file. In this example, the first 794 bytes are used for a DICOM format header, which describes the image dimensions and retains other text information about the scan. The size of this header varies depending on how much header information is stored. Here, the header defines an image which has the dimensions 109x91x2 voxels, with a data resolution of 1 byte per voxel (so the total image size will be 19838). The image data follows the header information (the header and the image data are stored in the same file). [11]

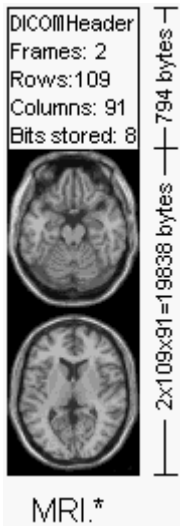


Figure 4.3: Hypothetic DICOM image file [11]

First 128 bytes: unused by DICOM format
Followed by the characters 'D','I','C','M'
This preamble is followed by extra information e.g.:

0002,0000,File Meta Elements Group Len: 132
0002,0001,File Meta Info Version: 256
0002,0010,Transfer Syntax UID: 1.2.840.10008.1.2.1.
0008,0000,Identifying Group Length: 152
0008,0060,Modality: MR
0008,0070,Manufacturer: MRIcro
0018,0000,Acquisition Group Length: 28
0018,0050,Slice Thickness: 2.00
0018,1020,Software Version: 46\64\37
0028,0000,Image Presentation Group Length: 148
0028,0002,Samples Per Pixel: 1
0028,0004,Photometric Interpretation: MONOCHROME2.
0028,0008,Number of Frames: 2
0028,0010,Rows: 109
0028,0011,Columns: 91
0028,0030,Pixel Spacing: 2.00\2.00
0028,0100,Bits Allocated: 8
0028,0101,Bits Stored: 8
0028,0102,High Bit: 7
0028,0103,Pixel Representation: 0
0028,1052,Rescale Intercept: 0.00
0028,1053,Rescale Slope: 0.00392157
7FE0,0000,Pixel Data Group Length: 19850
7FE0,0010,Pixel Data: 19838

Figure 4.4: Example of DICOM header [11]

The Figure 4.4 shows a more detailed list of the DICOM header. Note that DICOM requires a 128-byte preamble (these 128 bytes are usually all set to zero), followed by the letters 'D', 'I', 'C', 'M'. This is followed by the header information, which is organized in 'groups'. For example, the group 0002hex is the file meta information group, and (in the following example on the left) contains 3 elements: one defines the group length, one stores the file version and the third stores the transfer syntax.

The DICOM elements required depends on the image type, and are listed in Part 3 of the DICOM standard. For example, this image modality is 'MR' (see group:element 0008:0060), so it should have elements to describe the MRI echo time. The absence of this information in this image is a violation of the DICOM standard. In practice, most DICOM format viewers do not check for the presence of most of these elements, extracting only the header information which describes the image size.

The NEMA standard preceded DICOM, and the structure is very similar, with many of the same elements. The main difference is that the NEMA format does not have the 128-byte data offset buffer or the lead characters 'DICM'. In addition, NEMA did not explicitly define multi-frame (3D) images, so element 0028,0008 was not present.

Of particular importance is group:element 0002:0010. This defines the '**Transfer Syntax Unique Identification**' (see the Figure 4.5). This value reports the structure of the image data, revealing whether the data has been compressed. Note that many DICOM viewers can only handle uncompressed raw data. DICOM images can be compressed both by the common lossy JPEG compression scheme (where some high frequency information is lost) as well as a lossless JPEG scheme that is rarely seen outside of medical imaging (this is the original and rare Huffman lossless JPEG, not the more recent and efficient JPEG-LS algorithm). These codes are described in Part 5 of the DICOM standard.

Transfer Syntax UID	Definition
1.2.840.10008.1.2	Raw data, Implicit VR, Little Endian
1.2.840.10008.1.2.x	Raw data, Eplicit VR x = 1: Little Endian x = 2: Big Endian
1.2.840.10008.1.2.4.xx	JPEG compression xx = 50-64: Lossy JPEG xx = 65-70: Lossless JPEG
1.2.840.10008.1.2.5	Lossless Run Length Encoding

Figure 4.5: Transfer Syntax Unique Identification Table

Note that as well as reporting the compression technique (if any), the Transfer Syntax UID also reports the byte order for raw data. Different computers store integer values differently, so called 'big endian' and 'little endian' ordering. Consider a 16-bit

integer with the value 257: the most significant byte stores the value 01 (=255), while the least significant byte stores the value 02. Some computers would save this value as 01:02, while others will store it as 02:01. Therefore, for data with more than 8-bits per sample, a DICOM viewer may need to swap the byte-order of the data to match the ordering used by your computer. [11]

In addition to the Transfer Syntax UID, the image is also specified by the Samples Per Pixel (0028:0002), Photometric Interpretation (0028:0004), the Bits Allocated (0028:0100). For most MRI and CT images, the photometric interpretation is a continuous monochrome (e.g. typically depicted with pixels in grayscale). In DICOM, these monochrome images are given a photometric interpretation of 'MONOCHROME1' (low values=bright, high values=dim) or 'MONOCHROME2' (low values=dark, high values=bright). However, many ultrasound images and medical photographs include color, and these are described by different photometric interpretations (e.g. Palette, RGB, CMYK, YBR, etc). Some colored images (e.g. RGB) store 3-samples per pixel (one each for red, green and blue), while monochrome and paletted images typically store only one sample per image. Each images store 8-bits (256 levels) or 16-bits per sample (65,535 levels), though some scanners save data in 12-bit or 32-bit resolution. So a RGB image that stores 3 samples per pixel at 8-bits per can potentially describe 16 million colors (256 cubed).

Chapter 5. Oracle interMedia

5.1. What is interMedia

According to [1], Oracle interMedia can be defined as a feature that enables Oracle Database to store, manage, and retrieve images, audio, video, or other heterogeneous media data in an integrated fashion with other enterprise information. Oracle interMedia extends Oracle Database reliability, availability, and data management to multimedia content in traditional, Internet, electronic commerce, and media-rich applications. Oracle interMedia does not control media capture or output devices; this function is left to application software.

interMedia manages multimedia content by providing the following:

- Storage and retrieval.
- Media and application metadata management.
- Support for popular formats.
- Access through traditional and Web interfaces.
- Querying using associated relational data.
- Querying using extracted metadata.
- Querying using media content with optional specialized indexing.

5.2. interMedia Capabilities

The capabilities of interMedia include the storage, retrieval, management, and manipulation of multimedia data managed by Oracle Database.

Multimedia applications have common and unique requirements. interMedia object types support common application requirements and can be extended to address application-specific requirements. With interMedia, multimedia data can be managed as easily as standard attribute data.

Is accessible to applications through both relational and object interfaces. Database applications written in Java, C++, or traditional third-generation languages (3GLs) can interact with interMedia through modern class library interfaces, or PL/SQL and Oracle Call Interface (OCI).

It supports storage of the popular file formats, including desktop publishing images, and streaming audio and video formats in databases. interMedia provides the means to add audio, image, and video, or other heterogeneous media columns or objects

to existing tables, and insert and retrieve multimedia data. This enables database designers to extend existing databases with multimedia data, or to build new end-user multimedia database applications. interMedia developers can use the basic functions provided here to build specialized multimedia applications.

interMedia uses object types, similar to Java or C++ classes, to describe multimedia data. These object types are called ORDAudio, ORDDoc, ORDImage, and ORDVideo. We can see an example of ORDImage in the Figure 5.1. An instance of these object types consists of attributes, including metadata and the media data, and methods. **Media data** is the actual audio, image, or video, or other heterogeneous media data. In the Figure 5.1 the Media data is stored n the `source` attribute. **Metadata** is information about the data, such as object length, compression type, or format. **Methods** are procedures that can be performed on the object, such as `getContentLenght()` and `setProperties()`.

The interMedia objects have a common media data storage model. The media data component of these objects can be stored in the database, in a BLOB under transaction control. The media data can also be stored outside the database, without transaction control. In this case, a pointer is stored in the database under transaction control, and the media data is stored in:

- File-based large object (BFILE)
- An HTTP server-based URL
- A user-defined source on a specialized media data server, or other server

Media data stored outside the database can provide a convenient mechanism for managing large, existing or new, media repositories that reside as flat files on erasable or read-only media. This data can be imported into BLOBs at any time for transaction control.

The Oracle interMedia storage model includes a common set of operations for multimedia content:

- **BLOB operations:** Load, fetch, and delete multimedia content
- **External operations:** Open, close, trim (clip), read/write a buffer, store in a temporary BLOB, import/export between the external source and a BLOB
- **Other operations:** Extract multimedia metadata, set storage metadata, perform data manipulation, and pass commands to external data storage.

Media metadata is stored in the database under interMedia control. Whether media data is stored within or outside the database, interMedia manages metadata for all the media types and may automatically extract it for audio, image, and video.

In addition to metadata extraction methods, a minimal set of image manipulation methods is provided. For images, this includes performing format conversion, page selection, and quantize operations, and compression, scaling, cropping, copying, flipping, mirroring, rotating, and adjusting the gamma (brightness) of images.

It is extensible. It supports a base set of popular audio, image, and video data formats for multimedia processing that also can be extended, for example, to support additional formats, new digital compression and decompression schemes (**codecs**), data sources, and even specialized data processing algorithms for audio and video data.

Is a building block for various multimedia applications rather than being an end-user application. It consists of object types along with related methods for managing and processing multimedia data. Some example applications for interMedia are:

- Repositories for digital check images.
- Electronic health records, including DICOM medical images.
- Call centers (for example, Emergency Numbers and product call centers)
- Physical asset inventories.
- Distance learning and online learning.
- Real estate marketing.
- Stock photography archives (for example, digital art galleries and professional photographers).
- Document imaging archives.
- Financial news service customer information.
- Web publishing.

5.3. interMedia and Images

This section contains information about digitized image concepts and using the ORDImage object type to build image applications or specialized ORDImage objects.

5.3.1. Digitized Images

ORDImage integrates the storage, retrieval, and management of digitized images in a database.

ORDImage supports two-dimensional, static, digitized raster images stored as binary representations of real-world objects or scenes. Images may be produced by a document or photograph scanner, a video source such as a digital camera or VCR connected to a video digitizer or frame grabber, other specialized image capture devices, or even by program algorithms. Capture devices take an analog or continuous signal such as the light that falls onto the film in a camera, and convert it into digital values on a two-dimensional grid of data points known as pixels. Devices involved in the capture and display of images are under application control.

5.3.2. Image Components

Digitized images consist of the image data (digitized bits) and attributes that describe and characterize the image data. Image applications sometimes associate application-specific information, such as the name of the person pictured in a photograph, description of the image, date photographed, photographer, and so forth, with image data by storing this descriptive text in an attribute or column in the database table.

The image data (pixels) can have varying depths (bits per pixel) depending on how the image was captured, and can be organized in various ways. The organization of the image data is known as the data format. ORDImage can store and retrieve image data of any data format. ORDImage can process and automatically extract properties of images of a variety of popular data formats. We can see in the next picture the aspect of an ORDImage object:

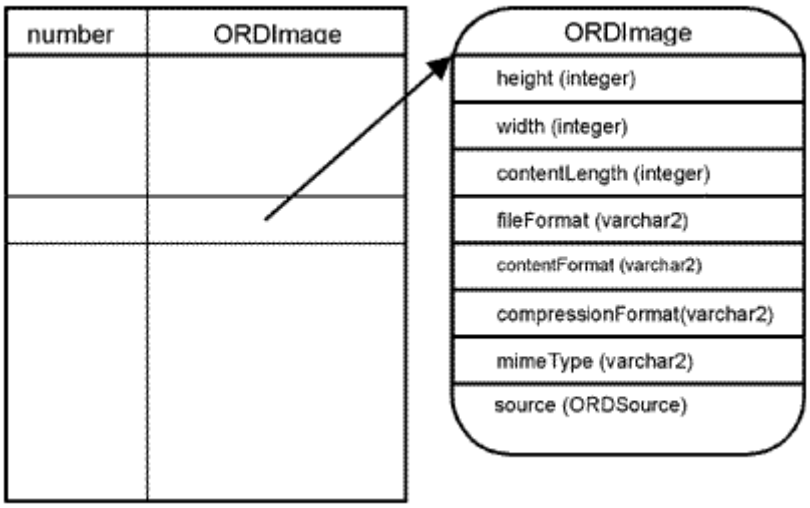


Figure 5.1: ORDImage type structure [1]

- **height**: the height of the image in pixels.
- **width**: the width of the image in pixels.
- **contentLength**: the size of the on-disk image file in bytes.
- **fileFormat**: file type or format in which the image data is stored (TIFF, JIFF...).
- **contentFormat**: the type of image (monochrome and so forth).
- **compressionFormat**: the compression algorithm used on the image data.
- **mimeType**: the MIME type information.
- **source**: the source of the stored image data.

In addition, certain foreign images (formats not natively supported by ORDImage) have limited support for image processing.

The storage space required for digitized images can be large compared to traditional attribute data such as numbers and text. Many compression schemes are available to squeeze an image into fewer bytes, thus reducing storage device and network load. Lossless compression schemes squeeze an image so that when it is decompressed, the resulting image is bit-for-bit identical with the original. Lossy compression schemes do not result in an identical image when decompressed, but rather, one in which the changes may be imperceptible to the human eye. As compared with lossless schemes, lossy schemes generally provide higher compression.

Image interchange format describes a well-defined organization and use of image attributes, data, and often compression schemes, allowing different applications to create, exchange, and use images. Interchange formats are often stored as disk files. They may also be exchanged in a sequential fashion over a network and be referred to as a **protocol**. There are many application subdomains within the digitized imaging world and many applications that create or utilize digitized images within these. ORDImage supports storage and retrieval of all image data formats, and processing and attribute extraction of many image data formats.

5.3.3. Image Processing

interMedia supports image processing, such as image format transcoding, image cutting, image scaling, and generating thumbnail images. In addition, specifically when the destination image file format is RAW Pixel (RPIX) format or Microsoft Windows Bitmap (BMPF) image format, interMedia supports a variety of operators for changing the format characteristics.

The more powerful tools provided by interMedia for processing images, are the `process()` and `processCopy()` methods:

- **`process()`:** Performs one or more image processing operations on a BLOB, writing the image back onto itself.
- **`processCopy()`:** Copies an image stored internally or externally to another image stored internally in the source LocalData attribute (of the embedded ORDSource object) and performs one or more image processing operations on the copy.

These methods receive a command as a parameter. This command consists of a serie of modifications that will be executed over the image. Some of the operators we can use are: `fileFormat`, `scale`, `rotate`, `mirror`, `maxScale`, `gamma`, `flip`, `contrast`, `compressionFormat`, `compressionQuality`, `xScale`, `yScale`, `cut`,...

We can see more examples in the section 6.1 and in particular at Code 6.6 where from a DICOM image we obtain a JPEG image and its thumbnail using the `fileFormat` and `maxScale` commands.

5.3.4. Image support for Java

Oracle provides through its Oracle.Ord.Im package a complete Java API to work with interMedia.

Java programmers are intimately familiar with Java objects, but they are often unaware that Oracle Database is an object-relational database, and as such supports storage and retrieval of objects. As we saw before, Oracle interMedia provides the database type ORDImage which is used to store images in a database table just like any other relational data. Some interMedia functionality (such as thumbnail generation) may also be used if images are stored in BLOB (Binary Large Object) columns, but Oracle Corporation recommends storing images in ORDImage columns. The reader can review the structure of ORDImage type in the previous Figure 5.1. and remember that this data type allows the access to this data so we can work with their values.

Even though the JDBC specification does not support object-relational databases directly, Oracle interMedia database objects can be used in JDBC programs by means of the interMedia Java Client. The interMedia Java Client contains high performance proxy Java objects that allow quick object property retrieval and convenient upload/download. The proxies forward any requests for the ORDImage object computation back to the database server.

A schematic diagram of how a database ORDSYS.ORDImage object is related to the Java ORDImage object is shown below on Figure 5.2. It is easy to see that ORDImage Java objects are merely proxies for database objects — they must be created from a database ORDImage object.

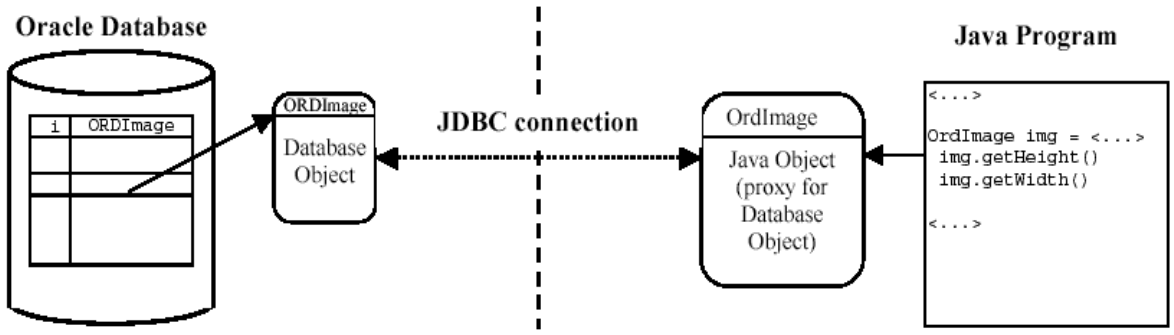


Figure 5.2: Relation between ORDSYS.ORDImage and Java ORDImage Object [1]

The connection to the database with JDBC is similar to a connection to a normal database unless one detail: Oracle InterMedia uses BLOB columns internally to store data. This implies that if the `autoCommit` flag must be set to false or any operation that involves BLOB´s will fail. This flag is put to false with the instruction `conn.setAutoCommit(false);` afterwards create the connection to the database.

5.4. interMedia and Metadata

5.4.1. Metadata concepts

Image files can contain information about the content of the images, the image rasters and image metadata. In general, data about data is referred to as **metadata**. In this case, metadata refers to additional information about the actual images, which is stored in the image files along with the images.

Several types of metadata can be stored in an image file, and each type can serve a different purpose. One type, **technical metadata**, is used to describe an image in a technical sense. For example, technical metadata can include attributes about an image, such as its height and width, in pixels, or the type of compression used to store it. Another type, **content metadata**, can further describe the content of an image, the name of the photographer, and the date and time when a photograph was taken.

Metadata is stored in image files using a variety of mechanisms. Digital cameras and scanners automatically insert metadata into the images they create. Digital photograph processing applications like Adobe Photoshop allow users to add or edit metadata to be stored with the image. Annotating digital images with additional metadata is a common practice in photographic and news gathering applications and for image archiving usages, as well as at the consumer level.

Storing metadata together with image data in the same containing file provides encapsulation. With encapsulation, both types of data can be shared and exchanged reliably as one unit. Metadata that is stored in the image file format is referred to as **embedded metadata** and this is the case of the DICOM images.

As was said before, in 5.2 media metadata is stored in the database under interMedia control. Whether media data is stored within or outside the database, interMedia manages metadata for all the media types and may automatically extract it for audio, image, and video. This metadata includes the following attributes:

- Storage information about audio, image, and video, or other heterogeneous media data, including the source type, location, and source name, and whether the data is stored locally (in the database) or externally
- Update time stamp information for audio, image, and video, or other heterogeneous media data.
- Audio and video data description.
- Audio, image, and video, or other heterogeneous media data format.
- MIME type of the audio, image, and video, or other heterogeneous media data.
- Audio characteristics: encoding type, number of channels, sampling rate, sample size, compression type, and play time (duration).
- Image characteristics: height and width, image content length, image content format, and image compression format.

- Video characteristics: frame width and height, frame resolution, frame rate, play time (duration), number of frames, compression type, number of colors, and bit rate.
- Extracted metadata in XML, such as the director or producer of a movie.

5.4.2. Metadata in Images

Oracle Database 10g, Release 2 adds an image metadata feature to interMedia. The Metadata feature enhances the current behavior of the interMedia ORDImage object type by adding the ability to read (or extract) and write (or embed) application metadata in images. In addition, this feature adopts a standard way to represent metadata when it is separate from an image file. Metadata can be stored in a database, indexed, searched, and made available to applications using the standard mechanisms of Oracle Database.

Oracle Database 10g, Release 2 also adds the Digital Imaging and Communications in Medicine (DICOM) feature to interMedia. The DICOM feature enhances the current behavior of the interMedia ORDImage object type by allowing interMedia to recognize standalone DICOM objects and extract a subset of embedded DICOM attributes relating to patient, study, and series.

5.4.3. Managing Metadata

For a large number of image file formats, Oracle interMedia ("interMedia") can extract and manage a limited set of metadata attributes. These attributes include: height, width, contentLength, fileFormat, contentFormat, compressionFormat, and mimeType. For a limited number of image file formats, interMedia can extract a rich set of metadata attributes. This metadata is represented in schema-based XML documents. These XML documents can be stored in a database, indexed, searched, updated, and made available to applications using the standard mechanisms of Oracle Database.

interMedia can also write or embed metadata supplied by users into a limited number of image file formats. The application provides the metadata as a schema-based XML document. interMedia processes the XML document and writes the metadata into the image file.

Once metadata has been extracted and stored, you can index the metadata for powerful full text and thematic media searches using Oracle Text. Thus, the database can be queried to locate the media data based on the metadata extracted from the media.

5.5. interMedia and DICOM

As we said before (p. 35) Oracle Database 10g Release 2 adds DICOM as a supported format for the ORDImage data type. This data type is similar in structure and function to a Java Class or a C++ object. Using this data type, any column of any table can hold DICOM or other image types. The DICOM medical image is another attribute

of the entity that is stored in the table. Using a simple relational query on a table containing patient information, you can also retrieve the associated DICOM images. [1]

The ORDImage object used for DICOM images includes the following components:

- A set of simple image format attributes
- The DICOM image contents which is identical to the information delivered to the database
- A set of methods used to store, manipulate, or retrieve DICOM images

5.5.1. DICOM Metadata

Image format metadata includes attributes such as image height, width, and compression scheme. These format attributes are stored with the images and are used to help the application determine how to display those images.

Application metadata is supplied by image creators and stored with the images in the database. In the case of DICOM format images, application metadata can include patient information, physician information, modality type, and series and study identifiers. Image metadata can be parsed from the DICOM image at any time. After parsing, metadata is returned to the application as an object of XMLType. Using XML DB technology, the application can use this metadata for display or store it in another column of XMLType for indexing purposes.

5.5.2. DICOM Object Content

To preserve the quality of the original medical image, the bits used to represent the pixels of a DICOM format image are identical to those of the original DICOM image that was imported into the database. Thus, the entire DICOM file object is preserved exactly as it was presented to the database. The DICOM standard requires this replication for quality and legal reasons. New DICOM images can be created by the application code to correct mistakes in the original application metadata, to make an image anonymous, or to display only a portion of an original image. The original image, however, must not be modified.

5.5.3. DICOM Image Methods

Methods are functions or procedures that can perform operations on the DICOM format image objects stored in the ORDImage data type. The most important methods are as follows:

- **`getDicomMetadata()`:** This method parses application metadata, such as patient name and modality, from the DICOM image making it easily available as an XML document. This XML document is returned to the application as XMLType and can be inserted into another column in the database, indexed, and searched using standard database features such as XML DB or Oracle Text. XML

DB searches support Xpath queries that can be used to find one or more specific tagged metadata attributes. Searches can also be based on any Boolean combination of attributes, which can enable powerful longitudinal studies.

- **`processCopy()`:** This method makes it possible to copy and convert a DICOM image into a JPEG image or any other popular supported formats for display in a browser or publication in a document. For example, this method can produce a new image with JPEG format while leaving the original DICOM image untouched. The JPEG format image can be scaled to an appropriate size for use as a clickable thumbnail in an application, printed in a report, or delivered with the same resolution as the original DICOM image for display in a browser.
- **`export()`:** This method copies data from the DICOM image to a corresponding external file.
- **`import()`:** This method transfers image data from an external image data source, such as a DICOM file, to the database.

5.5.4. DICOM Image Validation

In [1] Oracle ensures that Oracle Database 10g Release 2 can accept any DICOM file for storage in an ORDImage column. The file will be returned to the application exactly as it was delivered to the database, fulfilling the requirement that DICOM images cannot be modified. When the `getDicomMetadata()` method is used to extract the DICOM metadata, interMedia validates the metadata to assure that it complies with the DICOM standard. interMedia returns an error if the metadata does not comply with the requirements of the DICOM standard, and interMedia does not generate an XML representation of the DICOM metadata.

Optionally, the database administrator can specify that the DICOM validation should return an XML representation of as much of the DICOM metadata as can be read and interpreted. In this case, valid DICOM fields will be returned as XML tags. Invalid fields will be marked in the XML document and a binary or text representation of the invalid data will be delivered to the application

5.5.5. DICOM Storage Alternatives

Oracle believes that there are compelling advantages to storing DICOM images directly in the database. These advantages include access control, auditing, atomic transactions, encryption, and simple, consistent, and powerful management tools. Storing the images directly in the database is not a requirement. DICOM images can be stored separately in files, while the local file specification or URL can be stored in the database.

The `import()` and `export()` methods of the ORDImage data type can be used to move DICOM image data back and forth between the database and the file system. The `getDicomMetadata()` and `processCopy()` methods can be used to retrieve metadata and images in different formats from the DICOM image, even if the DICOM image is stored separately from the database in a file.

The Oracle interMedia DICOM technology supports the uploading and downloading of DICOM images to and from Oracle Database. Both uploading and downloading between the file system and the database can be performed using a simple PL/SQL procedure. In addition, uploading can be performed on a bulk set of images using PL/SQL procedures, SQL*Loader, and DataPump. Using any of these methods allows for rapid loading of large sets of images.

5.5.6. DICOM Image Viewing

DICOM images are typically viewed in one of two ways:

- Radiologists usually want to see images using specialized viewers that can read and interpret DICOM images directly. These viewers typically have fine controls for zooming, panning, and adjusting contrast. Delivering a DICOM image stored in an Oracle database to a radiologist's workstation is a simple matter of invoking the `export()` method after locating the appropriate table row that contains the DICOM image column.
- Clinicians and referring doctors usually want to view images from any location using standard personal computers and a standard Web browser. This can be easily accomplished using the `processCopy()` method.

Chapter 6. interMedia-Java Samples

6.1. Image insertion into database

One of the most important things that we must do in a multimedia database is to insert new data. In this particular case we will insert in the database a DICOM image, a copy of it in JPEG format, a thumbnail of the JPEG and the related DICOM metadata in a separate field. This is the typical problem we will need to solve for the most of the applications with DICOM images over a database.

Imagine that we have in our Oracle database a table like this:

IMAGE_DICOM				
id (NUMBER)	**file** (ORDIMAGE)	**image_jpg** (ORDIMAGE)	**thumb** (ORDIMAGE)	**metadata** (XMLTYPE)
...	...	...	...	...

Figure 6.1: Table example for image insertion

The "id" field is a numeric value to identify the image that we store. The next field "file" is a ORDImage and it store the dicom file understated it like a amount of bit stored in a BLOB. The next one "image_jpg" stores only the image part of the DICOM file in jpeg format. "thumb" will have a smaller version of the previous column and finally "metadata" stores the metadata embedded in the DICOM file with a XML structure.

The process of insertion new image data is not very similar to a normal insertion and it consists on three stages:

- Create a new row in the table. We can not leave the ORDImages fields empty or subsequent actions will fail. For this reason we must use a creation method provided by Oracle for this data type. A normal insertion statement could be:

```
insert into IMAGE_DICOM
      (id, file, image_jpg, thumb, metadata)
values
      (1,ordsys.ordimage.init(),ordsys.ordimage.init(),
      ordsys.ordimage.init(),NULL);
```

Code 6.1: Insertion of new row for Dicom Image

- After that we have created a new row but the fields are empty. We must insert or upload the images in the database. For that, first we must obtain the OrdImage field in which we will insert the image executing the instruction:

```
select
      file, image_jpg, thumb, metadata
from
      IMAGE_DICOM
where
      id =1 for update;
```

Code 6.2: Selection of the new row for Dicom Image

Then, we must create the proxies linked to this images, this operation make us possible to use the pool of methods provided by Oracle for working with images. The proxies is created using:

```
OrdImage file_proxy  =
      (OrdImage) rset.getCustomDatum(1,OrdImage.getFactory());

OrdImage image_proxy =
      (OrdImage) rset.getCustomDatum(2, OrdImage.getFactory());

OrdImage thumb_proxy =
      (OrdImage) rset.getCustomDatum(3, OrdImage.getFactory());
```

Code 6.3: Creation of proxies linked to new Images

Where `rset` contains the result of the previous instruction. Then, we have to load the DICOM file using the method of the OrdImage object and the URL of the file as a parameter:

```
file_proxy.loadDataFromFile(nameOfTheImage);
```

Code 6.4: Upload Image File to proxy

After that we should use the method `setProperties()` of the new image. This way interMedia writes the characteristics of the foreign image into the appropriate attribute fields of the OrdImage object, based on a set of characteristics that describes the image properties. With this information, interMedia is able to process certain foreign image formats.:

```
file_proxy.setProperties();
```

Code 6.5: Set properties of the DICOM File

After that we copy and process the dicom image file to obtain the desired transformations and the metadata:

```
file_proxy.processCopy("fileFormat=JFIF", image_proxy);
image_proxy.processCopy("maxScale=50,50", thumb_proxy);
XMLType metadata = file_proxy.getDicomMetadata("imageGeneral");
```

Code 6.6: Process of the DICOM file

- Finally, we only need to update the current data in the proxies for the database to be coherent. We can do it with the following code:

```
OraclePreparedStatement stmt =
   (OraclePreparedStatement) conn.prepareStatement
   (
      "update IMAGE_DICOM "+
      "set file=?, image_jpg=?, thumb=?, metadata=? "+
      "where ID_VISIT = 1"
   );

stmt.setCustomDatum(1, file_proxy);
stmt.setCustomDatum(2, image_proxy);
stmt.setCustomDatum(3, thumb_proxy);
stmt.setString(4, metadata.getStringVal());
stmt.execute();
stmt.close();
```

Code 6.7: Update of the row into the database

This previous steps are common to the insertion of every image, the only difference is that in DICOM images we can extract metadata. In summary:

1) Insertion of a new row with the value `ordsys.ordimage.init()` in the image fields.

2) Selection of this new row for creating a proxy to the image field.

3) Load of the image from the file into the proxy.

4) Process the image in the proxy if needed

5) Update of the field in the database row with the modified proxy.

6.2. Image download from database

The process of downloading an image to a local directory is less complicated than upload to the database. Supposing again that we are working over the table shown in Figure 6.1, and that we want to download the JPEG image to our file system, the code we should use for this operation is:

```
OraclePreparedStatement stmt;

String query = "select image_jpg from IMAGE_DICOM"+
               " where id = 1";

OracleResultSet rset =
      (OracleResultSet)stmt.executeQuery(query);

rset.next();

OrdImage imageProxy =
      (OrdImage)rset.getCustomDatum(
            "image_jpg",OrdImage.getFactory());

rset.close();

imageProxy.getDataInFile(nameOfTheDownloadedImage);
```

Code 6.8: Download of a image in the database

As in the case of uploading an image, you must:

1) Query for the row which contains the image in one of its fields.

2) Create the proxy to the correspondent image field

3) Invoke a method of the proxy object for downloading the image to the file system.

This steps and code are common to all the image formats including DICOM.

Chapter 7. Sample Application: "DICOM Manager"

7.1. Introduction

The main aim in the construction of the sample application is to show how DICOM images and their metadata can be managed in a Oracle – Java environment. For showing the DICOM functionalities, a medical aid application was chosen. It consists in a consultations register where a doctor can do the following actions:

- Add visits of a specific stored patient enclosing or not DICOM images and a summary of the visit.
- Search previous visits following different search criterions. Id, Name, Summary, Visit Date...
- Visualize the DICOM images related to a visit and their metadata in a tree structure.

This application as well as being a sample, could be also useful for exploitation in a real medical environment and can be easily upgraded with new functionalities. Moreover this application can be used in the client computer like stand-alone or in http environment like an applet which make it perfect for working from a browser over the net of a hospital.

After the implementation this application is definitely usable except for some developments missed due to be out of the objectives of the thesis. For example, security issues.

7.2. Analysis and Design

7.2.1. Architecture

The architecture of the system will be very easy and based on the client-server schema. The innovative characteristic are that, as was explained in the previous page, the program could run as stand-alone or applet fitting the two-tier or three-tier architectures respectively as the pictures show below:

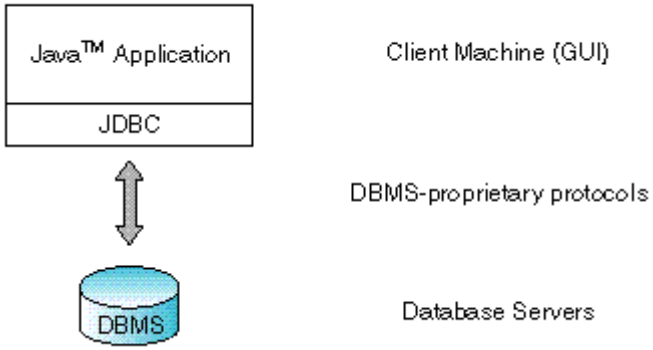


Figure 7.1: Two-tier architecture

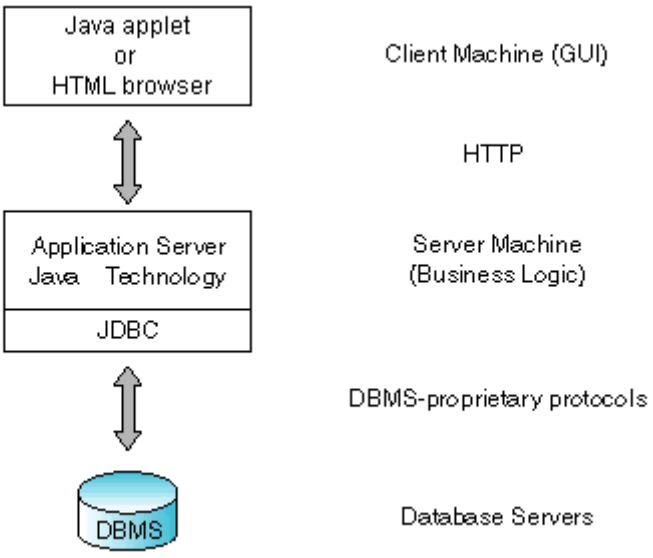


Figure 7.2: Three-tier architecture

The connection with the database is trough the API provided by Oracle to Java called Java Database Connectivity (JDBC). This API is the industry standard for database-independent connectivity between the Java programming language and a wide range of databases.

7.2.2. Use Cases

For demonstration purposes of the sample, the application only considers an actor called "**Doctor**" as we can see in the following diagram (Figure 7.3). That is because only a doctor is supposed to be able of log into the application.

At the beginning the doctor must **Log In** for accessing to all the other functionalities. Only then, the doctor can:

- **Insert a new visit** which always includes **searching the specific patient** of the visit and optionally **insert or delete DICOM images** from the temporary list of images that will be inserted in this new visit.
- **Search a patient data.**
- **Search a previous visit.**
- **Visualize a DICOM Image** related to a specific **searched visit.**
- **Log Out**.

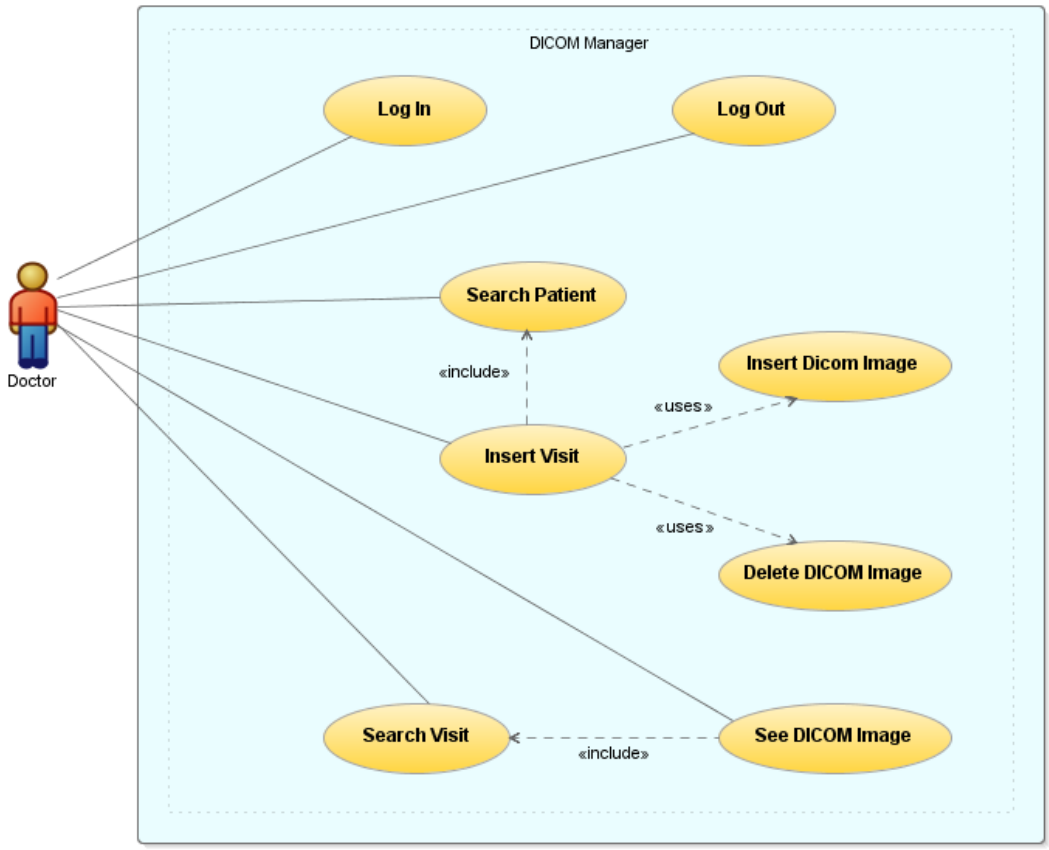


Figure 7.3: Use Case Diagram

For understand easily how each use case works and the relations between them, the following pages are dedicated to the formal description for all of them.

UC01 - Log In

Use Case-01	**Log In**	
Description	The main actor needs to log in at the welcome screen for having available the other functionalities of the application. The login and password are the correspondent for accessing the selected database.	
Pre-condition	The main actor must be logged out (UC02) (default at start-up).	
Normal sequence	**Step**	**Action**
	1	The user press the Log In button.
	2	The system logs in the user in the chosen database with the login and the password provided by the doctor in the correspondent fields.
	3	The system enables the other disabled functionalities as buttons or tabs.
	4	The system retrievals the data related to the doctor logged in and show the information in the welcome screen.
Post-condition	The doctor is logged in.	
Exceptions	**Step**	**Action**
	2	If the system is not able to login the doctor shows a message with the probable cause. For example: “Login/Pass error” , “DB connection error”... and the use case ends.
Frequency	At least one time for execution.	
Significance	High	
Urgency	Medium	

Figure 7.4: UC01 - Log In

UC02 - Log Out

Use Case-02	**Log Out**	
Description	Is necessary for changing the user or for ending properly the connection with the database before exiting.	
Pre-condition	The user must be logged in (UC01).	
Normal sequence	**Step**	**Action**
	1	The user presses the Log Out button.
	2	The system drops the connection.
	3	The system clear the data of the logged out doctor in the welcome screen and disables all the functionalities except Log In.
Post-condition	The doctor is logged out.	
Frequency	At least one time for execution.	
Significance	Medium	
Urgency	Medium	

Figure 7.5: UC02 - Log Out

UC03 –Search Patient

Use Case-03	**Search Patient**	
Description	The patient is queried by id and the system shows his/her related data.	
Pre-condition	The user must be logged in (UC01).	
Normal sequence	**Step**	**Action**
	1	The user press the Search button in the “New Visit” tab.
	2	The system queries content of the patient id field and retrievals the related data showing it on the “Patient Data” panel.
	3	The system enables the options in the “Visit Summary” panel.
Post-condition	The data of the searched patient is loaded.	
Exceptions	**Step**	**Action**
	2	If the system is not able to find the selected patient in the database shows a message: “Patient ID not found” and the use case ends.
Frequency	High	
Significance	High	
Urgency	High	

Figure 7.6: UC03 - Search Patient

UC04 - Insert DICOM Image

Use Case-04	**Insert DICOM Image**	
Description	A new temporary DICOM Image is a added to the current visit of the current patient.	
Pre-condition	The current patient must be loaded by a search (UC03).	
Normal sequence	**Step**	**Action**
	1	The user press the ”Search...” button in the Visit Summary Panel.
	2	The system shows a file browser.
	3	The doctor chooses a DICOM file an press “OK”.
	4	The system shows the complete URL of the selected file.
	5	The doctor press the ”Add DICOM” button in the Visit Summary Panel.
	6	The system shows the thumbnail or the path in the list of temporary images in the Visit Summary Panel.
Post-condition	The list of temporary images for the visit has a new element.	
Exceptions	**Step**	**Action**
	5	If the image was already inserted in the temporary list, nothing is added to the list of temporary images and the URL field is cleared. The use case finishes.
	5	If the selected file hasn’t “.dcm” extension nothing is added to the list of temporary images and the URL field is cleared. The use case finishes.
Frequency	Medium	
Significance	High	
Urgency	Medium	

Figure 7.7: UC04 - Insert DICOM Image

UC05 - Delete DICOM Image

Use Case-05	**Delete DICOM Image**	
Description	A temporary DICOM Image is deleted from the current visit of the current patient.	
Pre-condition	The current visit must have at least a previous inserted DICOM Image (UC04).	
Normal sequence	**Step**	**Action**
	1	The user selects the thumbnail or the url of the desired image in the list of temporary images.
	2	The user press the ”Delete DICOM” button in the Visit Summary Panel.
	3	The system deletes the selected file from the temporary list.
Post-condition	The selected image is deleted from the temporary list.	
Exceptions	**Step**	**Action**
Frequency	Low	
Significance	Medium	
Urgency	Medium	

Figure 7.8: UC05 - Delete DICOM Image

UC06 - Insert Visit

Use Case-06	**Insert Visit**	
Description	A new visit is inserted in the database.	
Pre-condition	The current patient must be loaded by a previous search (UC03).	
Normal sequence	**Step**	**Action**
	1	The doctor can make up the list of the temporary images should be included in the visit following UC04 and UC05.
	2	The doctor can introduce a summary of the visit.
	3	The doctor presses the “Finish Visit” button.
	4	The system inserts the new visit and the images into the database.
	5	The system clears the information about the current patient and no patient data is loaded.
	6	The system disables the “Visit Summary” panel.
Post-condition	A new visit is inserted into the database.	
Exceptions	**Step**	**Action**
	4	If the visit doesn’t have any image on the temporary list and the summary field is empty. The system doesn’t make any insertion and keeps the current patient loaded.
Frequency	High	
Significance	High	
Urgency	Medium	

Figure 7.9: UC06 - Insert Visit

UC07 - Search Visit

Use Case-07	**Search Visit**	
Description	Retrieval of the visits which follows the chosen criterions.	
Pre-condition	The user must be logged in (UC01).	
Normal sequence	**Step**	**Action**
	1	The doctor presses the “Search” button in the “Search Visit Tab”.
	2	The system executes a query against the database with the parameters provided by the user in the fields of “Search Data” panel.
	3	The system shows the data retrieved in the “Search Results” table.
Post-condition	The table shows the results that fit the query parameters.	
Exceptions	**Step**	**Action**
Frequency	Medium	
Significance	High	
Urgency	Medium	

Figure 7.10: UC07 - Search Visit

UC08 – See DICOM Image

Use Case-08	**See DICOM Image**	
Description	A new tab shows a image and its metadata.	
Pre-condition	There is at least one result in a previous search of visits (UC07).	
Normal sequence	**Step**	**Action**
	1	The doctors select the desired rows in the “Search Results” table and press the “Show Selected Images” button.
	2	The system creates a new tab for each row selected, in which shows the metadata tree and the image.
	3	The use case finishes when the “Close” button of each new tab is pressed.
Post-condition	The system shows the desired images.	
Exceptions	**Step**	**Action**
	1	If the doctor didn’t select any image no new tabs are created.
Frequency	Medium	
Significance	High	
Urgency	Medium	

Figure 7.11: UC08 – See DICOM Image

7.2.3. Database Design

Now we can define the structure created for the database. We need to store mainly the data about the patients, the doctors and the visits with their images. The compact view of the entity-relationship diagram is as follows in Figure 7.12:

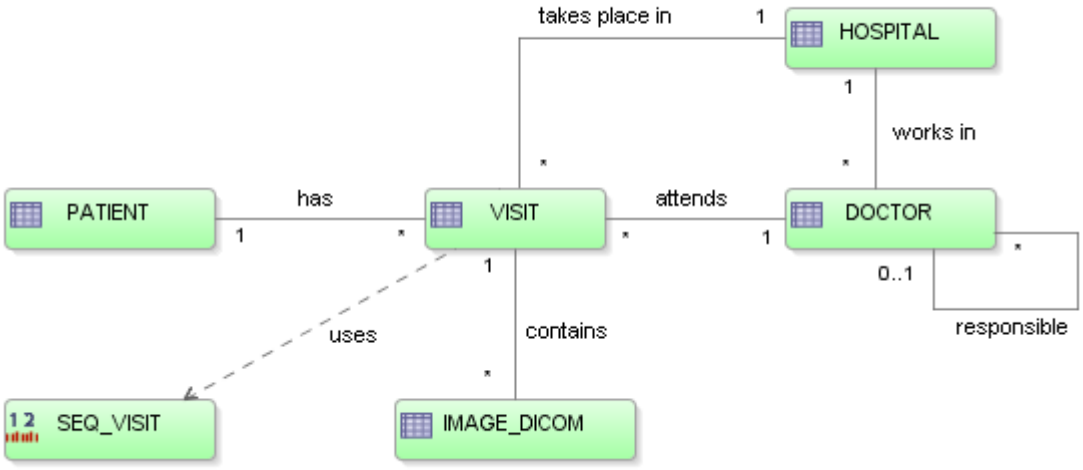


Figure 7.12: Entity-Relationship diagram (Compact)

The main idea is as follows: For storing visits we need the patient and the doctor involved, the hospital where the visit took place and all the images related to the visit. Other relations are; every doctor works in a hospital and every doctor can be responsible of other doctors.

The primary key of the visit is taken from a sequential number given by the "SEQ_VISIT" index and it has three different foreign keys for the patient, the doctor and the hospital of the visit.

The id for the patient is the unique identity number given by the government and similar for the doctors and hospitals. Moreover, the doctor has two foreign keys which determine the hospital where he/she works or the doctor who is his/her responsible.

More difficult is to understand the relation between a visit and its images. Helped by the next Figure 7.13, we observe that in every visit we can have zero or more images, then the primary key of the images is a compound key by the visit id (ID_VISIT) and a sequential number (SEQ) from 1 to the number of images for the same visit. Then if we imagine the visit number "2526" with three images, there should be three rows in the IMAGE_DICOM table which ID_VISIT is = "2526" and SEQ = "1" for the first image and "2" and "3" respectively for the followings.

We can see a more detailed view of this database structure with all the attributes and their types in the next diagram (Figure 7.13). We can take a look also to the primary and foreign keys explained before.

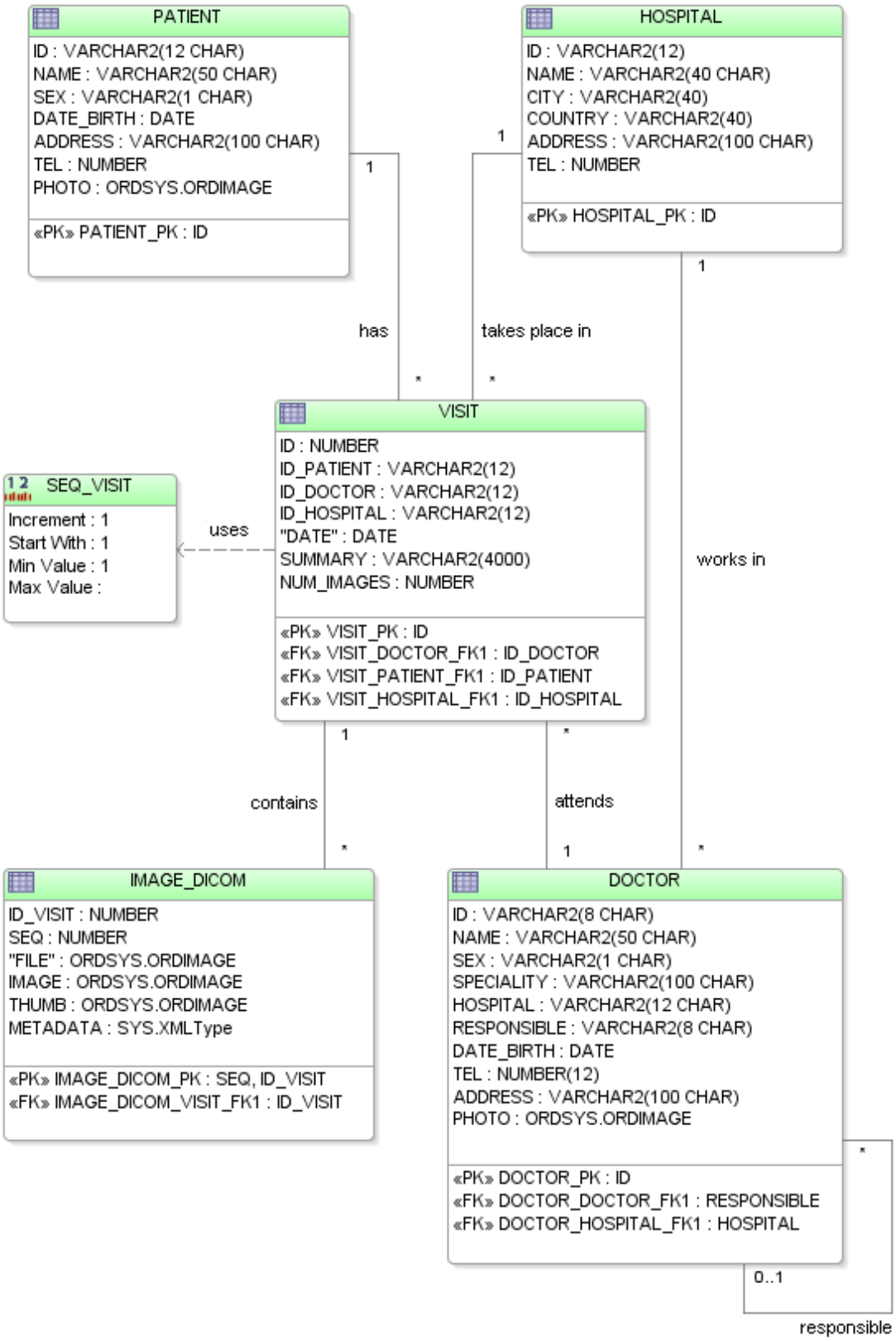


Figure 7.13: Database structure

7.2.4. Static Structure

Class Diagram

For a first glance and for making more comprehensible the application static structure, we can see as follows (Figure 7.14) a compact version of the class diagram where we can observe the relations between the more important classes in the program.

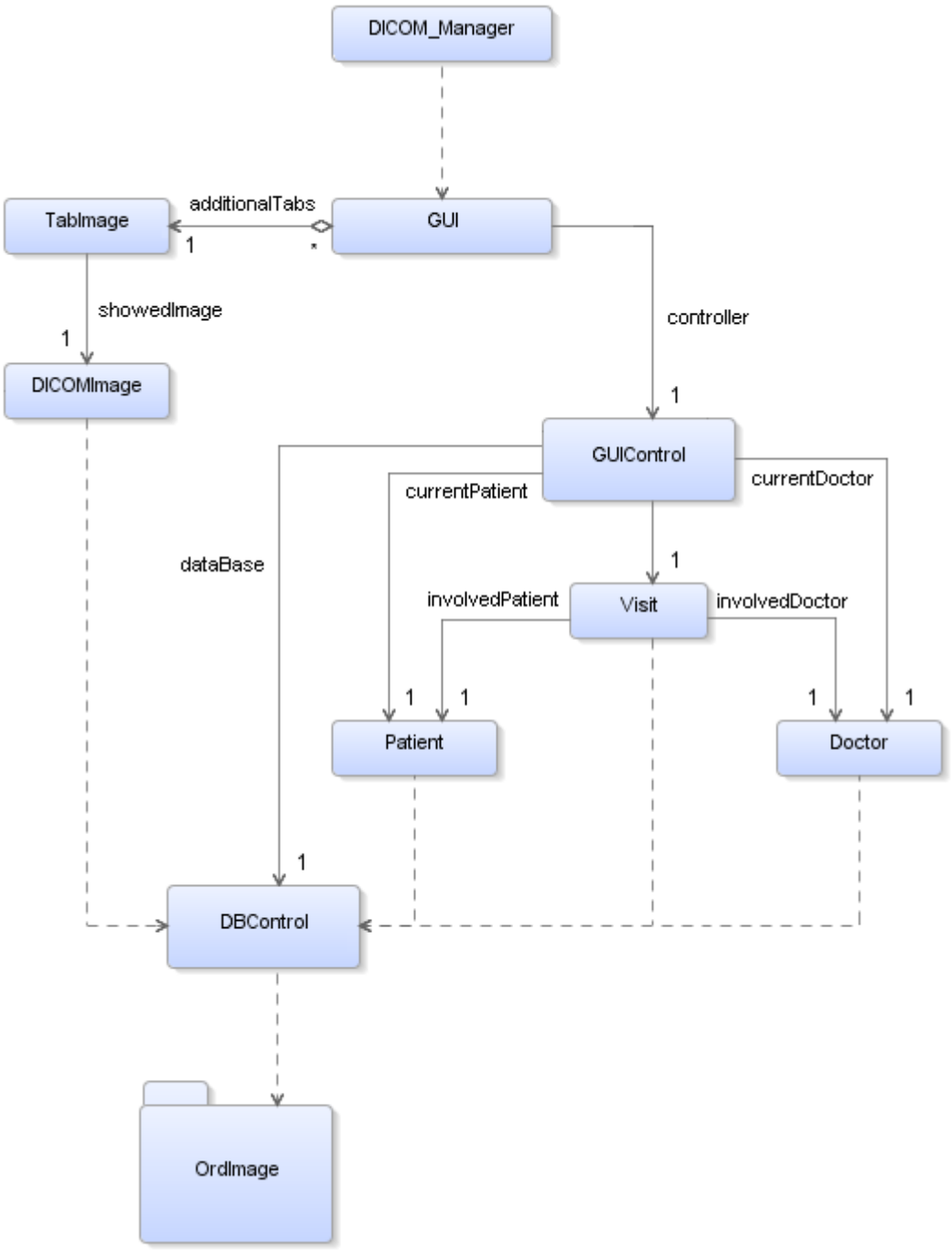


Figure 7.14: Class Diagram

For storing the current data model the “Doctor”, “Patient”, “Visit” and “DICOMImage” classes are necessary and their data are retrieved using the unique instance of the “DBControl” class. This class is the connection between the data model and the physical database and uses the Oracle.ord.im.OrdImage class and other from the Oracle interMedia library for working with the multimedia data stored. The other classes are only for presentation or control purposes and will be explained at the next dynamic structure section in

7.2.5. Dynamic Structure

First of all, and using as reference the previous Figure 7.14, we observe that the class that starts up the application in an applet or stand-alone is the “Dicom_Manager” class. This uses the “GUI” which is the class that stores all the Graphic User Interface components and their properties (panels, layouts, labels, fields...). When this class receives an event (for example a button pressed), the event is redirected to the “GUIControl” class which decides what to do with this event and triggers off the succession of actions needed for accomplishing the use case started by the event. The “GUI” class also contains a group of instances of “TabImage” and each one af this shows a “DICOMImage” instance.

7.2.6. Detailed Design

For having a clearer distribution of the classes and implementing a more understandable model, a package structure is proposed as follows in this section. We can see in detail the attributes or methods of all the classes.

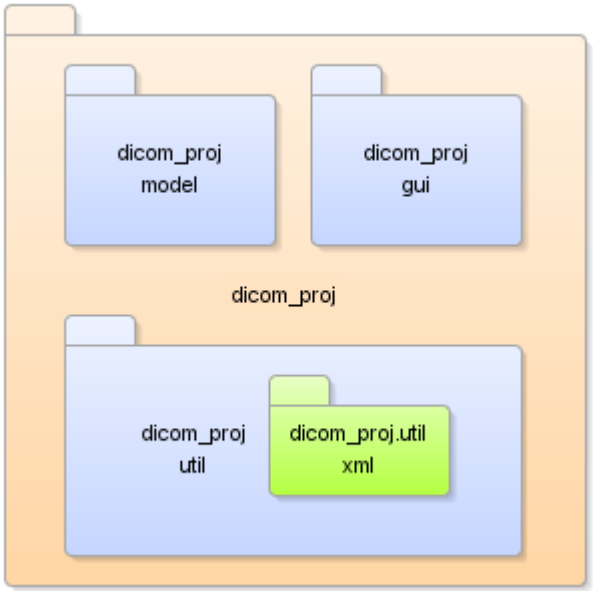


Figure 7.15: Package distribution

In the following pages there is a detailed view of each package and each class. All the developed classes for the application can be found here. There are more than in the previous class diagram showed in Figure 7.14 because some of them, like the xml utilities, are not relevant for understanding how the application works.

a) dicom_proj

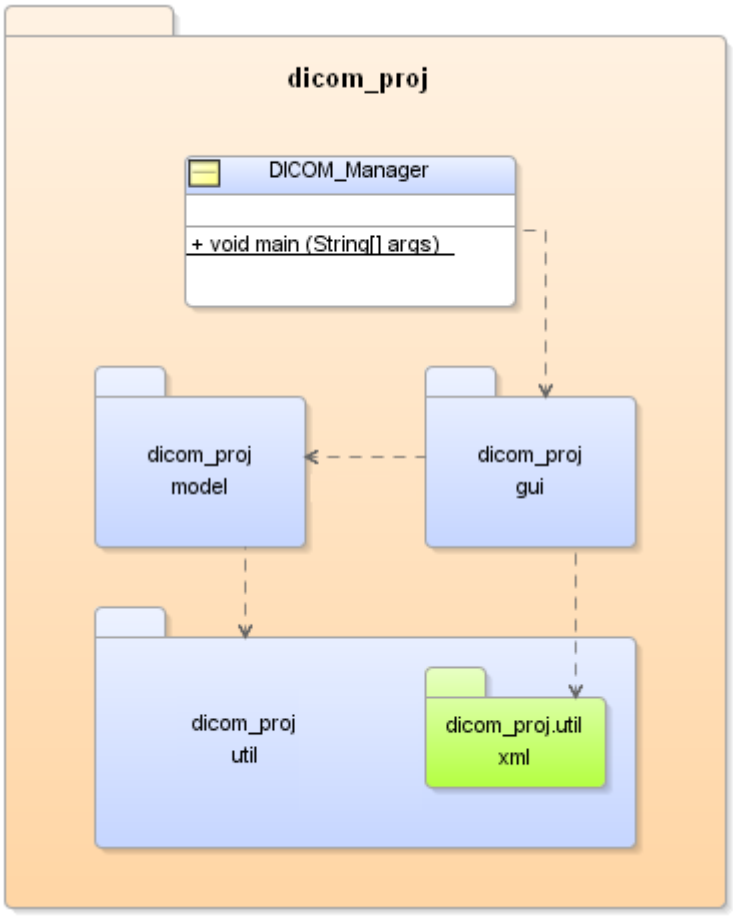


Figure 7.16: Package view - dicom_proj

We can see that the only class in this package is the launcher of the application; "DICOM_Manager" and only contains the main method. Moreover we can see the dependences between the other packages.

b) dicom_proj.gui

As is shown in this package, it is conformed by three classes. "GUI", stores all the visual elements of the graphic user interface and their characteristics. The attributes are not shown cause they are not relevant but at the end of the methods list we can see the correspondent to the press button events. Each of these methods will invoke the correspondent method in the "GUIControl" object and this is the responsible of triggering the needed actions for responding the user needs. "GUI" also contains a vector of "TabImage" elements where each one of this tabs will show a different DICOM image and their metadata

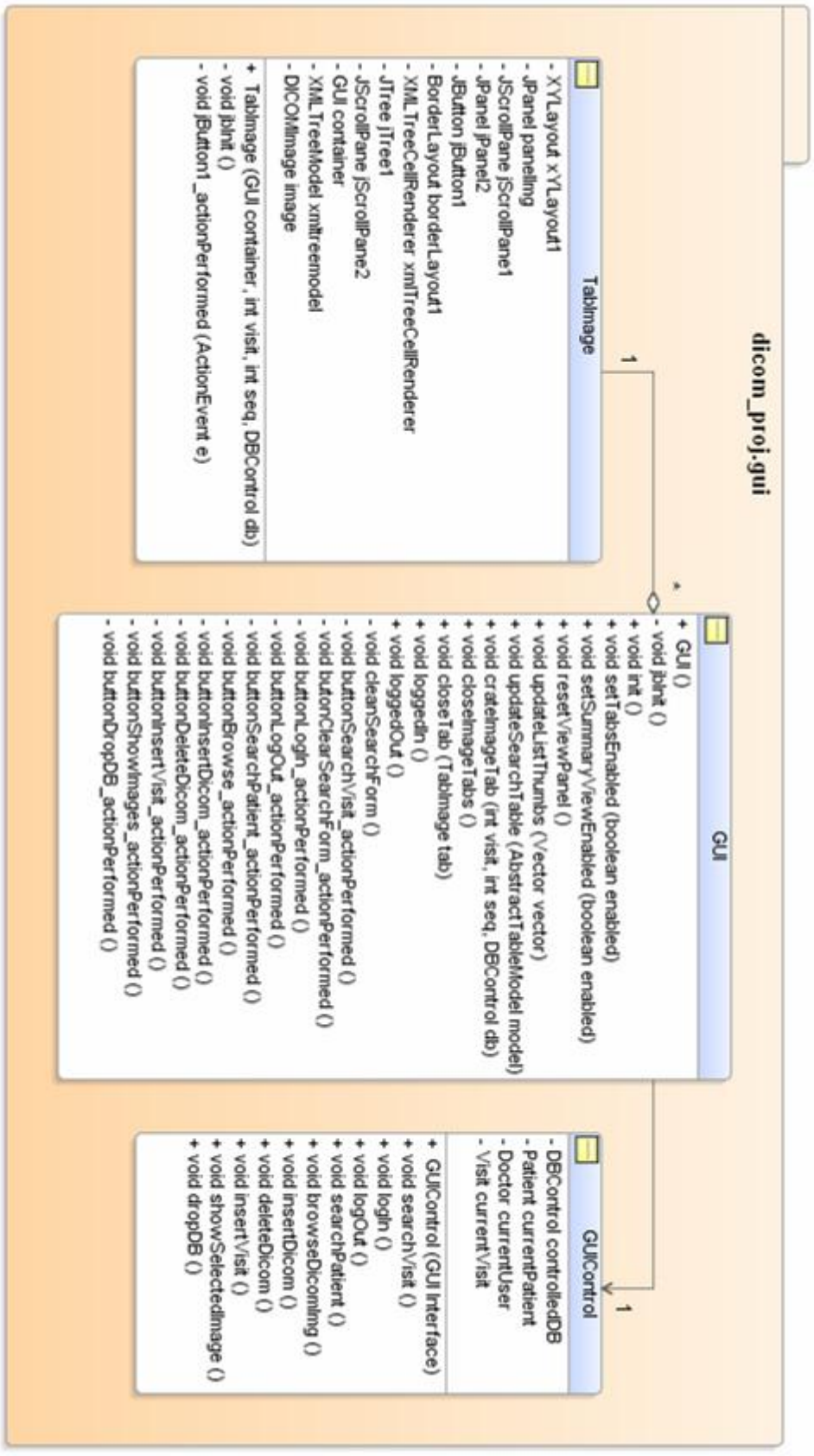


Figure 7.17: Package view - dicom_proj.gui

c) dicom_proj.model

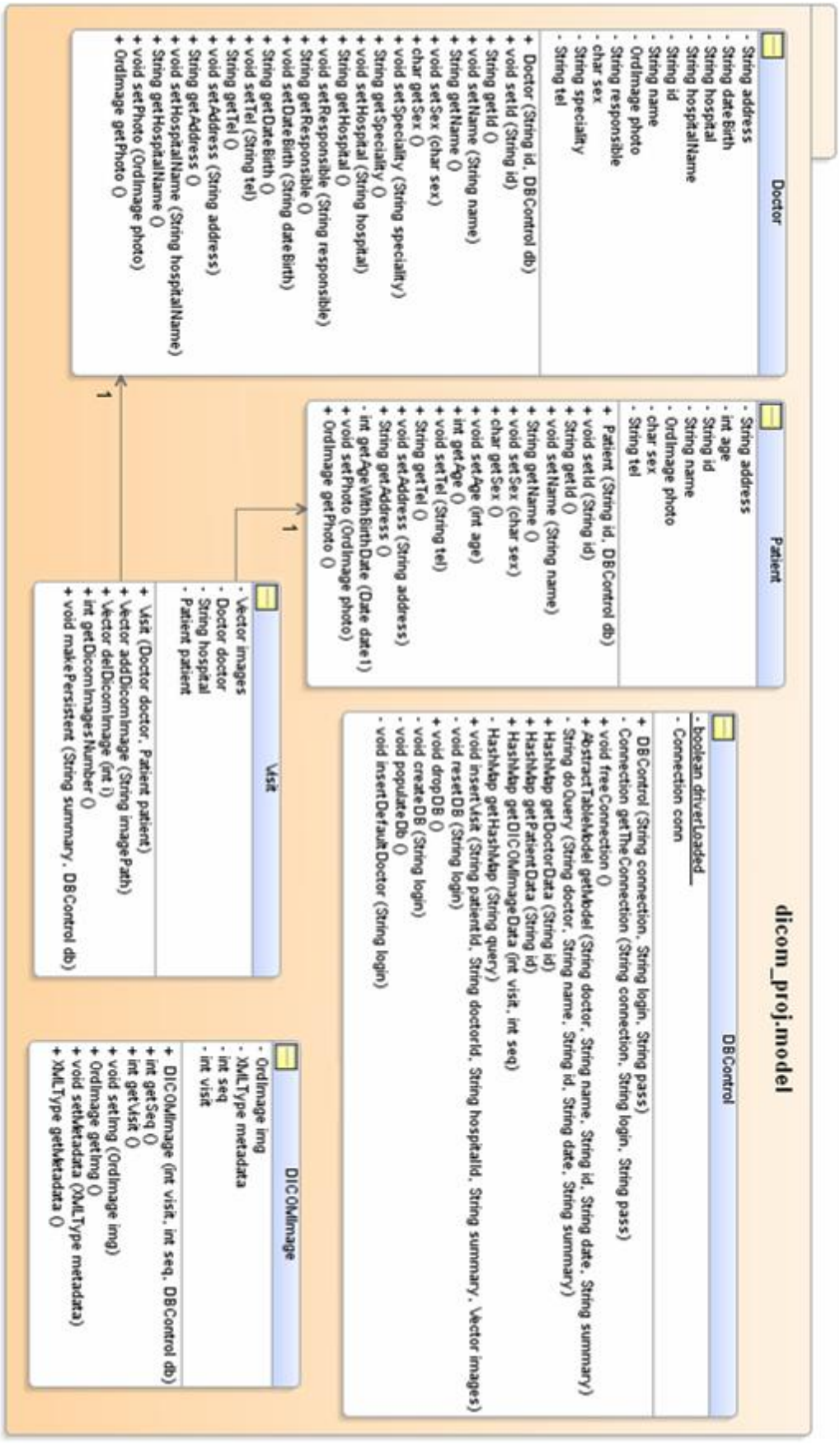


Figure 7.18: Package view - dicom_proj.model

In the previous Figure 7.18 is clear the model schema. The classes “Patient”, “Doctor” and “Visit” are quite a lot similar to their respective database tables. Each patient, doctor or DICOMImage instance is created receiving the id and the unique instance of the “DBControl” class for retrieving the value of their attributes. For each attribute they have the necessary set and get methods. In particular the “DBControl” class has all the needed methods for managing the connection and the data retrieval or update in the physical database. Moreover for testing issues contains methods for allowing dropping or creating all the tables involved in the application

d) dicom_proj.util

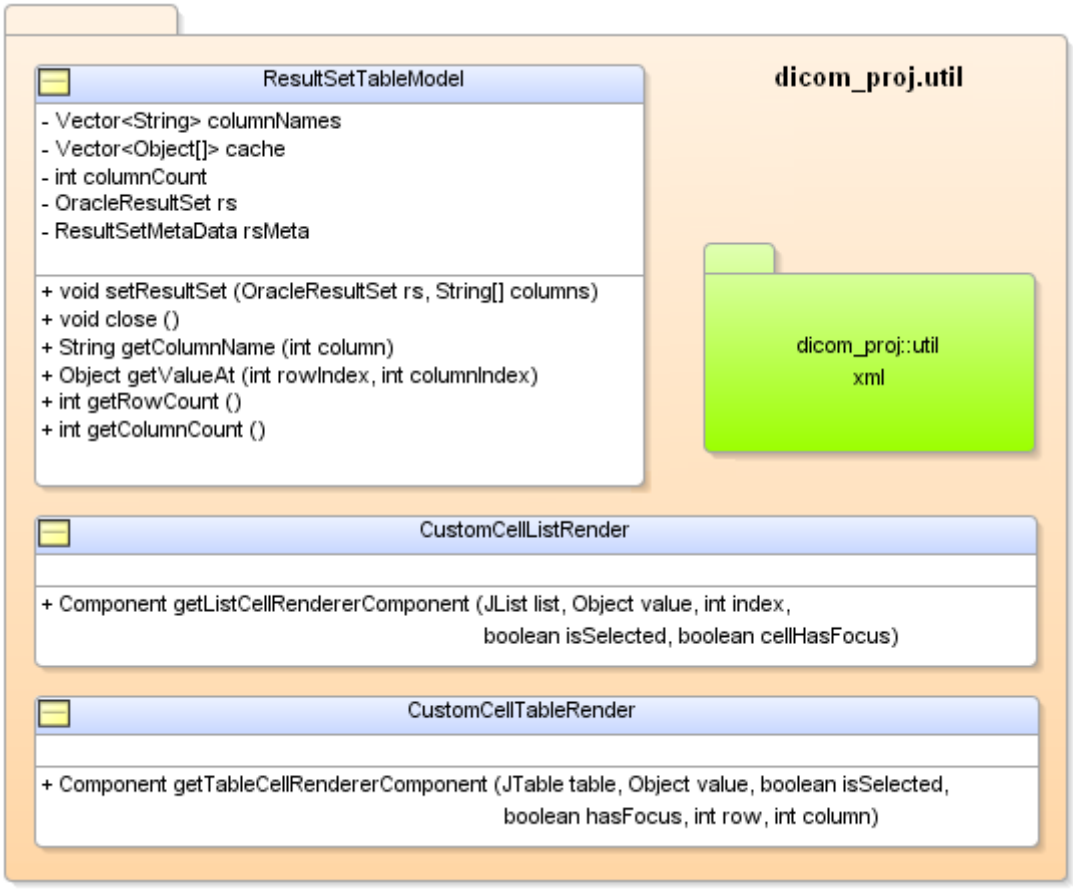


Figure 7.19: Package view - dicom_proj.util

This package contains some utility classes with no relevance for the problem. In two of them we redefine the standard render method for cells in a list and in a table for been able of showing thumbnails from images. The other class is a model for populating a table (in the graphic interface) from a result set, very useful for the task of showing a table with the recovered data from a query.

It contains also a specialized package in xml data.

e) dicom_proj.util.xml

The classes contained are used for showing in the graphic interface a Jtree with the data of a XMLType retrieved from the database. We should remember that each DICOM file has a related metadata inside, which is stored into the database and the application shows it by the side of the DICOM image in every “ImageTab”.

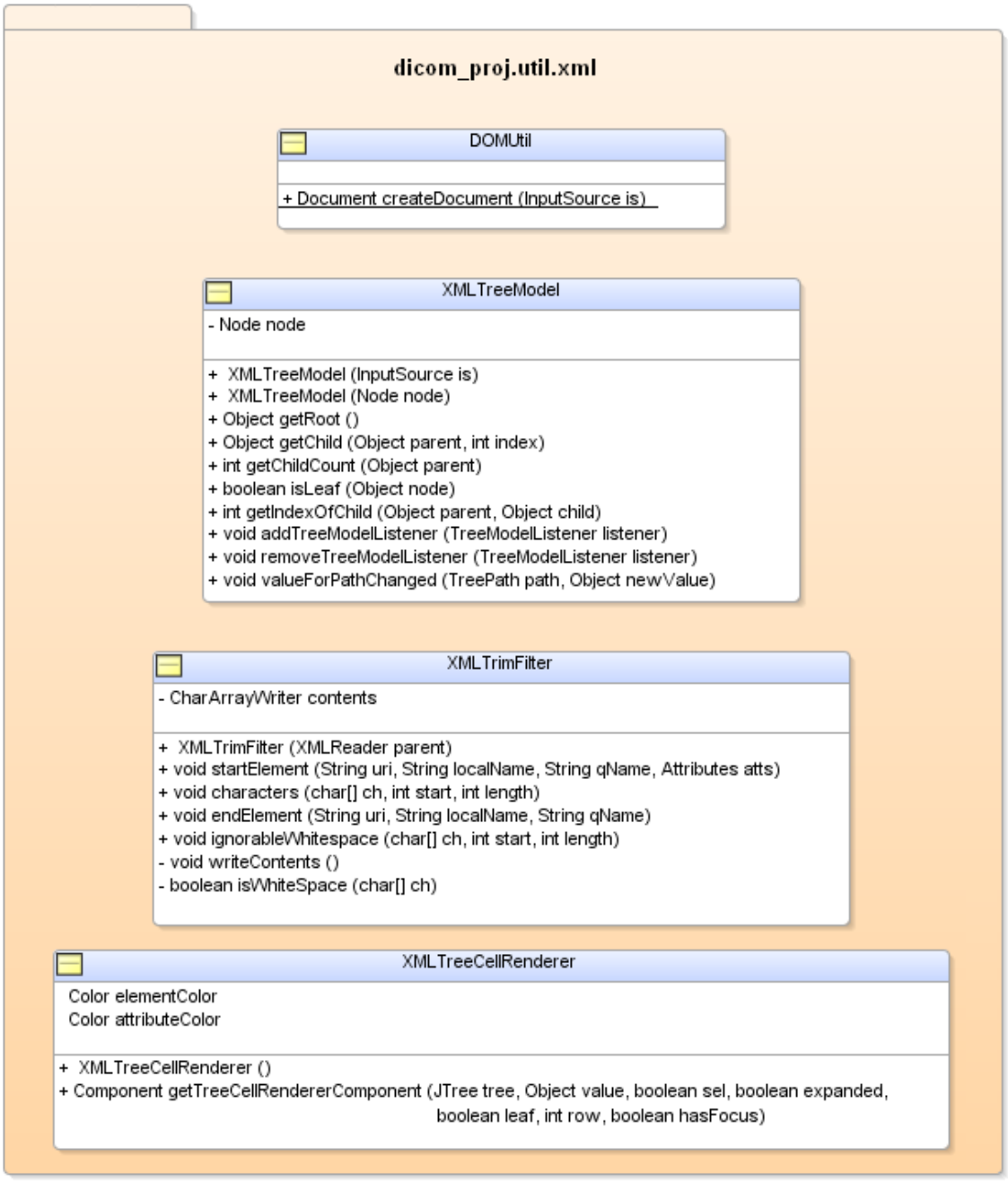


Figure 7.20: Package view - dicom_proj.util.xml

7.3. Implementation

After all the process concerning to the analysis and design stages, we reach the implementation. As we mentioned before, the chosen language was Java because JDBC (Java Database Connectivity) provides a good API for working with Oracle Databases and this way we can use the Oracle interMedia library for working with the DICOM images under Java. Other reason for choosing Java was because provides the Swing library for creating an attractive and easy graphic user interface.

Then, the second decision about the implementation was the development environment. JDeveloper was the chosen IDE (Integrated Develop Enviroment) and the stronger reasons are because is provided free by Oracle, manages easily a lot of aspects of the database and at the same time has all the features of a Java developer environment (GUI design assistant, debug capability, navigability between classes, code help,...). With this IDE, the programmer can also create all the necessary diagrams from the documentation of a project, following the UML standard.

The version of the used JDeveloper was "Studio Edition Version 10.1.3.2.0" but configured using the last client version of Oracle "Oracle Database 10*g* Release 2 (10.2.0.1.0)" because in the previous there isn't exist the DICOM functions in the interMedia package.

Chapter 8. Conclusions

In the final part we will review the steps followed for covering all the purposed goals in the specification of this Master's Thesis and remark the successes reached in each stage.

First was made a wide study about the theoretical background including concepts about multimedia databases, the more spread medical standard for images, called DICOM, and the link between both of them, in this case Oracle interMedia 10g Release 2 (June 2005) because the previous versions didn't include support for DICOM images. One of the most important characteristic of DICOM images is that the same file is container for the image and all the metadata related to the capture. That is why in the theoretical background was included other chapter studying the metadata and a schema example.

After that, there were included some technical examples about how to work with the DICOM files using the library Oracle interMedia and Java as language.

In the last place, an example application was developed for leading the previous concepts studied to the practice. This application was object of a software engineer process beginning with an analysis, a design and later an implementation using Java as language and assisted by the Oracle libraries for managing multimedia data, called interMedia. This application interacts with an Oracle database for storing and retrieving the necessary data. Working with the sample application brings us the possibility of probe that Oracle interMedia and JDBC provide a good API to work and manipulate images in a database

Moreover, the most important success of this application is that it isn't a simple example; it's may be a useful and intuitive application which can help the doctors in hospitals for keeping a register of visits. These visits can include a summary and the related DICOM images for each visit. It makes possible the queries over the previous visits including the visualization of the medical images and the associated metadata.

Other important points about this application lies in that is based on the last version of the Oracle client (from Oracle Database 10*g* Client Release 2 10.2.0.1.0 finished on June 2005). This means that is a modern technology which use is increasing every year.

8.1. Future Work

The future development of the implemented program mostly depends on the requirements of doctors and other users involved in the hospital tasks, like radiologists.

Some of these improvements could be:

- **Searching by metadata:** It could be interesting to perform searching over the metadata of the DICOM images instead of the visit or the patient data only. This way a doctor can search, for example, all the images that where taken the same day or a particular radiologic machine.
- **Similarity search:** This way a doctor can compare images from the same part of the body between different patients. It could be very useful making studies about a particular disease or comparing the results between different treatments, etc.
- **Security issues:** Actually the application declines all the responsibilities about security and let it to the database. It should be implemented a better way for distinguish the allowed accesses or for having a control of users, login register, etc. in the future.
- **New interfaces:** Maybe it will be necessary to access this application from other platforms different than a laptop like a PDA or other mobile devices.

Besides, the proposed application is being distributed under GPL open source license. It is documented with the correspondent diagrams during the engineer process, so it can be reused or modified by the other developers.

References

[1] ORACLE Corporation. *Oracle interMedia User's Guide, 10g Release 2 (10.2)* [on line]. Sue Pelski. [Redwood City, USA] June 2005. Available at: <http://download-uk.oracle.com/docs/cd/B19306_01/appdev.102/b14302.pdf> [Last seen: 16th of May 2007]

[2] ORACLE Corporation. *Oracle Database 10g Release 2 DICOM Medical Image Support* [on line]. Bill Gettys [Redwood Shores, USA] September 2005. Available at: <http://www.oracle.com/technology/products/intermedia/pdf/dicom_technical_wp.pdf> [Last seen: 19th of May 2007]

[3] FDEZ.-DÍVAR, Ignacio. *Image Data Management*. Supervisor: Ing. Petr Chmelař. Brno University of Technology, Faculty of Information Technology, 2006. p.67

[4] THURAISINGHAM, Bhavani. *Managing and Mining Multimedia Databases*. 1st ed. USA: CRC Press LLC, 2001. p.11-79 ISBN: 0-8493-0037-1

[5] DJERABA, Chabane. *Multimedia Mining. A Highway to Intelligent Multimedia Documents.* 1st ed. USA: Kluwer Academic Publishers, 2003. p.139-158 ISBN: 1-4020-7247-3

[6] SUBRAHMANIAN, V.S. *Principles of Multimedia Database Systems.* 1st ed. USA: Morgan Kaufmann Publishers, Inc., 1998. p.19-63 ISBN: 1-55860-466-9

[7] BOOCH, Grady. JACOBSON, Ivar. RUMBAUGH, James. *UML 2 and the Unified Process: practical object-oriented analysis and design.* 2nd ed. USA: Addison Wesley, 2005. p.569 ISBN: 0-321-32127-8

[8] RSNA, Radiological Society Of North America, Inc. *A Non Technical Introduction to DICOM* [on line]. [Oak Brook, USA] 2007. Available at: <http://www.rsna.org/ Technology/DICOM/intro/> [Last seen: 31st of May 2007].

[9] ANALYSER Sales Ltd. *The DICOM Standard* [on line] Steven C. Horiil, Fred W. Prior, W. Dean Bidgood, Jr., Charles Parisot, Geert Caléis [Bashurst Hill Slinfold, UK], 2003. Available at: <http://www.dicomanalyser.co.uk/html/introduction.htm> [Last seen: 31st of May 2007]

[10] NEMA, National Electrical Manufacturers Association, *Digital Imaging and Communications in Medicine (DICOM)* [on line] [Rosslyn, Virginia, USA] Available at: <http://medical.nema.org/dicom/2007> [Last seen 29th of May 2007]

[11] RORDEN, Chris. Associated Professor in University of South Carolina. *The DICOM Standard* [on line] [Columbia, USA] Available at: <http://www.sph.sc.edu/comd /rorden/dicom.html> [Last seen: 30th of May 2007]

[12] BARRY & ASSOCIATES, Inc. *Transparent persistence in object-relational mapping* [on line] Available at: <http://www.service-architecture.com/object-relational-mapping/articles/transparent_persistence.html> [Last seen: 4th of Jun 2007]

[13] GIRALDA RODRÍGUEZ, Alejandro. *Image Databases Indexing*. Supervisor: Ing. Petr Chmelař. Brno University of Technology, Faculty of Information Technology, 2006. p.92

[14] GUDIVADA, V.N.; RAGHAVAN, V.V. *Content based image retrieval systems* [on line] [Ohio Univ, USA] Available at: < http://ieeexplore.ieee.org/iel1/2/9181/00410145 .pdf> [Last seen: 4th of Jun 2007]

[15] JEONG, Sangoh. *Histogram – Based Color image retrieval* [on line] Available at: <http://scien.stanford.edu/class/psych221/projects/02/sojeong/> [Last seen: 6th of Jun 2007]

[16] NISO, National Information Standards Organization. *Understanding Metadata* [on line][Bethesda, USA] Available at: <http://www.niso.org/standards/resources/UnderstandingMetadata.pdf> [Last seen: 7th of Jun 2007]

[17] ODL, Oxford Digital Library. *Metadata in the Oxford Digital Library*. [on line] [Oxford, UK] Available at:<http://www.odl.ox.ac.uk/metadata.htm> [Last seen: 4th of Jun 2007]

Chapter 9. Appendixes & Attachments

9.1. CD Content

- Documentation – PDF and MS Word formats.
- Sources – Application codes and JDeveloper complete project.
- Includes – Necessary prerequisites.

9.2. DICOM Standard 2007

As we said in the section 4.4, the new specification of 2007 includes a total of sixteen parts:

- PS 3.1: Introduction and Overview (this document)
- PS 3.2: Conformance
- PS 3.3: Information Object Definitions
- PS 3.4: Service Class Specifications
- PS 3.5: Data Structure and Encoding
- PS 3.6: Data Dictionary
- PS 3.7: Message Exchange
- PS 3.8: Network Communication Support for Message Exchange
- PS 3.9: Retired
- PS 3.10: Media Storage and File Format for Data Interchange
- PS 3.11: Media Storage Application Profiles
- PS 3.12: Media Formats and Physical Media for Data Interchange
- PS 3.13: Retired
- PS 3.14: Grayscale Standard Display Function
- PS 3.15: Security Profiles
- PS 3.16: Content Mapping Resource

These parts of the Standard are related but independent documents. A brief description of each Part is provided in this section.

PS 3.2: Introduction and Overview

Provides an overview of the entire Digital Imaging and Communications in Medicine (DICOM) Standard.

PS 3.2: Conformance

Defines the principles that implementations claiming conformance to the DICOM Standard shall follow:

- Conformance requirements. PS 3.2 specifies the general requirements which must be met by any implementation claiming conformance. It references the conformance sections of other parts of the Standard.

- Conformance Statement. PS 3.2 defines the structure of a Conformance Statement. It specifies the information which must be present in a Conformance Statement. It references the Conformance Statement sections of other parts of the Standard.

Does not specify a testing/validation procedure to assess an implementation's conformance to the Standard. A Conformance Statement consists of the following parts:

- Set of Information Objects which is recognized by this implementation
- Set of Service Classes which this implementation supports
- Set of communications protocols or physical media which this implementation supports
- Set of security measures which this implementation supports

PS 3.3: Information Object Definitions

Specifies a number of Information Object Classes which provide an abstract definition of real-world entities applicable to communication of digital medical images and related information (e.g., waveforms, structured reports, radiation therapy dose, etc.). Each Information Object Class definition consists of a description of its purpose and the Attributes which define it. An Information Object Class does not include the values for the Attributes which comprise its definition.

Defines a model of the Real World along with the corresponding Information Model that is reflected in the Information Object Definitions. Future editions of this Standard may extend this set of Information Objects to support new functionality.

PS 3.4: Service Class Specifications

Defines a number of Service Classes. A Service Class associates one or more Information Objects with one or more Commands to be performed upon these objects. Service Class Specifications state requirements for Command Elements and how resulting Commands are applied to Information Objects. Service Class Specifications state requirements for both providers and users of communications services.

PS 3.4 defines the characteristics shared by all Service Classes, and how a Conformance Statement to an individual Service Class is structured. It contains a number of normative annexes which describe individual Service Classes in detail. Examples of Service Classes include the following:

- Storage Service Class
- Query/Retrieve Service Class
- Basic Worklist Management Service Class
- Print Management Service Class.

PS 3.4 defines the operations performed upon the Information Objects defined in PS 3.3. PS 3.7 defines the Commands and protocols for using the Commands to accomplish the operations and notifications described in PS 3.4.

PS 3.5: Data Structure and Encoding

Specifies how DICOM applications construct and encode the Data Set information resulting from the use of the Information Objects and Services Classes defined in PS 3.3 and PS 3.4 of the DICOM Standard. The support of a number of standard image compression techniques (e.g., JPEG lossless and lossy) is specified.

PS 3.5 addresses the encoding rules necessary to construct a Data Stream to be conveyed in a Message as specified in PS 3.7 of the DICOM Standard. This Data Stream is produced from the collection of Data Elements making up the Data Set.

Also defines the semantics of a number of generic functions that are common to many Information Objects. PS 3.5 defines the encoding rules for international character sets used within DICOM.

PS 3.6: Data Dictionary

Is the centralized registry which defines the collection of all DICOM Data Elements available to represent information, along with elements utilized for interchangeable media encoding and a list of uniquely identified items that are assigned by DICOM.

For each element, PS 3.6 specifies:

- Its unique tag, which consists of a group and element number
- Its name
- Its value representation (character string, integer, etc)
- Its value multiplicity (how many values per attribute)
- Whether it is retired

For each uniquely identified item, PS 3.6 specifies:

- Its unique value, which is numeric with multiple components separated by decimal points and limited to 64 characters
- Its name
- Its type, either Information Object Class, definition of encoding for data transfer, or certain well known Information Object Instances
- In which part of the DICOM Standard it is defined

PS 3.7: Message Exchange

Specifies both the service and protocol used by an application in a medical imaging environment to exchange Messages over the communications support services defined in PS 3.8. A Message is composed of a Command Stream defined in PS 3.7 followed by an optional Data Stream as defined in PS 3.5.

PS 3.8: Network Communication Support for Message Exchange

Specifies the communication services and the upper layer protocols necessary to support, in a networked environment, communication between DICOM applications as specified in PS 3.3, PS 3.4, PS 3.5, PS 3.6, and PS 3.7. These communication services and protocols ensure that communication between DICOM applications is performed in an efficient and coordinated manner across the network.

PS 3.9: Retired

It has been retired. Previously specified the services and protocols used for point-to-point communications in a manner compatible with ACR-NEMA 2.0.

PS 3.10: Media Storage and File Format for Data Interchange

Specifies a general model for the storage of medical imaging information on removable media (see Figure 9.1). The purpose of this Part is to provide a framework allowing the interchange of various types of medical images and related information on a broad range of physical storage media.

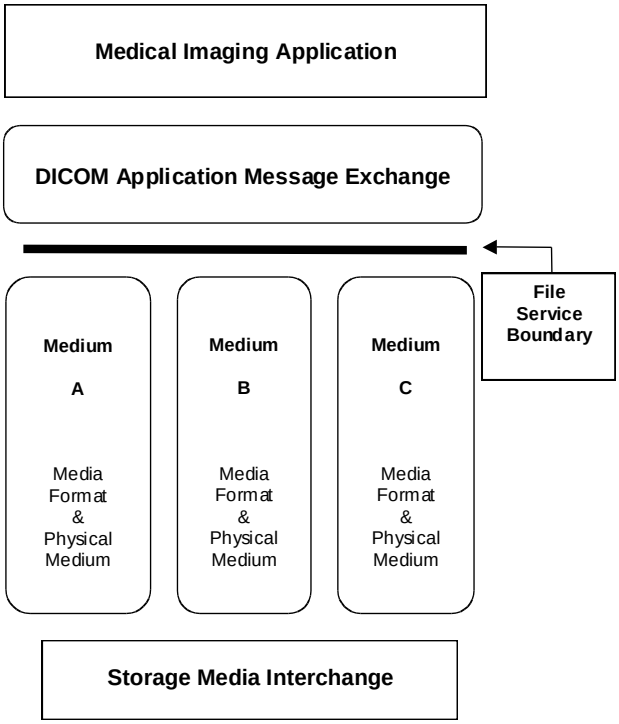


Figure 9.1: DICOM Media Communication Model [10]

See Figure 4.1 (pag. 22) for understanding how the media interchange model compares to the network model.

PS 3.11: Media Storage Application Profiles

Specifies application specific subsets of the DICOM Standard to which an implementation may claim conformance. These application specific subsets will be referred to as Application Profiles in this section. Such a conformance statement applies to the interoperable interchange of medical images and related information on storage media for specific clinical uses. It follows the framework, defined in PS 3.10, for the interchange of various types of information on storage media.

PS 3.12: Media Formats and Physical Media for Data Interchange

This part of the DICOM Standard facilitates the interchange of information between applications in medical environments by specifying:

- A structure for describing the relationship between the media storage model and a specific physical media and media format.
- Specific physical media characteristics and associated media formats.

PS 3.13: Retired

It has been retired. Previously specified the services and protocols used for point-to-point communication of print management services.

PS 3.14: Grayscale Standard Display Function

Specifies a standardized display function for consistent display of grayscale images. This function provides methods for calibrating a particular display system for the purpose of presenting images consistently on different display media (e.g. monitors and printers).

PS 3.15: Security Profiles

Specifies security and system management profiles to which implementations may claim conformance. Security and system management profiles are defined by referencing externally developed standard protocols, such as DHCP, LDAP, TLS and ISCL. Security protocols may use security techniques like public keys and “smart cards”. Data encryption can use various standardized data encryption schemes.

This part does not address issues of security policies. The standard only provides mechanisms that can be used to implement security policies with regard to the interchange of DICOM objects. It is the local administrator’s responsibility to establish appropriate security policies.

PS 3.16: Content Mapping Resource

PS 3.16 of the DICOM Standard specifies:

- Templates for structuring documents as DICOM Information Objects
- Sets of coded terms for use in Information Objects
- A lexicon of terms defined and maintained by DICOM
- Country specific translations of coded terms

9.3. interMedia - DICOM encoding rules

Digital Imaging and Communications in Medicine (DICOM) includes a complete set of encoding rules for medical images. These encoding rules are also called transfer syntax. Oracle interMedia provides DICOM encoding rules that support metadata extraction and image content processing.

The following Figure 9.2 lists the DICOM encoding rules supported by Oracle interMedia.

The following abbreviations are used:

- **M** = Metadata extraction support is provided for the `getDicomMetadata()` and `setProperties()` methods.
- **I** = Image content support is provided for the `processCopy()` and `setProperties()` methods.
- **N** = Not supported by interMedia.

Value	Name	interMedia Support
1.2.840.10008.1.2	Implicit VR Little Endian Default Transfer Syntax	MI
1.2.840.10008.1.2.1	Explicit VR Little Endian Transfer Syntax	MI
1.2.840.10008.1.2.1.99	Deflated Explicit VR Little Endian	N
1.2.840.10008.1.2.2	Explicit VR Big Endian	N
1.2.840.10008.1.2.4.50	JPEG Baseline (Process 1)	MI
1.2.840.10008.1.2.4.51	JPEG Extended (Process 2 & 4)	M
1.2.840.10008.1.2.4.52	JPEG Extended (Process 3 & 5) (Retired)	M
1.2.840.10008.1.2.4.53	JPEG Spectral Selection, Non-Hierarchical (Process 6 & 8) (Retired)	M
1.2.840.10008.1.2.4.54	JPEG Spectral Selection, Non-Hierarchical (Process 7 & 9) (Retired)	M
1.2.840.10008.1.2.4.55	JPEG Full Progression, Non-Hierarchical (Process 10 & 12) (Retired)	M
1.2.840.10008.1.2.4.56	JPEG Full Progression, Non-Hierarchical (Process 11 & 13) (Retired)	M
1.2.840.10008.1.2.4.57	JPEG Lossless, Non-Hierarchical (Process 14)	M
.2.840.10008.1.2.4.58	JPEG Lossless, Non-Hierarchical (Process 15) (Retired)	M
1.2.840.10008.1.2.4.59	JPEG Extended, Hierarchical (Process 16 & 18) (Retired)	M

Value	Name	interMedia Support
1.2.840.10008.1.2.4.60	PEG Extended, Hierarchical (Process 17 & 19) (Retired)	M
1.2.840.10008.1.2.4.61	JPEG Spectral Selection, Hierarchical (Process 20 & 22) (Retired)	M
1.2.840.10008.1.2.4.62	JPEG Spectral Selection, Hierarchical (Process 21 & 23) (Retired)	M
1.2.840.10008.1.2.4.63	JPEG Full Progression, Hierarchical (Process 24 & 26) (Retired)	M
1.2.840.10008.1.2.4.64	JPEG Full Progression, Hierarchical (Process 25 & 27) (Retired)	M
1.2.840.10008.1.2.4.65	JPEG Lossless, Hierarchical (Process 28) (Retired)	M
1.2.840.10008.1.2.4.66	JPEG Lossless, Hierarchical (Process 29) (Retired)	M
1.2.840.10008.1.2.4.70	JPEG Lossless, Non-Hierarchical, First-Order Prediction (Process 14 [Selection Value 1])	M
1.2.840.10008.1.2.4.80	JPEG-LS Lossless Image Compression	M
1.2.840.10008.1.2.4.81	JPEG-LS Lossy (Near-Lossless) Image Compression	M
1.2.840.10008.1.2.4.90	JPEG 2000 Image Compression (Lossless Only)	M
1.2.840.10008.1.2.4.91	JPEG 2000 Image Compression	M
1.2.840.10008.1.2.4.100	MPEG2 Main Profile @ Main Level	M
1.2.840.10008.1.2.5	RLE Lossless	MI

Figure 9.2: DICOM encondig rules supported by interMedia [1]

9.4. XML Schema for DICOM Metadata

This schema is the content model for Digital Imaging and Communications in Medicine (DICOM) metadata retrieved from images. The namespace for this schema is `<http://xmlns.oracle.com/ord/meta/dicomImage>`.

This schema defines the set of DICOM metadata published by Oracle interMedia ORDImage function `getDicomMetadata()`.

The purpose of this set of functions is to extract from a DICOM image a set of attributes to describe the image in XML, allowing easy browsing and retrieval.

```
DICOM_IMAGE

        ORD_DICOM_HEADER
                VERSION
                DICOM_STANDARD_VERSION
                DICOM_STANDARD_RELEASE

        FILE_META_HEADER
                MEDIA_STORAGE_SOP_CLASS_UID
                MEDIA_STORAGE_SOP_INSTANCE_UID
                TRANSFER_SYNTAX_UID
                IMPLEMENTATION_CLASS_UID
                IMPLEMENTATION_VERSION_NAME
                SOURCE_APPLICATION_ENTITY_TITLE

        PATIENT
                NAME
                ID
                BIRTH_DATE
                SEX

        GENERAL_STUDY
                INSTANCE_UID
                DATE
                TIME
                REFERING_PHYSICIANS_NAME
                ID
                ACCESSION_NUMBER
                DESCRIPTION

        PATIENT_STUDY
                ADMITTING_DIAGNOSES_DESCRIPTION
                ADMITTING_DIAGNOSES_CODE_SEQUENCE
```

```
GENERAL_SERIES
       MODALITY
       INSTANCE_UID
       DATE
       TIME
       PERFORMING_PHYSICIANS_NAME
       BODY_PART_EXAMINED
       PATIENT_POSITION
       PERFORMED_PROCEDURE_STEP_ID
       PERFORMED_PROCEDURE_STEP_START_DATE
       PERFORMED_PROCEDURE_STEP_START_TIME
       PERFORMED_PROCEDURE_STEP_DESCRIPTION
       PERFORMED_PROTOCOL_CODE_SEQUENCE

GENERAL_EQUIPMENT
       MANUFACTURER

GENERAL_IMAGE
       INSTANCE_NUMBER
       ACQUISITION_NUMBER
       ACQUISITION_DATE
       ACQUISITION_TIME
       ACQUISITION_DATETIME
       PATIENT_ORIENTATION
       FRAME_LATERALITY
       ANATOMIC_REGION

IMAGE_PIXEL
       SAMPLES_PER_PIXEL
       PHOTOMETRIC_INTERPRETATION
       ROWS
       COLUMNS
       BIT_ALLOCATED
       BIT_STORED
       HIGH_BIT
       PIXEL_REPRESENTATION
              PLANAR_CONFIGURATION
       PIXEL_ASPECT_RATIO

SOP_COMMON
       CLASS_UID
       INSTANCE_UID
       SPECIFIC_CHARACTER_SET
```

NOTE: In *italics* means optional items

9.5. User Manual

9.5.1. DB Login

This Figure 9.3 is the first appearance of the application when a user run it. We are looking at the "DB Login" tab and here the doctor can Log In and Log Out. For testing issues we have also a button for dropping all the tables related to the application. If the doctor is not logged in the other tabs and options are disabled as we can see beneath.

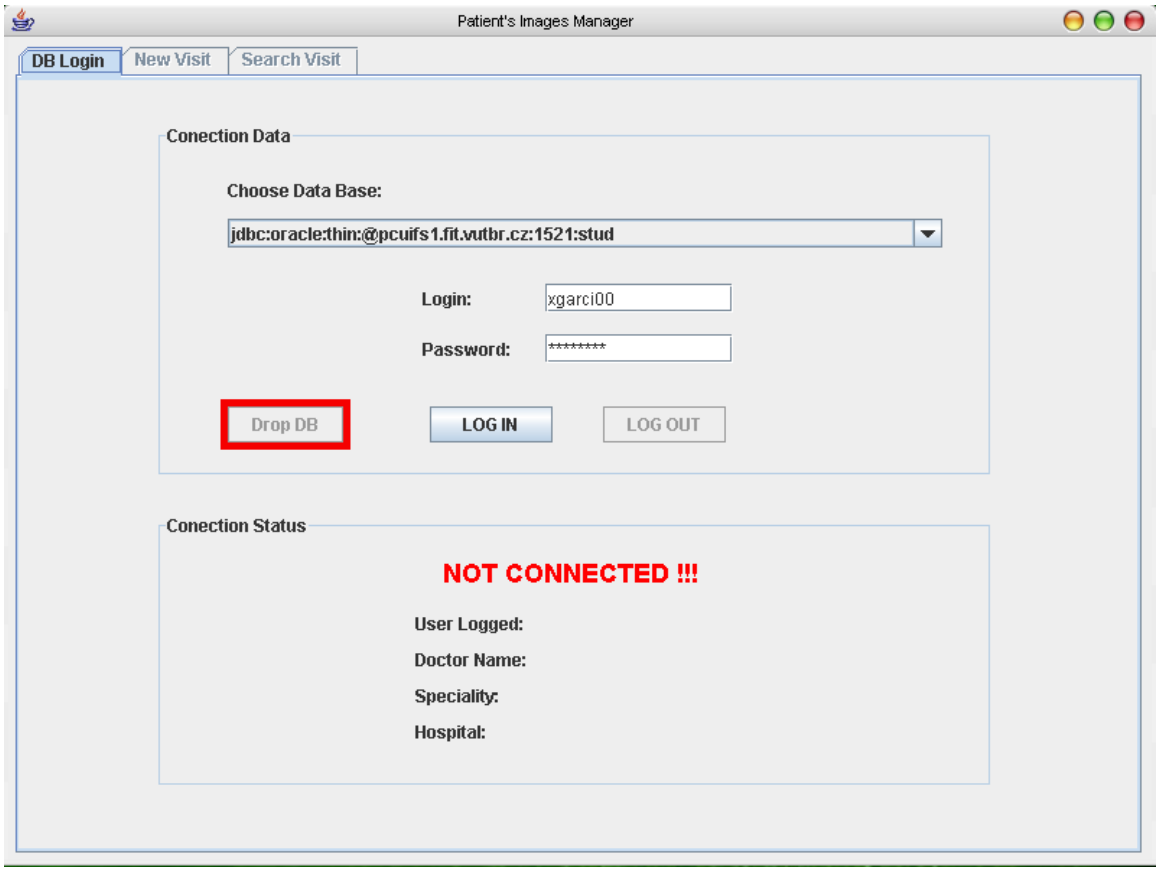


Figure 9.3: Manual - DB Login tab (1)

The program can be configured for showing a list of different databases. The user only have to chose one from the list and to fill the login and password fields. Later, by pressing the "LOG IN" button, if the doctor is allowed in the database the application looks like in the following Figure 9.4.

The application shows the data related to the doctor logged in and enables the other tabs and the "LOG OUT" and "Drop DB" buttons. For log out or dropping the database the doctor only has to press the correspondent button. When the doctor logs out all the tabs and options previously enabled are disabled again and the doctor information is cleared.

At the moment of logging in can happen that the login or password is not valid or that the program couldn't establish a connection with the desired database. This situations are detected by the application and warned to the user with the correspondent message: "LOGIN/PASS ERROR" or "DB CONNECTION ERROR".

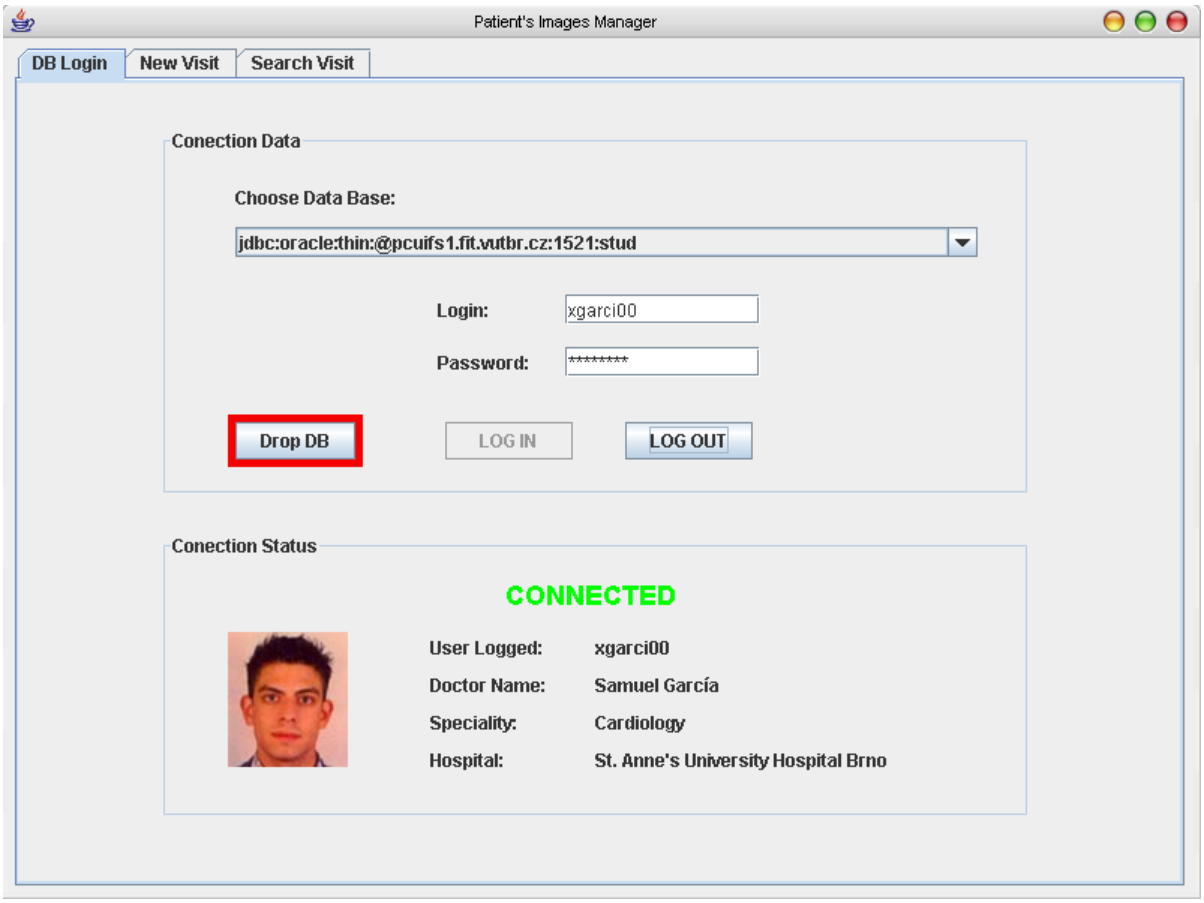


Figure 9.4: Manual - DB Login tab (2)

9.5.2. New Visit

The next tab is the "New Visit". Each time a patient goes to the doctor, this can add a new visit to the patient data. The visit can contain a summary and DICOM files. The first appearance of this tab is with all the options disabled, except the patient ID search, waiting for knowing the related patient to the visit. A "PATIENT DATA NOT LOADED" is shown.

If the doctor inserts a valid id and press the search button, the associated data of the patient is shown and the options for adding the summary or the DICOM images are enabled as we can see in the Figure 9.5. If the id is not valid a message with the text "PATIENT ID NOT FOUND" is shown in the patient data panel.

Now the doctor can add an optional description of the problem in the text area of the visit summary panel. In addition, now is possible to make up the list of the DICOM files will be attached to this visit.

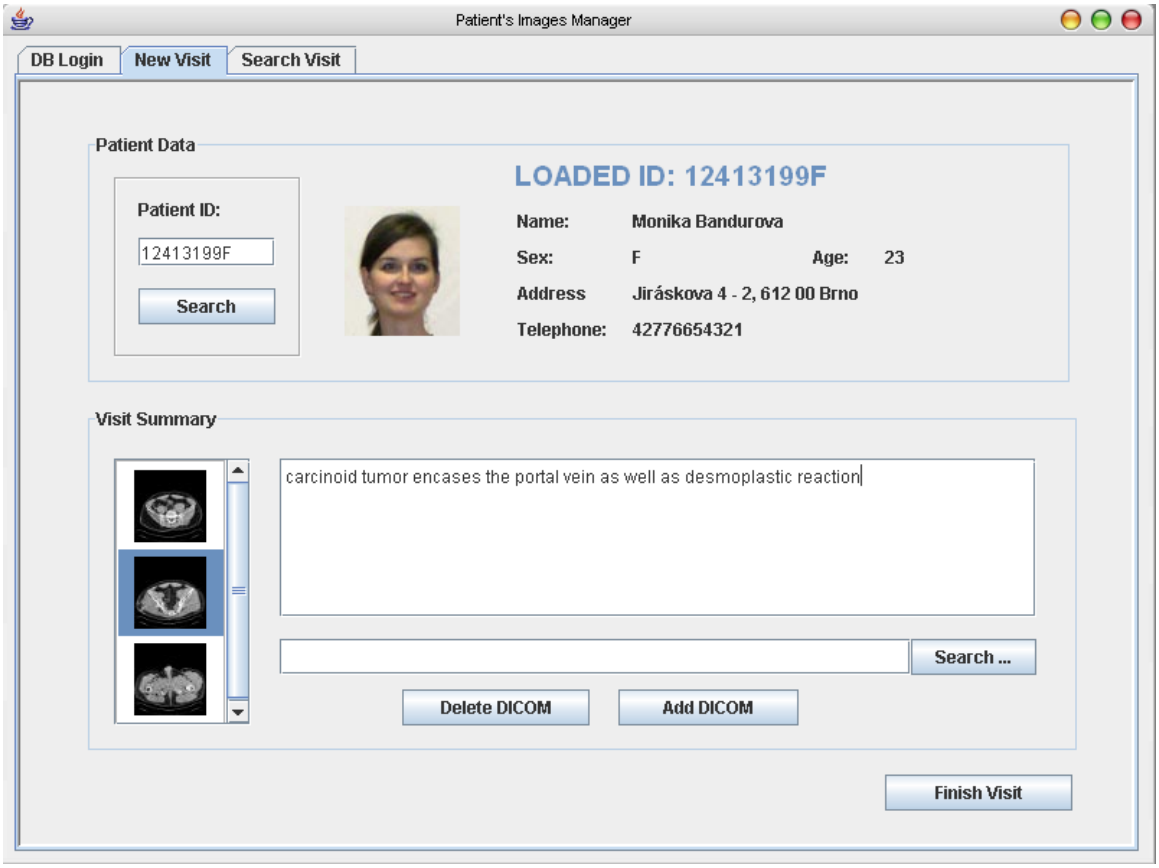


Figure 9.5: Manual - New Visit tab

For adding a new file, the doctor must press the "Search..." button in the Visit Summary panel and a file explorer window will appear. By choosing the desired file and pressing "OK" at this explorer window, the complete URL of the chosen file is showed at the correspondent field. Now, by pressing the "Add DICOM" button the file is added to the list on the left showing its thumbnail or url. During the process of making up the list, the doctor can select the elements from the list and remove it by pressing the "Delete DICOM" button. The thumbnail of the selected file will disappear.

When the doctor finish the visit and wants to make it permanent in the database, only has to press the "Finish Visit" button and the window will be cleared, the current patient unloaded and the visit summary panel disabled again waiting for a loaded patient.

For the new visit to be inserted into the database it should have at least some text in the summary or a file in the list but never a visit without summary and without files will be inserted cause it has no sense.

Once the new visit is inserted in the database, the doctor can search for it or for the previous visits for looking at the summaries, at the original size images or at the metadata trees related to each images.

9.5.3. Search Visit

As said before the doctor can access to all the data stored from the previous visits for looking at the summary, image, metadata, etc... For doing this, it should be possible to make queries combining different criteria like the patient id, part of the name, part of the summary or the visit date. This is possible in the "Search Visit" panel as is shown in the Figure 9.6.

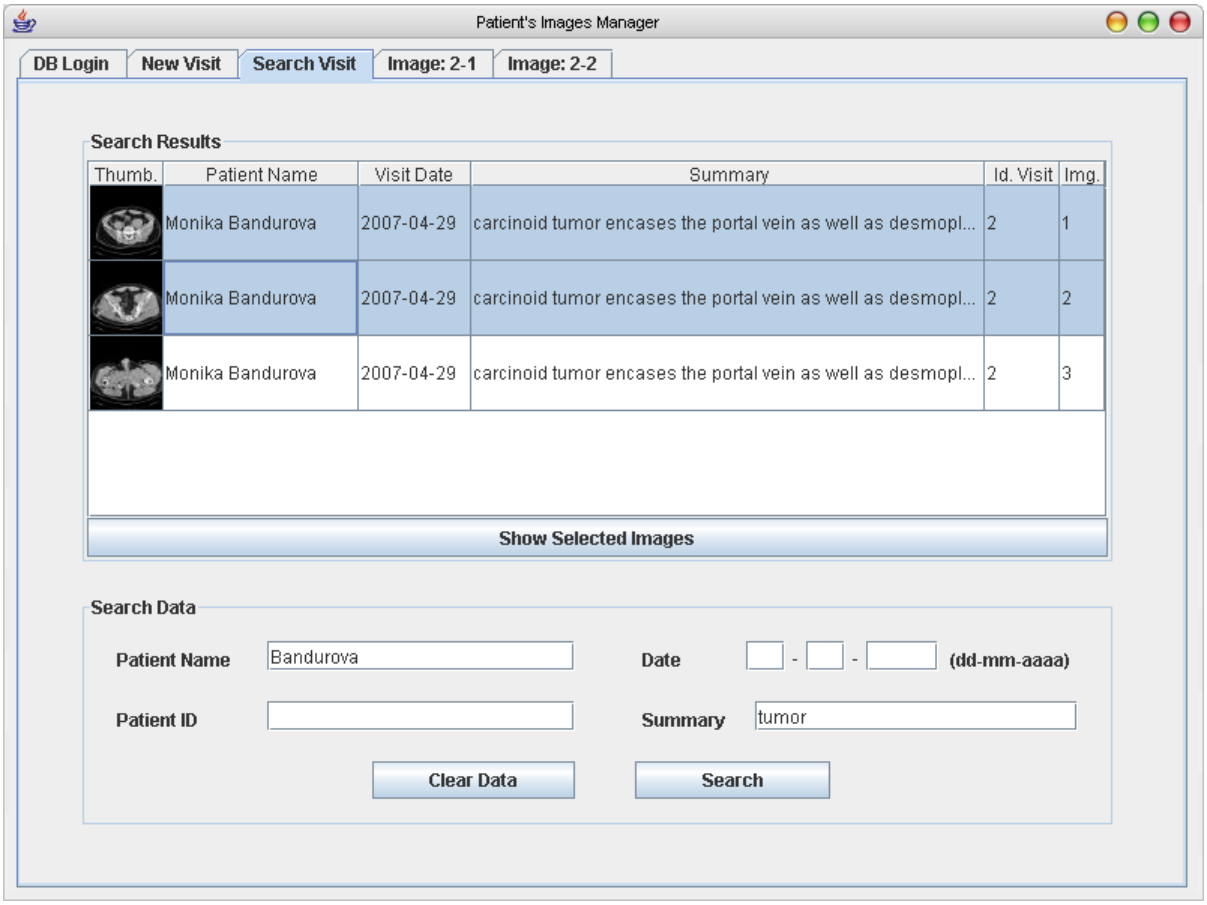


Figure 9.6: Manual - Search Visit tab

In the previous image we can see the result of making a query with a surname and a word from the summary.

9.5.4. View DICOM Image

The doctor can select from the table as many rows as he/she wants by pressing the "Ctrl" key for adding a new row, pressing "Shift" for defining a rank or combining both. This is the usual mechanism in the file explorers of the windows operative systems. Once the doctor has selected the images that wants to see detailed, by clicking over the "Show Selected Images" button, the application will open as many new tabs as selected files.

In every tab as shown in the Figure 9.7, we can see on the left the tree related to the metadata contained in the Dicom file and on the right the original size image. We can navigate through the nodes of the tree expanding or contracting the branches by pressing the left "o-" symbol.

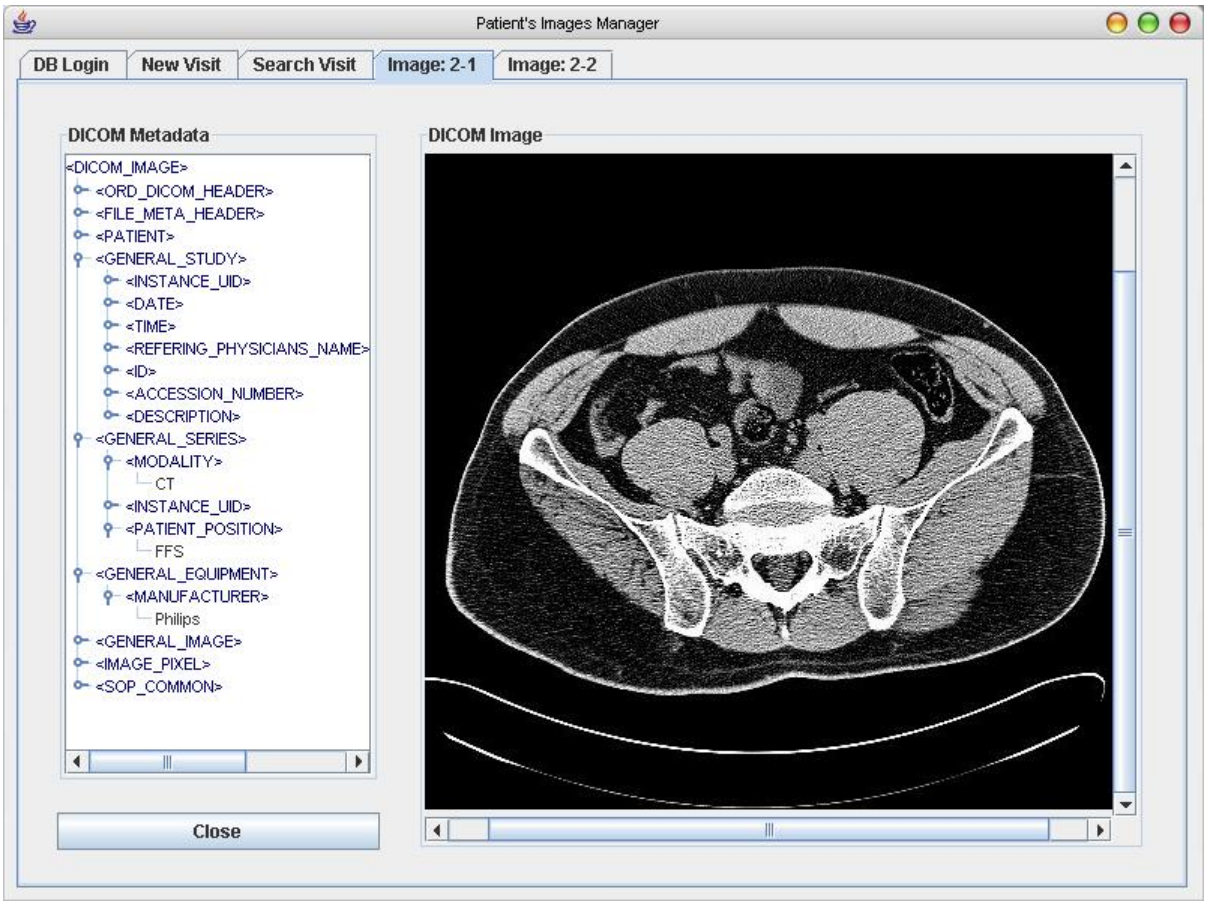


Figure 9.7: Manual - Image View tab

This example shows the result of the search in the Figure 9.6. We can see that there are two new tabs correspondent to the visit number 2 and its first two images (1 and 2). On this tree we can difference the structure of the DICOM metadata. In particular we can see in this figure the values for the modality of the image "CT" (Computed Tomography), the patient position or the manufacturer of the machine which made the capture.

The doctor can close each new tab by pressing the "Close" button in the left inferior area. When the doctor is logged out all the new tabs are closed too.

.