J Digit Imaging (2014) 27:297–308
DOI 10.1007/s10278-013-9665-9

Optimization of PACS Data Persistency Using Indexed Hierarchical Data

Thiago C. Prado · Douglas D. J. de Macedo · M. A. R. Dantas · Aldo von Wangenheim

Published online: 9 January 2014

Abstract We present a new approach for the development of a data persistency layer for a Digital Imaging and Communications in Medicine (DICOM)-compliant Picture Archiving and Communications Systems employing a hierarchical database. Our approach makes use of the HDF5 hierarchical data storage standard for scientific data and overcomes limitations of hierarchical databases employing inverted indexing for secondary key management and for efficient and flexible access to data through secondary keys. This inverted indexing is achieved through a general purpose document indexing tool called Lucene. This approach was implemented and tested using real-world data against a traditional solution employing a relational database, in various store, search, and retrieval experiments performed repeatedly with different sizes of DICOM datasets. Results show that our approach outperforms the traditional solution on most of the situations, being more than 600 % faster in some cases.

T. C. Prado · M. A. R. Dantas · A. von Wangenheim
Department of Informatics and Statistics, INE, Florianópolis, Brazil

T. C. Prado
e-mail: coelho@telemedicina.ufsc.br

M. A. R. Dantas
e-mail: mario@inf.ufsc.br

A. von Wangenheim
e-mail: awangenh@inf.ufsc.br

D. D. J. de Macedo (✉)
Post-Graduate Program in Knowledge Engineering and Management—PPGEGC, Federal University of Santa Catarina—UFSC, Florianópolis, Brazil
e-mail: macedo@inf.ufsc.br

D. D. J. de Macedo
Departamento de Informática e Estatística—Sala 320, Universidade Federal de Santa Catarina—UFSC, Campus Universitário Trindade 320, 88040-900 Florianópolis, SC, Brazil

Keywords DICOM · Hierarchical data format · PACS · Data Indexing

Introduction

The timeframe during which a given medical imaging examination has to be available depends directly on the laws of each country. In the USA, it varies from 5 to 10 years, with the possibility of extension if there is a governmental demand [1]. In some countries, there are specific timeframes depending on the modality of the examination: In the UK, mammograms have to be stored for 9 to 15 years [2]; in Germany, X-rays must be available for 30 years [3]; and in Brazil, the timeframe is 20 years, independently of the modality [4]. Besides these timeframes, many countries provide also quality criteria for the storage procedures to be employed.

The need for the creation of a highly scalable storage format, allowing extensions in order to encompass any kind of scientific data, motivated the development of the Hierarchical Data Format (HDF) [5]. In its present version, HDF5, it is a widely employed data format that can be found in various kinds of scientific projects, from atmospheric research to financial data [6].

In this work, we analyze the applicability of HDF5 as an underlying structure for Digital Imaging and Communications in Medicine (DICOM)-compliant Picture Archiving and Communications Systems (PACS). This work was performed in the context of the Santa Catarina State Integrated Telemedicine and Telehealth System—STT/SC.

In May 2005, the health office of the state of Santa Catarina, in Brazil, began operating the Santa Catarina Telemedicine Network (Rede Catarinense de Telemedicina (RCTM)) as part of the government plan to decentralize services in the state [7]. During its 5 years of operation, the network has grown, now offering distance exam services in

287 municipalities and at 360 sites, with approximately 53,000 consultations performed monthly and over one million examinations in its database so far. In September 2010, the RCTM and the Brazilian Telehealth Program in Santa Catarina were integrated, forming the STT/SC (Sistema Integrado Catarinense de Telemedicina e Telessaúde), which uses an integrated software platform specifically developed for this purpose by the Federal University of Santa Catarina [8]. The STT/SC offers distance services ranging from highly complex exams, such as magnetic resonance imaging, to clinical analysis, as well as continuing education and support services. It involves the active participation of more than 5,400 health professionals. The existence of a telemedicine and telehealth operation on the scale of STT's raises many questions about the adequacy of a technological model based on a single and specifically developed software platform and its impact on both patients and health professionals. One of our main concerns is the scalability of the PACS that resides behind such a network, considering that the STT/SC employs a statewide database and a strong expansion of the services is expected within the next 5 years. Figure 1 shows the evolution of the image database at STT/SC, considering only permanently stored DICOM image data.

These concerns with storage capacity and access efficiency motivated a set of first experiments that showed promising results [9, 10]. These results were used as a starting point for the work performed later and described here.

The objective of this work is the empirical evaluation of a new approach to efficiently store, search, and retrieve medical images while maintaining the interoperability offered by the DICOM standard. This was performed in order to evaluate the applicability, in the field of PACS, of high-performance scientific data encoding standards developed for distributed environments. In this context, we focused on providing interfaces for the storage, search, and retrieve mechanisms compliant with the DICOM standard. They were implemented as a service layer over data stored using the HDF5 high-performance hierarchical data format together with an inverted index for efficient metadata access employing an application programming interface (API) called CLucene [11].

Related Work

DICOM data structures model information that is hierarchical in nature. This poses the question: Why not employ a hierarchical approach to store DICOM objects? The present-day standard for hierarchical data storage is the HDF, presently in its fifth version, HDF5 [12]. No satisfying answer for this question was found in the PACS-related literature; probably because medical imaging solutions employ either relational database management systems (RDBMS) or simple file system-based directory tree solutions [13]. In this context, a particular commercial RDBMS even offers native DICOM interfaces [14].

Although using HDF5 has been shown to be promising in former storage and retrieval performance tests, some problems during the search operations were detected. Being a hierarchical approach, HDF5 lacks the flexibility of relational approaches in handling metadata that represent secondary keys, which make up most of non-composite object instance data in DICOM objects, i. e., pixel data, waveforms, encapsulated document, SR document, among other contents. The need to create a flexible metadata search mechanism for secondary keys in hierarchical structures posed the need to search for solutions for metadata indexing also in other, nonmedical,

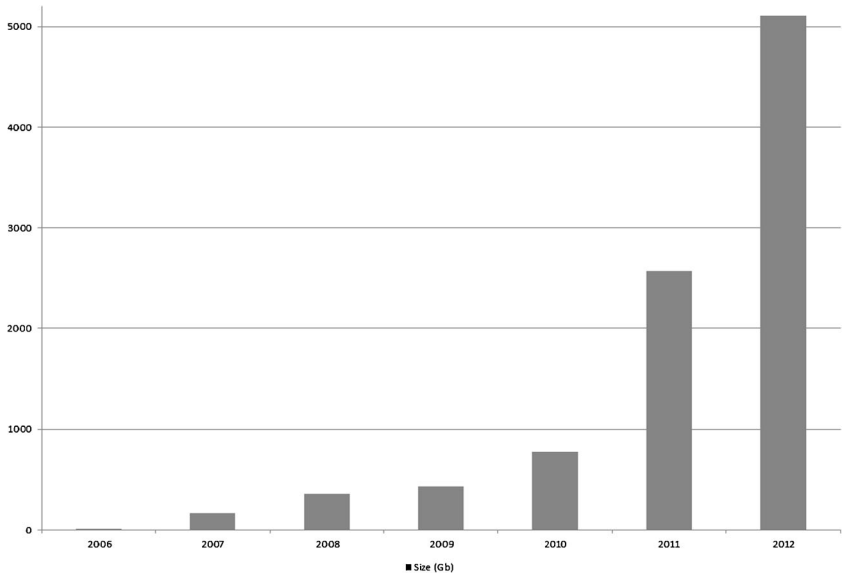

Fig. 1 Number of DICOM images stored since 2006 at STT/SC

areas, which also employ hierarchical databases such as HDF and netCDF [15]. We present them briefly below.

Costa et al. present an approach to substitute the traditional indexing mechanisms for DICOM objects through document indexing using Lucene [16]. They justify it with the higher degree of flexibility offered by document indexing when applied to DICOM objects, which can be seen as a collection of secondary keys. The test setup they validated had the purpose to make available metadata in DICOM file in *part10* syntax stored in directory structures according to [17]. Besides metadata for all standard DICOM searches, all other metadata were indexed, including the generation of thumbnails to facilitate pre-visualizations. Despite the interesting approach, no performance validation study was performed.

Gosink et al. propose an approach where scientific HDF5-stored data are indexed through bitmap indexes, supporting interval queries [18]. This approach, called *HDF5-FastQuery* obtained good performance results with multidimensional data when compared to native HDF5 access. Bitmap indexes, however, are costly to update and are indicated only for nonvolatile data, which applies only partially to medical data, since old data should never be changed, but new data are constantly being added. Since a PACS has to support more search types than interval searches and also support efficient addition of new data, this approach was considered inadequate.

Sahoo and Agrawal propose a solution for the problem of automatically creating efficient data manipulation services for high-level visualization of low-level data [19]. This solution employs a mapping from HDF5 into XML schema in order to achieve a way to manipulate low-level data through functions using XQuery. These functions are afterwards transformed into native HDF5 API functions. This provides for data layout transparency.

Folino et al. propose a solution for managing protein similarity data [20]. The generation of similarity data between sets of proteins generates a huge data volume that is inhomogeneous in size and type and needs an optimal storage and retrieval medium in order to be later processed. Objective of the processing is the generation of a knowledge that allows understanding the structures of the proteins. The authors perform various experiments comparing HDF5 to RDBMS solutions, concluding that HDF5 is more efficient both in terms of response time and storage space.

Cohen et al. use meteorological data to compare the efficiency of using a vector data type called *VArray* and *Oracle nested tables* against three different scientific purpose data containers: netCDF, HDF4, and HDF5 [21]. They show that HDF5 is the most efficient storage mechanism for large data volumes of meteorological data.

Magnus et al. [22] discussed a study on the application of NetCDF data format as a basic platform for storage of medical DICOM images. The paper compared a standard relational database (RDBMS), HDF5, as a storage backend and NetCDF as storage solutions for medical images. Experimental tests using a real set of images indicated that the time to retrieve images from the NetCDF for large-scale images has a higher latency compared to the other two methods.

Abduljwad and Ning propose an efficient way to manage hierarchical structures represented as XML data [23]. XML data are processed in order to obtain the relationship tree of the information nodes. Afterwards, data are stored in two tables in a RDBMS: one for the hierarchy representation and another for path/value pairs.

Technological Background

In this section, we present a brief overview of the technological foundations upon which this work is built. It is here with the intention to clearly define technologies and methods employed and cited later in this work.

DICOM

DICOM is the name of the standard created by the American College of Radiology and the United States National Electrical Manufacturers Association in the 1990s to standardize the transfer of medical images and data across equipment and computer systems of different manufacturers [24]. The DICOM standard defines both image and data formats and image communication services.

DICOM handles the digital description of real-world entities through entity–relationship models represented through abstractions called Information Object Definitions (IODs) [25]. Each IOD is built from data elements that contain an identifier with the format (GGGG, EEEE) called Unique Global Identifier (UID) and an information part which can be normalized or composite [26]. Composite IODs can be defined across data structures and contain also information needed to contextualize these objects, even if this implies in some redundancy.

DICOM service classes are used to describe which communication services a DICOM-compliant device can offer [27]. A Service Class Provider (SCP) is an entity that provides communication services and a Service Class User is the entity that requests a service. These classifications are role oriented and a software instance or a device can enact both roles.

In this work, we will discuss from the point of view of data persistency only composite IODs and the services related to them. Three types of services will be considered:

- Storage: *Composite Store* (C-Store)
- Search: *Composite Find* (C-Find)
- Retrieval: *Composite Get/Move* (C-Get/C-Move)

DICOM services should be transparent, returning to the involved entities only the status of the operation. Service classes obey the DICOM Message Service Element protocol, an application layer protocol [28]. C-Store services should finish with the object stored on some media for a time window large enough to allow useful retrieval operations [27].

Service Classes

Information contained in a DICOM object possesses a hierarchical relationship to each other. A patient may have, for example, several studies, which are, each one, composed by any number of series and each series possesses a set of images, waveform objects, or structured reports, depending on the modality of the examination. Considering this structure, two kinds of search (C-Find) and retrieval (C-Move/C-Get) services are defined, employing either the patient (*patient root*) or the examination (*study root*) as the root of the search [27]. For study root searches, patient data are considered part of the study information. Figure 2a illustrates the first search model and Fig. 2b the second. Each level presents a series of attributes, which can be treated as image metadata, composed by a unique key, a set of minimally required attributes and optional attributed, divided into DICOM general and vendor-specific attributes.

Several kinds of searches can be performed in a PACS such as:

- Single value: Only exact matches to the passed key are retrieved. Used together with other values in order to restrict searches
- Universal: Query for retrieving all possible values for a given key
- Wildcard: Searches using regular expressions containing wildcards such as "M*" to retrieve, e.g., all patients whose name begins with "M"
- UID List: Group search for a set of keys
- Time window: Search for objects acquired during a given time interval

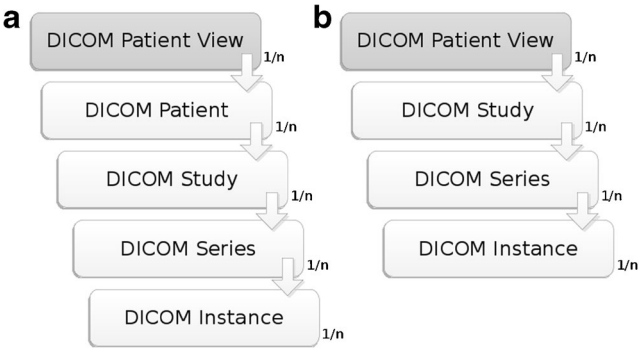

Fig. 2 **a**, **b** DICOM query retrieve model

- *SequenceItem* search: Search within nested data elements. Will not be considered in this study

DICOM object retrieval is performed through the UIDs of the level searched for and through operators for single and universal values and UID lists.

Data Persistency for PACS

As the DICOM standard does not impose the storage technology to be used to achieve data persistency in a PACS, different approaches have been implemented. In this section, we will provide a brief review of these technologies.

The syntax and structure of a stored binary DICOM object are defined in a document called *DICOM Part 10* [29]. A RDBMS is the most commonly used form to implement data storage. Several commercial and open-source solutions exist, where PostgreSQL is an example of an open-source product [30]. RDBMS are traditionally operated by application software using the Structured Query Language (SQL) [31] and data are organized in tables. Individual data elements, handled as table fields, are normally simple data, but some RDBMS offer the possibility to manipulate data called *large objects*, which are generic chunks of binary data of any size.

Hierarchical Data Format

The HDF was created by the US National Center for Supercomputing Applications aiming at providing a means for the efficient long-term store of large volumes of scientific data expressed as hierarchies [32]. HDF data are manipulated through APIs, which exist for FORTRAN and "C." The present version is HDF5, which is structured in *groups* and *datasets* [33]. In analogy to a file system, a group would represent a directory and a dataset an individual file. Groups can recursively be containers of other groups. Datasets are self-descriptive and can be composed by atomic and composed data types and support multidimensional internal data representations.

HDF5 offers various features, besides the hierarchical representation on itself, which can be useful for the storage of DICOM objects:

- Random access: HDF5 allows access with fine granularity, enabling the access of individual metadata or pixel data
- Portability and platform independence: HDF5 was developed for a wide variety of platforms and is a well-documented and open-source solution
- Manipulation of large data quantities: Useful for telemedicine or hospital networks with large centralized databases
- Long-term storage: Planned to be used for data stored for decades, which is a requirement of medical data

Internally, HDF uses a B-tree implementation in order to accelerate data recovery. This can be a disadvantage for users that require a kind of data recovery similar to traditional RDBMS and surely is a handicap when the indexation of highly redundant metadata is required. To overcome this, different approaches are found in the literature [18, 34, 35]. In our approach, we followed a strategy that employs inverted indexes, a traditional solution for the indexation of keywords in text files and data with a large quantity of secondary keys such as metadata elements in DICOM objects. For the purpose of the experiments described here, we decided not to implement inverted indexation, but to employ an open-source API called CLucene [11].

CLucene

Lucene is a project of the Apache Foundation that offers a means for the indexation and retrieval of metadata in documents through an API originally developed in Java [16]. These metadata are grouped by Lucene through abstractions called documents, where each document will be indexed. Internally, Lucene uses inverted indexes that can be *stored* (retrievable), *indexed* (searchable), or *tokenized* (an optimized representation for faster searches). All three attributes can be combined. Various document search tools are also offered that allow all kinds of searches supported by inverted indexes, such as wildcard, interval, list of terms, and Boolean searches. Besides, Lucene allows for relevance-based searches of terms based upon statistical measures such as the Vector Space Model [36]. As stated earlier, in this work, we employed an API called CLucene, which is an implementation in "C" of the original Lucene Java API [11].

Methods

In this section we briefly describe the methods presently being used to provide PACS services at the Santa Catarina State Integrated Telemedicine and Telehealth System and also the new approach proposed in this work. Both approaches were developed using open-source tools and run in GNU/Linux environments.

Relational Database Management Systems at STT/SC

In order to allow for the reproducibility of our results, we tested our approach against an open-source RDBMS, which was considered the golden standard in the context of this work. We employed PostgreSQL ver. 8.4, which is distributed together with GNU/Linux Ubuntu Server 10.10. This is a mature open-source RDBMS that has found wide acceptance for more than 15 years and that effectively supports large objects and databases of more than 1 terabyte. It also is the same implementation which is part of the *Cyclops-DicomServer*, the DICOM server that is one of the elements of the PACS used at the STT/SC and that successfully has served the whole statewide PACS network for more than 6 years now, presently being responsible for the storage of more than 1.5 million studies [37]. An important point about it that is all these studies are distributed in servers and distributed databases, causing a data management problem. This version of PostgreSQL was also chosen because it is in conformity with the ACID (atomicity, consistency, isolation, durability) reliability properties and presents an implementation of the SQL standard that is fairly complete [38]. It also provides programming interfaces for most common languages and a set of third-party extensions that allow coping with most different scenarios. Our installation employed the default 8,192 bytes paging of PostgreSQL and had the shared memory cache set to 24 megabytes.

In order to avoid the overhead generated by communication and security layers present at a statewide Internet-PACS and allow for reproducibility of our results, for our tests we implemented a set of simple interfaces to the RDBMS that simulate the SCPs. The tables, columns and data types were created based upon the entity-relationship (E-R) model given in [27]. Each of the four levels is represented by a separate table. All mandatory and non-null attributes at each level are created as primary keys and also as foreign keys for the next level. Besides the representation of the *data elements* as columns accordingly to [27], the last level contains an *object identifier* (OID) linked to a *binary large object* (BLOb), which contains a DICOM object in *part10* syntax. The DICOM-to-E-R modeling can be seen in Fig. 3a.

C-Store

The storage operation is performed through the SQL commands *select, insert* and *update*. A relationship between the levels of a DICOM object is created through the unique IDs at each table. At the last level, binary storage operations for each DICOM object are executed, as shown in Fig. 3a all tables and columns are affected by this operation.

C-Find

The C-Find service is executed over the columns of all tables, except the OID of the Instance table, as shown in Fig. 3c. We employ the relational search model described in [27]. The requesting service indicates the operation level and one or more keys that should be retrieved. Then a mapping into an SQL query occurs employing the table as the level and the columns as the keys.

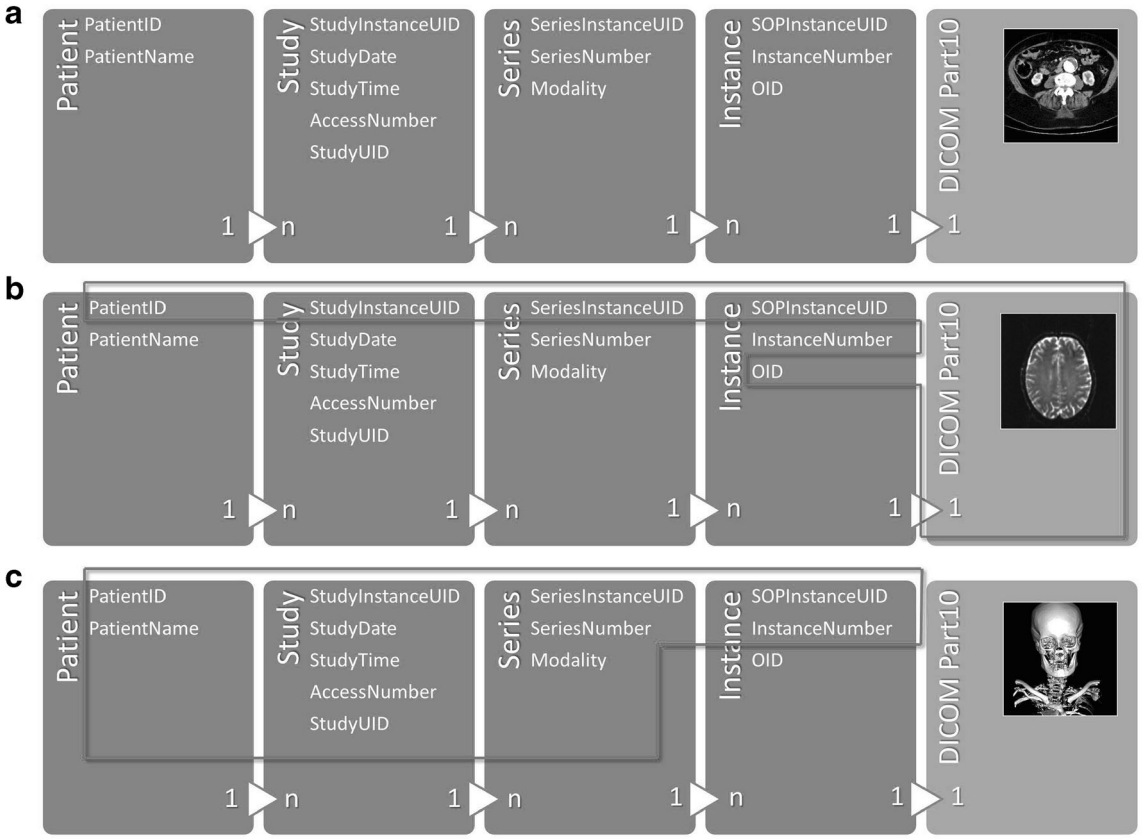

Fig. 3 **a–c** DICOM query retrieve model represented by its entity–relationship model

C-Move and C-Get

As shown in Fig. 3b, the primary keys of each level are queried and subsequently the DICOM binary objects of each instance are retrieved through these keys employing the PostgreSQL API and sent to the requesting service.

A PACS Employing Hierarchical Data Storage and Inverted Indexing

In this section we describe our proposal of a PACS employing hierarchical data storage and inverted indexing using HDF5 and CLucene. HDF5 allows expressing DICOM objects more naturally, as hierarchical structures. Each level is represented through a group that has its UID as an identifier. The *datasets* of each group are a representation of the DICOM *data elements* at each level. The translation of the hierarchical structure of an IOD into HDF5 can be seen in Fig. 4: Ellipses represent the containers that group datasets, shown as rectangles, or other groups. Each group is identified by the value of the data element and it dataset is identified by data element tag. Under the root group ("/") sits the *Study* level, which, besides containing the examination and patient metadata, contains the groups representing the *Series* belonging to the examination and its metadata. Finally, connected to each Series, are the metadata of each *Instance*. The way to access an element from HDF5 is through Unix-like path starting from root, i. e., to retrieve the pixel data it is necessary to provide the given path: "/[Study Instance UID value]/[Series Instance UID value]/[SOP Instance UID value]/7FE00010". The experiments described in this paper were built upon a previous implementation of our group [9, 39]. This code was ported from the C++ language to "C" in order to allow the use of the Message Passing Interface for parallel access, which is not supported by the C++ version of the HDF5 API.

Indexes are created in three different kinds of documents, one for each search and retrieve level established in the DICOM standard and illustrated by Fig. 2. Each document also contains a unique key for the levels above and its own obligatory key. This means that a document at the *Study* level contains the *study instance UID*, at the *Series* level contains the *study instance UID* and the *series instance UID*, and at the *Instance* level all identifiers above plus the *sop instance UID*, as shown in Fig. 5. The terms indexed with Lucene are marked as *stored, indexed*, and *not tokenized*, allowing for search and retrieval operations.

C-Store

The service provider entity is responsible for receiving and storing an IOD, creating conditions for the later search and

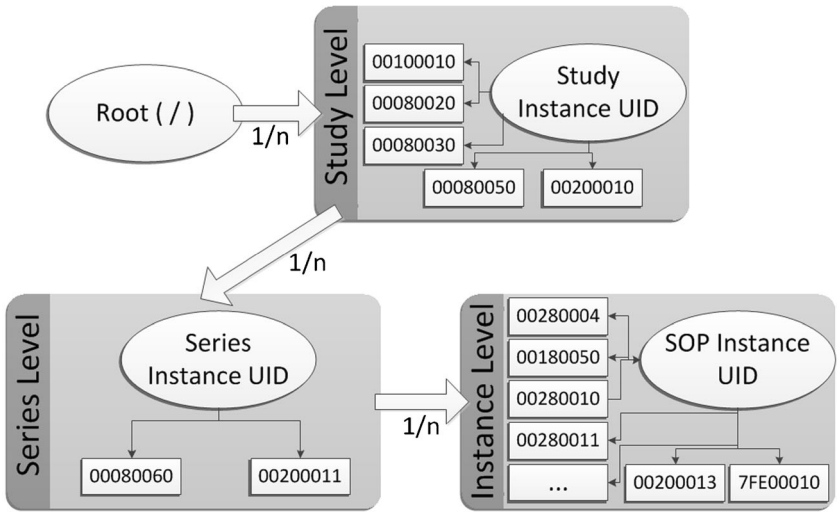

Fig. 4 DICOM object mapped to HDF5

retrieval operations. An IOD is mapped into the HDF5 format through the HDF API with the guidance of an XML descriptor file that contains a representation of the intended storage model. Immediately after and using the same descriptor file, the creation of the metadata indexes is performed through a call to the CLucene API.

C-Find

The C-Find service performs search through the indexes previously stored during C-Store operations. All attribute sets belonging to the requested level were retrieved. The searches *unique value*, *wildcard*, and *range* are natively supported by Lucene and performed accordingly to [27]. The *universal* operator searches are executed obtaining all values of a field and the UID list search is performed through the combination of unique value searches.

C-Move and C-Get

The first step in a search operation is obtaining the needed unique keys from the inverted index using the search key for the required level. As an example, if only the UID of a Study is given, the keys for of its Series and Instances are necessary, and with them, the retrieval of the binary DICOM objects from the HD5 is performed. If *pixel data* are requested, then the UIDs for Study, Series, and Instance are retrieved from CLucene and then the DICOM pixel data (07FE, 0010) object is accessed in HDF through the path/*study*/*series*/*instance*/07FE0010.

Experimental Results

This section describes the results obtained with the evaluation performed with all SCPs, covering storage, search, and

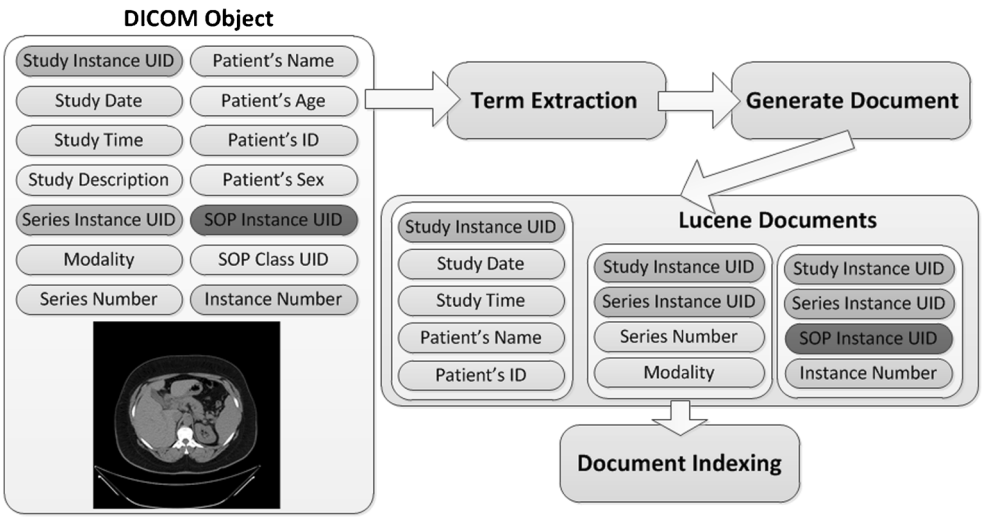

Fig. 5 Extraction of the terms and index creation from DICOM object

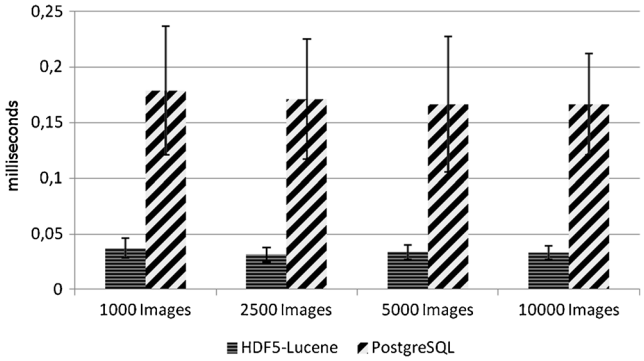

Fig. 6 Comparison of average time per image from HDF5-Lucene and PostgreSQL on C-Store results

retrieval. As a test environment, a server with 1 GB RAM, Intel 1.6 GHz processor, and GNU/Linux Ubuntu Server 10.10 was employed. Several repetitions were performed for all tests and each approach (RDBMS and HDF) in order to guarantee a minimum statistical significance. In order to evaluate the scalability of both approaches, each battery of 25 tests was repeated with databases composed of 1,000, 2,500, 5,000, and 10,000 DICOM image objects in different modalities for storage and retrieval evaluations. The sample was composed by four modalities: CT (75 %), MR (12 %), US (7 %), and CR (6 %). Individual store and retrieval times were recorded for each operation; therefore, the collected measures were 25,000, 625,000, 1,250,000, and 2,500,000. For search evaluation, 1,000 repetitions were made for each kind of operation to increase the sample.

All previously described five kinds of search operations of the DICOM retrieval model were performed with both approaches. Three different retrieval scenarios were taken into consideration, with different search levels for each scenario: only a single DICOM Instance and all instances of a Series and all series of a complete Study. First, the C-Store operation is performed, and then, the created database is employed by the C-Find and C-Get/C-Move operations.

In Fig. 6, the storage service performance between the hierarchical (HDF5-Lucene) and the relational (PostgreSQL) approaches is compared. The mean storage time in the

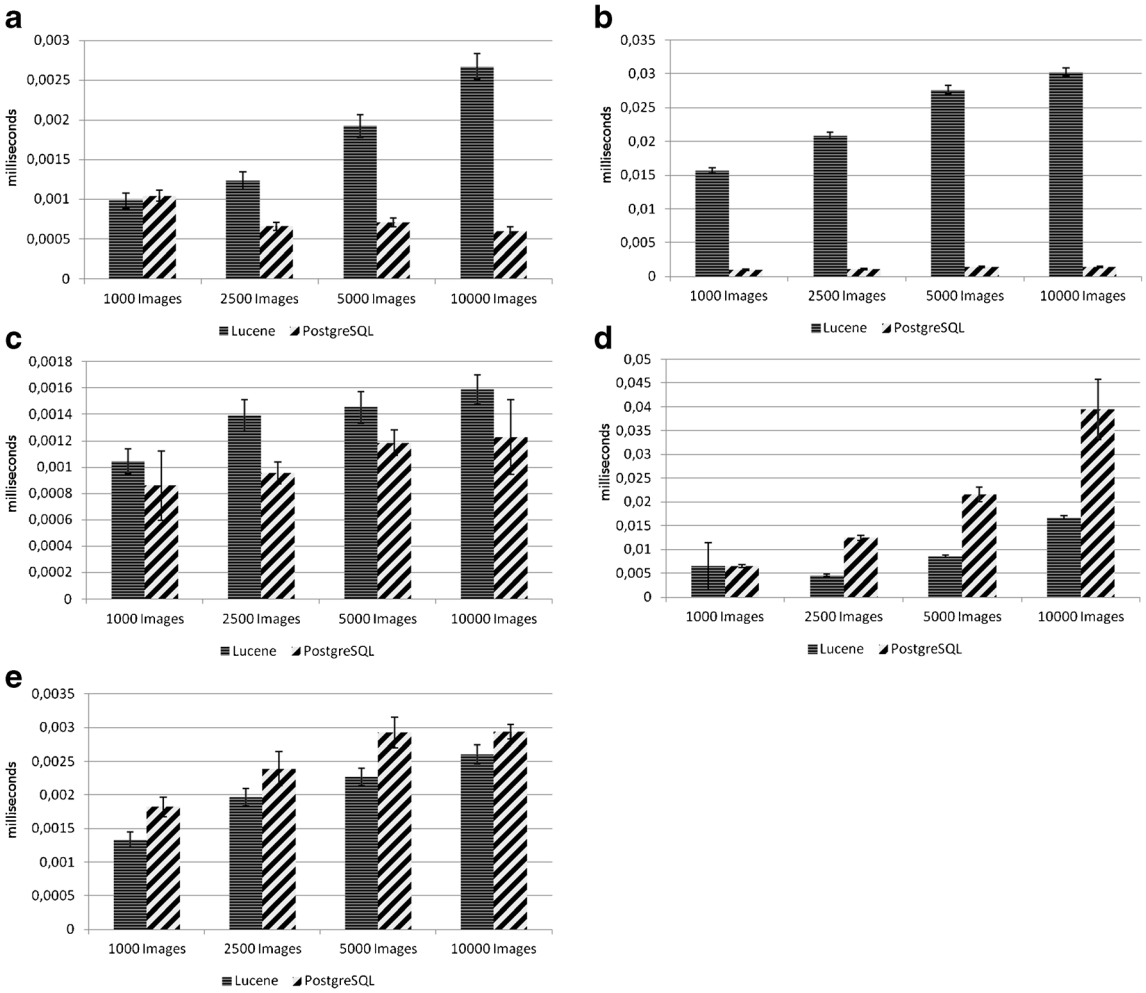

Fig. 7 Each C-Find operation. **a** List of UID. **b** Range. **c** Single value. **d** Universal. **e** Wildcard

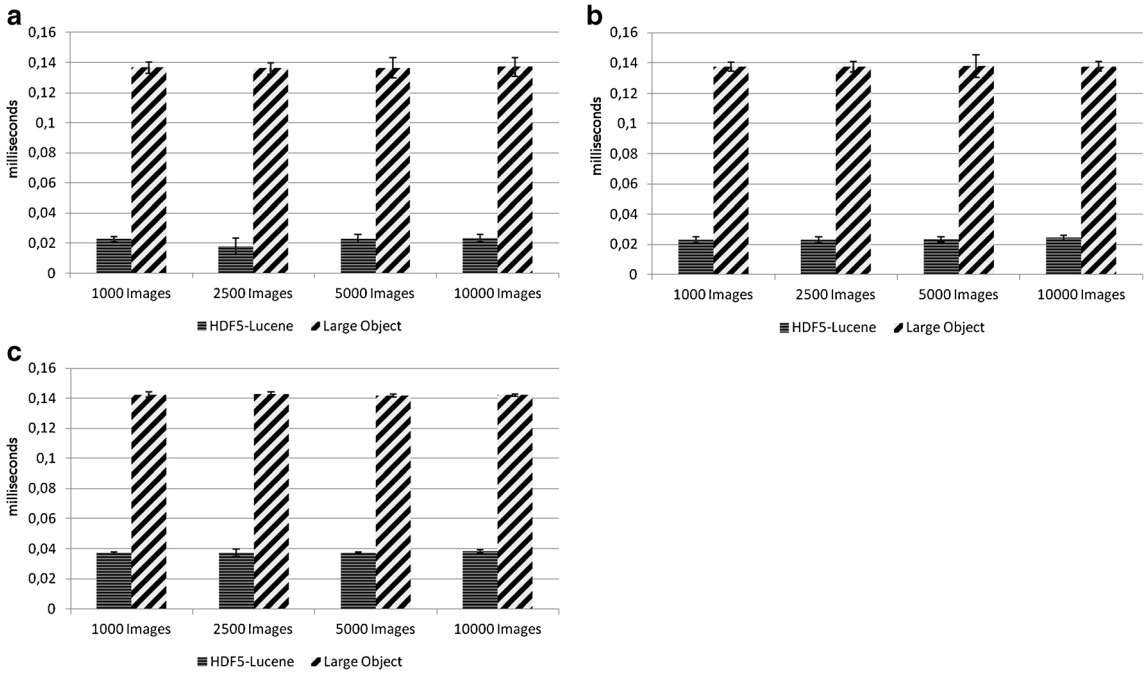

Fig. 8 Results of each level from C-Move. **a** Study level. **b** Series level. **c** Instance level

hierarchical approach is 600 % better, when compared to the relational solution.

In Fig. 7, the results for the search operation are shown. When a list of UID queries is performed, the CLucene approach tends to grow linearly while the relational database approach shows a stable performance. Range query behaviors shown by PostgreSQL and CLucene are the same as in the list of UID query. The behavior of single value query is similar both for PostgreSQL as for CLucene, with a tenuous advantage in milliseconds for PostgreSQL. There is a linear growth on universal query for CLucene and PostgreSQL, but the growth rate of PostgreSQL is higher than in CLucene. The same linearity happens in wildcard query, with a slightest performance advantage in milliseconds of CLucene.

In spite of the inferior results, where searching inverted indexes was involved, in comparison with the relational approach, the retrieval results depicted in Fig. 8 show that the mean total retrieval time for an Instance in the hierarchical approach is better. A retrieval operation first searches for UIDs and metadata, either in the CLucene inverted indexes or in the PostgreSQL internal indexes, and then reads the Instance data, which is performed either directly from the HDF5 hierarchy or from a PostgreSQL *large binary object*. The mean retrieval time for a complete Study was 630 % better for the hierarchical approach, whereas the mean retrieval time for a Series was 580 % better for the hierarchical approach and the mean retrieval time for an individual DICOM image instance was 380 % better for the hierarchical approach.

Table 1 Summary of all performed DICOM operations on each approach

		HDF5				PostgreSQL			
Number of images		1,000	2,500	5,000	10,000	1,000	2,500	5,000	10,000
C-Store		3 min	7 min	17 min	36 min	8 min	21 min	40 min	85 min
C-Move	Study	17 s	17 s	16 s	15 s	20 s	20 s	24 s	18 s
	Series	14 s	14 s	13 s	12 s	16 s	16 s	19 s	16 s
	Instance	4 s	4 s	4 s	4 s	4 s	4 s	5 s	10 s
C-Find	List of UID	4 s	4 s	4 s	4 s	4 s	4 s	4 s	4 s
	Range	5 s	5 s	5 s	5 s	4 s	5 s	5 s	5 s
	Single value	4 s	4 s	4 s	4 s	4 s	4 s	4 s	4 s
	Universal	3 s	4 s	5 s	8 s	5 s	6 s	7 s	20 s
	Wildcard	4 s	4 s	4 s	4 s	4 s	4 s	4 s	4 s

Table 2 Size of DICOM objects on each approach

Number of images	1,000	2,50,0	5,000	10,000
HDF5	695 MB	1.7 GB	3.4 GB	6.6 GB
CLucene	1,004 k	1.2 MB	2 MB	3 MB
PostgreSQL	488 MB	1.2 GB	2.4 GB	4.7 GB
File system	562 MB	1.3 GB	2.7 GB	5.3 GB

Table 4 Overhead in megabytes of internal structures from PostgreSQL

Number of images	Number of rows	Table size	Index size	Total size
1,000	3,176	624	328	952
2,500	6,544	1,336	688	2,024
5,000	12,314	2,352	1,176	3,528
10,000	22,727	4,344	2,072	6,416

The consolidated performance of the experiments is summarized in Table 1. Even with some overhead presented in the results of the use of Apache Lucene on C-Find operations, the approach HDF5-CLucene shows a better overall performance.

Table 2 summarizes storage space usage. PostgreSQL employs an 8-KB page and thus does not use up much extra space [30]. HDF5, on the other side, allocates four different internal indexing structures, a B-Tree, a Heap, Group indexes, and Dataset indexes, which are more space intensive but also respond for a better performance.

The storage overhead of the internal structures of HDF5 is described in Table 3. The B-Tree indexes internal data, and the Heap stores addresses and the names of the links to objects. Groups and Datasets store metadata information. These structures are allocated in blocks of 2,048 bytes, whose size can be changed in order to optimize storage space. We employed the default size.

The number of rows, index size, and table size in kilobytes of the employed PostgreSQL tables is described in Table 4. It shows a proportional growth of the internal structure of the database as the number of images increases.

Discussion

We presented a new approach for the development of a data persistency layer for DICOM objects employing a hierarchical database. Our approach makes use of the HDF5 data storage standard for scientific data. Our approach overcomes limitations of hierarchical databases employing an inverted indexing approach for secondary key management and for efficient and flexible access to data through secondary keys. This inverted indexing is achieved through a general purpose document indexing tool called Lucene.

Experiments were performed on real-world data that have shown that our approach outperformed a traditional PACS solution, as is used in our own real-world, statewide telemedicine network, implemented with PostgreSQL. Some store and retrieve operations were up to 500 % faster. On the other side, for some situations, the relational approach showed slightly better results for search operations. These last results are due to the overhead generated by the inverted indexing used for secondary keys, which employed Apache Lucene, a general purpose, open-source document indexing tool. There is one case that is shown on Fig. 7d where there is a main advantage on CLucene. It lies in the fact that there is a file where the terms are stored. Instead of having to load all terms from a specific field, it is necessary only to load this file in memory; therefore, there is no real search operation. Still, the interior results shown in Fig. 7a occur because the UID value was stored at the RDBMS in a B-Tree structure that has an access complexity of O (log n). The same occurs in Fig. 7b but for a different reason. The linear growth of CLucene can be explained by query expansion to *BooleanQuery*; therefore, as the index grows, more time will be needed to expand all possible terms indexed [40]. These situations need to be analyzed in depth and, eventually, implementing a secondary key indexing mechanism that retains the inverted index strategy but is tailored for the problem at hand could be shown to be a better solution.

On the other hand, the single fact that HDF5 supported by secondary key indexing using a general purpose document indexing tool was enough to, in most situations, outperform PostgreSQL, which is considered one of the best open-source RDBMS, indicates that a hierarchical database approach to DICOM-compliant PACS data storage is a promising solution.

The overhead generated by the internal, general purpose, structures of the HDF5 is also something that can be

Table 3 Overhead in megabytes of internal structures from HDF5

Number of images	B-Tree	Heap	Groups	Datasets	Total
1,000	73.250336	11.044891	2.9208374	54.9685516	142.184616
2,500	158.61617	24.133049	6.27616882	122.571152	311.5965398
5,000	338.14694	51.055313	13.4405899	256.00621	658.6490529
10,000	673.81306	101.54826	26.7687225	509.721146	1,311.851189

optimized. These structures can be configured and parameterized. For our experiments, we employed the default values. Fine-tuning these structures is surely a try-and-error approach that can lead to a better performance. Taylor used these indexes for DICOM data and developed a *HDF5-PACS* model which would be a step further.

In our experiments, we worked only with image-related DICOM instances. Documents such as findings and reports stored as DICOM Structured Reporting documents were not taken into consideration, although the *CyclopsDicomServer* supports working with such documents and they are widely used in our telemedicine network. To include the possibility to index and retrieve such documents would pose another set of requirements to the secondary indexing strategy, since it would imply that all controlled vocabularies used in all kinds of reports of all the different image modalities would have to be indexed and managed as secondary key values. This seems to be more complicated than it really is because even if the total of the vocabularies represents a huge amount of possible individual data values, they would represent values for only a few metadata fields. We see this possibility as very interesting and feasible and are presently working on a solution.

An important observation about our study is related to the experimental tests, which standard configurations of both solutions were used; thus, they were not evaluated as aspects of RDBMS related to caching, BLOB handling, and optimized traversal through a set of trees. Issues related to transactional protections for both solutions still have not been evaluated. This way, these issues are suggested as future works.

References

1. Acr: Acr Practice Guideline for the Performance of Screening and Diagnostic Mammography, American College of Radiology, 2008, p 1
2. RCR: Retention and Storage of Images and Radiological Patient Data. The Royal College of Radiologists, London, 2008, p 3
3. BMJ: Verordnung über den Schutz vor Schäden durch Röntgenstrahlen. Bundesministerium der Justiz, 1987
4. CFM: RESOLUÇÃO CFM Nº 1.821/07, CFM, Editor. Conselho Federal de Medicina, 2007
5. HDFGROUP: The HDF Group - Information, Support, and Software. 2011 [cited 2012 Jun 05]; Available from: http://www.hdfgroup.org/
6. HDFGROUP: Who uses HDF? 2011 [cited 2012 Jun 05]; Available from: http://www.hdfgroup.org/users.html
7. Wallauer J, Macedo DDJ, Andrade R, and Von Wangenheim A. Building a national telemedicine network. IT professional 10(2):12–17, 2008
8. Wangenheim A, Junior CLB, Wagner HM, Cavalcante C: Ways to implement large scale telemedicine: The Santa Catarina Experience. Lat Am J Telehealth 1(3):364–377, 2010
9. Macedo DDJ, Von Wangenheim A, Dantas M, Perantunes HGW: An architecture for DICOM medical images storage and retrieval adopting distributed file systems. Int J High Perf Syst Arch 2(2): 99–106, 2009
10. Macedo DDJ, Capretz MAM, Prado TC, von Wangenheim A, Dantas M: An improvement of a different approach for medical image storage. 2011
11. CLucene: CLucene - lightning fast C++ search engine. 2011 [cited 2012 Jun 05]; Available from: http://clucene.sourceforge.net/
12. HDFGROUP: The HDF Group. Hierarchical data format version 5, 2000-2010. 2010; Available from: http://www.hdfgroup.org/HDF5
13. Eichelberg M, Riesmeier J, Wilkens T, Hewett AJ, Barth A, Jensch P: Ten years of medical imaging standardization and prototypical implementation: The DICOM standard and the OFFIS DICOM Toolkit (DCMTK), 2004
14. ORACLE: Oracle Database 11g DICOM Medical Image Support. 2011 [cited 2012 Jun 05]; Available from: http://www.oracle.com/technetwork/database/multimedia/overview/dicom11gr2-wp-medimgsupport-133109.pdf
15. Rew R, Davis G: NetCDF: an interface for scientific data access. Comput Graph Appl 10(4):76–82, 1990
16. Erik H, Otis G, Michael MC: Lucene in action. Manning Publications Co., 2005
17. Costa C, Freitas F, Pereira M, Silva A, Oliveira J: Indexing and retrieving DICOM data in disperse and unstructured archives. Int J Comput Assist Radiol Surg 4:71–77, 2009
18. Gosink L, Shalf J, Stockinger K, Wu K, Bethel W: HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices. In: Scientific and Statistical Database Management, 2006. 18th International Conference on, 2006
19. Sahoo S, Agrawal G: Supporting XML Based High-Level Abstractions on HDF5 Datasets: A Case Study in Automatic Data Virtualization. In: Eigenmann RL, Midkiff S Eds. Languages and Compilers for High Performance Computing. Springer, Berlin, 2005, pp 922–922
20. Folino G, Shah AA, Kransnogor N: On the storage, management and analysis of (multi) similarity for large scale protein structure datasets in the grid. in Computer-Based Medical Systems, 2009. CBMS 2009. 22nd International Symposium on, 2009
21. Cohen S, Hurley P, Schulz KW, Barth WL, Benton B: Scientific formats for object-relational database systems: a study of suitability and performance. SIGMOD Rec 35:10–15, 2006
22. Magnus M, Prado TC, Wangenheim Av, Macedo DDJ, Dantas MAR: A Study of NetCDF as an Approach for High Performance Medical Image Storage. Journal of Physics: Conference Series, 2012
23. Abduljwad F, Ning W, De X: SMX/R: Efficient way of storing and managing XML documents using RDBMSs based on paths. 2010
24. NEMA: Digital Imaging and Communications in Medicine (DICOM) Part 1: Introduction and Overview, 2011
25. NEMA: Digital Imaging and Communications in Medicine (DICOM) Part 3: Information Object Definition, 2011
26. NEMA: Digital Imaging and Communications in Medicine (DICOM) Part 5: Data Structures and Encoding, 2011
27. NEMA: Digital Imaging and Communications in Medicine (DICOM) Part 4: Service Class Specifications, 2011
28. NEMA: Digital Imaging and Communications in Medicine (DICOM) Part 8: Network Communication Support for Message Exchange, 2011
29. NEMA: Digital Imaging and Communications in Medicine (DICOM) Part 10: Media Storage and File Format for Media Interchange, 2011
30. Douglas K: Postgre SQL. Sams, 2005
31. ISO/IEC: Information technology – Database languages, in Part 1: Framework (SQL/Framework). 2008, ISO/IEC
32. Pourma E, Folk M: Balancing Performance and Preservation Lessons learned with HDF5, 2010
33. HDFGROUP: How is HDF5 different than HDF4? 2011 [cited 2012 Jun 05]; Available from: http://www.hdfgroup.org/h5h4-diff.html
34. PyTables: PyTables - Getting the most *out* of your data. 2011 [cited 2012 Jun 05]; Available from: www.pytables.org/

35. Nam B, Sussman A: Improving access to multi-dimensional self-describing scientific datasets, 2003
36. Salton G, Wong A, Yang CS: A vector space model for automatic indexing. Commun 18(11):613–620, 1975
37. Maia RS, von Wangenheim A, Nobre LF: A statewide telemedicine network for public health in Brazil. 2006
38. PostgreSQL: SQL Conformance. 2009 [cited 2012 Jun 05]; Available from: http://www.postgresql.org/docs/8.4/static/features.html
39. Amaral E, Comunello E, Dantas MAR, Macedo DDJ: Replicação Distribuída de Imagens Médicas sob o Formato de Dados HDF5, in 8th International Information and Telecommunication Technologies Symposium 2009, I2TS: Florianópolis, SC - Brazil
40. Lucene A: Lucene FAQ. 2011 12-2011 [cited 2012 Jun 05]; Available from: http://wiki.apache.org/lucene-java/LuceneFAQ#Why_am_I_getting_a_TooManyClauses_exception.3F