# Relational Databases versus Search Engines: A Performance Comparison for Storing and Querying DICOM Metadata

**Alexandre Savaris**[1], **Gabriela Bussolo Colonetti**[1,2], **Rodrigo Rodrigues Pires de Mello**[1,3], **Aldo von Wangenheim**[1,3]

[1] Telemedicine Laboratory
Brazilian Institute for Digital Convergence – INCoD
Florianópolis, Santa Catarina, Brazil

[2]Graduate Program in Information Systems – SIN
Federal University of Santa Catarina – UFSC
Florianópolis, Santa Catarina, Brazil

[3]Graduate Program in Computer Science – CCO
Federal University of Santa Catarina – UFSC
Florianópolis, Santa Catarina, Brazil

{savaris, gabriela, rmello}@telemedicina.ufsc.br, aldo.vw@ufsc.br

***Abstract.*** *The Digital Imaging and Communications in Medicine (DICOM) standard adopts files as individual, self-contained repositories for the storage of a mixed of alphanumerical and binary content regarding radiological images. Usually, groups of DICOM files are hierarchically organized in studies and series, physically disposed into filesystem directory trees. Despite its simplicity in storing content, ordinary filesystems do not provide index capabilities allowing searches by content – restricting access by directory names and file names. To surpass such limitation, Picture Archiving and Communication Systems (PACSs) often adopt Relational Database Management Systems (RDBMSs) as metadata repositories, benefiting from its general-purposed index structures. An alternative approach, not quite explored, considers the adoption of search engines as metadata catalogs, aiming to minimize the search time by exploring the engine's index optimizations. In order to evaluate the performance on managing DICOM metadata, this work compares relational database instances to a search engine in terms of storage space, storage time, and query time. Results show that, in the best case, the search engine is slightly slower in storing content; however, it requires 69% less disk space for the same dataset. For queries, in turn, the search engine performs up to 8.3 times faster in retrieving groups of tags.*

## 1. Introduction

Picture Archiving and Communication Systems (PACSs) have been developed over three decades [Huang 2011, Lemke 2011], incorporating new technologies and evolving into a complete integrated system far beyond radiology [Faggioni et al. 2011]. Nevertheless, the management of digital images continue to be its most appealing characteristic – resulting in improvements on productivity and efficiency, and in attending new or combined examination modalities [Mansoori et al. 2012, Singh et al. 2011].

Digital images stored and managed by PACSs are structured according to the precepts of the Digital Imaging and Communications in Medicine (DICOM) standard [Pianykh 2012]. Each image is stored as a file, being persisted in ordinary or specialized filesystems (according to the PACS implementation); usually, metadata extracted from images are persisted as well – aiming to provide search capabilities without involving the original files. Relational Database Management Systems (RDBMSs) are commonly adopted as repositories for DICOM metadata. It is possible to define database schemas lined up with the structure of DICOM tags, and the provided index structures contribute on reducing query execution time [Savaris et al. 2014]. Alternatives to RDBMSs have been used, exploring NoSQL implementations as a replacement for relational database instances [Rascovsky et al. 2012, Bastiao Silva et al. 2014]; these alternatives provide schemaless storage capabilities, aligned with native scalability and data partitioning.

This work investigates the behavior of search engines – tools not quite explored in managing DICOM metadata – acting as a replacement for RDBMSs in indexing and responding to queries. Experiments are performed in order to evaluate the performance on storing partial- and full-content metadata extracted from DICOM images, as well as in querying the stored content using different filter options and returning different sets of tags.

The remainder of this work is organized as follows: Section 2 presents background information about the DICOM standard, RDBMSs, RDB (Relational Database) instances, and search engines; Section 3 describes the experimental environment and experiments performed to measure performance on indexing DICOM metadata; Section 4 presents the obtained results, discussed in Section 5; Section 6 concludes the work, including future directions.

## 2. Background

### 2.1. The DICOM standard

Originally developed by the National Electrical Manufacturers Association (NEMA) in a partnership with the American College of Radiology (ACR), the DICOM standard covers a set of non-proprietary specifications including structure, format, and exchange protocols for digital-based medical images. Since the release of its first version in 1985 as ACR/NEMA 300, the standard evolves according to deliberations of academies, medical device manufacturers, and scientific societies organized as workgroups [Mildenberger et al. 2002, Bidgood Jr et al. 1997].

According to the standard a DICOM image is stored as a self-contained file, grouping basic data elements (tags). Each tag is characterized by a Value Representation (VR) and a Value Multiplicity (VM), specifying the content supported by the tag, formatting rules applicable to the content, and the number of allowed occurrences for the content inside the tag [National Electrical Manufacturers Association 2015b]. Tags with numerical and textual data types are candidates to be included in search and retrieval operations, being used as filters or as returning values for query expressions.

### 2.2. Relational Database Management Systems and Relational Database Instances

A Database Management System (DBMS) is a collection of interrelated and persistent data, together with a set of programs responsible for accessing that data and by guaran-

teeing availability, integrity, security, and independence [Sumathi and Esakkirajan 2007]. DBMSs implementing the relational model defined in [Codd 1970] are specified as RDBMSs. RDB instances, in turn, can be defined as a set of physical structures organized according to the relational model, responsible for the effective data storage and managed by RDBMSs.

The physical structure of an RDB instance follows a well-defined database schema, which describes and specifies its component objects (e.g. tables, fields, indexes, integrity constraints) [Elmasri and Navathe 2010]. To be inserted into an RDB instance, data must respect the underlying schema restrictions, which establishes a set of boundaries. Changes in data storage requirements for an RDB instance demand adjustments in its database schema definition.

### 2.3. Search engines

A search engine can be characterized as a practical application of Information Retrieval (IR) techniques, delivering performance, scalability and adaptability to scenarios demanding ranking, evaluation, and information storage, search and retrieval [Croft et al. 2009]. Commonly organized for storage and management of documents, it has loose restrictions when compared to RDB instances in terms of data schemas.

In performing transformations to the original stored content, including tokenization, stopping, and stemming [Baeza-Yates and Ribeiro-Neto 2011], search engines build high-performance indexes capable of answering to heterogeneous queries, with expressive results when compared to b-tree and hash indexes available on RDBMSs.

## 3. Materials and Methods

In order to evaluate the behavior of RDB instances and the search engine in managing DICOM content, this work adapts a series of experiments from [Savaris et al. 2014] for measuring storage space, storage time, and query time.

### 3.1. Experimental environment

The setup adopted for the experiments is based on the client-server model, with two nodes communicating to each other at a time through a 1Gbps LAN as depicted in Fig. 1. Server configuration: Intel® Xeon® X7460 2.66GHz, 1GB DDR3 RAM, 536.9GB SATA-HD, Debian 7.9 wheezy. Client configuration: Intel® Core™ i7-3610QM 2.30GHz, 8GB DDR3 RAM, 500GB SATA-HD, Linux Mint 17.2 rafaela.

For storage experiments, the client side is similar for both RDB instances and the search engine. Using dcm4che[1] v.2.0.28 (an open source implementation of the DICOM standard), a dataset composed by radiological images is read from an ordinary filesystem and sent to the server according to the DICOM network protocol via C-STORE [National Electrical Manufacturers Association 2015a]. The server side, in turn, has particularities for each evaluated implementation. RDB instances (Fig. 1, top) receive the radiological images through dcm4chee[2] v.2.18.1 (an open source DICOM clinical data management system, acting as a DICOM archive). For each received image, alphanumerical tags are extracted and inserted into a relational database instance managed by PostgreSQL[3] v.9.1.18 (an open source ORDBMS – Object-Relational Database Management System); next, the content from the image file is archived into a filesystem directory

tree. The search engine (Fig. 1, bottom), implemented using Elasticsearch[4] v.2.1.1, receives the radiological images through Mirth® Connect[5] v.3.3.1.7856.b91 (an open source interconnection engine). For each received image, its DICOM content is converted to an XML (eXtensible Markup Language) representation followed by the extraction of its alphanumerical tags, which compose a JSON (JavaScript Object Notation) document whose key-value pairs are fully indexed.

For queries, both client and server sides differ. RDB instances (Fig. 1, top) respond do C-FIND [National Electrical Manufacturers Association 2015a] operations fired from the client using dcm4che. Once received by dcm4chee on the server, the search request is converted to a single or multiple SQL instructions used to access the underlying relational database instance. Tag values who satisfy the search criteria compose a resultset, sent from the ORDBMS to dcm4chee; the archive, then, converts the resultset to a proper format according to the DICOM network protocol, returning it to the requester. The search engine (Fig. 1, bottom) is accessed through HTTP (Hypertext Transfer Protocol) GET requests sent from the client using cURL[6] v.7.35.0, an open source multi-protocol library and command line tool. Each request (received as a JSON document) is processed by Elasticsearch, who returns another JSON document including the found resultset.

[1] https://dcm4che.atlassian.net/wiki/display/lib/
[2] https://dcm4che.atlassian.net/wiki/display/ee2/
[3] http://www.postgresql.org/
[4] https://www.elastic.co/products/elasticsearch
[5] https://www.mirth.com/Products-and-Services/Mirth-Connect
[6] https://curl.haxx.se/

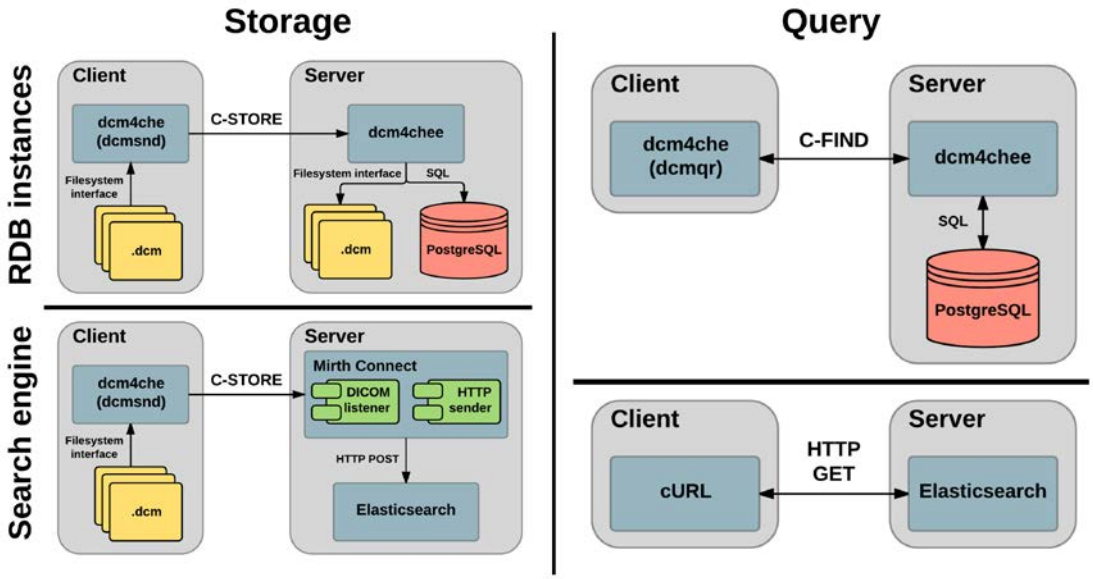



**Figure 1. Experimental environment for storage and query measurements using RDB instances and the search engine.**

### 3.2. Experiments

Both storage and query experiments are performed using computed tomography (CT) images from the public available LIDC-IDRI DICOM dataset [The Cancer Imaging Archive Team 2015], an integrating part of The Cancer Imaging Archive (TCIA) [Clark et al. 2013]. The adopted dataset includes 243,954 images with a resolution of 512x512, distributed in 1,010 series and in 1,010 studies belonging to 1,010 patients, occupying 128.4GB of disk space.

Experiments performed for storage measurements send the whole dataset to both RDB instances and to the search engine. Particularly to the relational database instances, two configurations made in dcm4chee are evaluated: the *original* configuration and the *extended* configuration. In the original configuration (depicted in the result charts as OC), only tags that physically exist on the database schema as fields are persisted; remainder tags are ignored. In the extended configuration (depicted in the result charts as EC), tags that physically exist are stored into its respective fields; remainder tags are concatenated into BLOB (Binary Large Object) fields, and become available for filtering and retrieval.

Query experiments are performed on a hierarchical-level basis; for each level defined in the DICOM standard (patient, study, series, and image) a set of tags is retrieved (if available) from the stored dataset. Tags are retrieved individually or as groups of 5, 10, 15, 20, 25, and 30 tags, and are filtered by the unique identifiers of each level (*patientid*, *studyinstanceuid*, *seriesinstanceuid*, and *sopinstanceuid*) up to the top of the hierarchy. All queries are executed 10 times each, using filter values randomly selected from the dataset.

## 4. Results

The results acquired through experiments described in Section 3 depict the behavior of RDB instances and the search engine in terms of the storage space needed to persist the DICOM dataset, the time spent for the complete persistence, and the time needed to search and retrieve single or multiple tags at once.

### 4.1. Storage space

The storage space needed for the persistence of the complete DICOM dataset is divided into three volumes: *data*, *indexes*, and *others*. Data comprehend alphanumerical tag values, being stored according to the specificities of each implementation. Indexes are built and updated based on data values; its organization, i.e., storage structure and order, is implementation-dependent. Other structures store complementary data (e.g., data dictionaries, lock control files) used by RDBMSs and search engines for its initial configuration and runtime management. Fig. 2 (left) shows the required space for each volume, for each evaluated implementation.

In observing Fig. 2, it is possible to perceive the lack of the data volume for the search engine. In this work, the search engine indexes JSON documents, each document including the complete metadata set extracted from a DICOM image; therefore, it is assumed that the index volume includes the data volume. RDB instances, in turn, adopt separate physical structures for data and indexes, using the former as tag repositories and the latter as structures for reducing query time. By indexing entire documents, the search

engine requires approximately 51% and 69% less disk space when compared, respectively, to the originally and extended configured RDB instances; when compared alone, the index volume generated by the search engine is approximately 29% and 34% smaller than its counterparts.

The storage space needed for the data volume increases considerably when an extended configuration is adopted for RDB instances: selecting all alphanumerical tags for persistence, the volume grows up approximately three times. Index volumes, in turn, present a slight increasing of 7.73% on size. This behavior indicates that the underlying database schema supports the insertion of new, heterogeneous sets of tags; however, it does not indicate that such tags are indexed.

Despite its importance in regulating RDBMSs and search engine runtimes, the storage space needed by the "others" volume is negligible: it corresponds to only 0.24% of the total space required by all evaluated implementations.

### 4.2. Storage time

The time spent to persist the whole DICOM dataset is divided into the following operations: *C-STORE*, which includes the transmission of all DICOM images from a client to the server hosting the evaluated implementations respecting the DICOM network protocol; conversions from *DICOM to XML*, performed automatically by the interconnection engine at the reception of each DICOM image; conversions from *XML to JSON*, performed by the interconnection engine to provide content supported by the search engine; and requests and responses via *HTTP POSTs*, performed by the interconnection engine to send data to the search engine for indexing. Individual times for each operation can be seen in Fig. 2 (right).

Common to all evaluated implementations, the C-STORE is also the most time-consuming operation. According to the chart in Fig. 2, storing and indexing all metadata tags contribute to increase the storage time in approximately 5.9 and 5.5 times when comparing the originally configured RDB instance to, respectively, the extended configured RDB instance and the search engine. When comparing similar configurations, i.e., those who store the full set of tags, the search engine performs 7% faster.

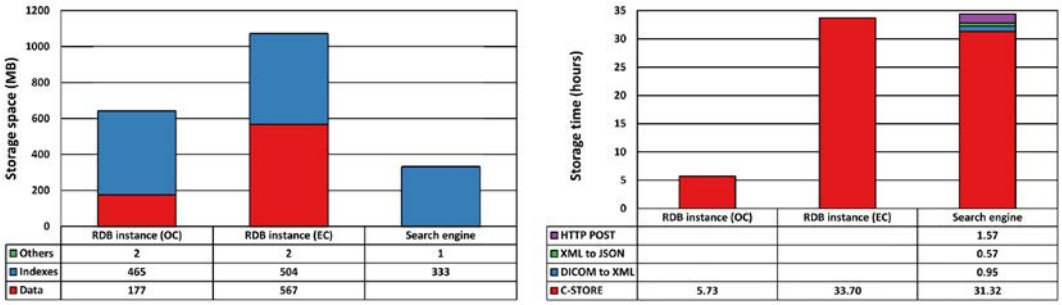



**Figure 2. Left: Storage space, in MB, used to persist the whole DICOM dataset. For both RDB instances and the search engine, the stored content is divided into data, indexes, and other structures. Right: Storage time, in hours, used to persist the whole DICOM dataset. For the search engine, the cumulative time includes conversions between data formats and HTTP POST operations.**

For both RDB instances, the C-STORE operation encompass the transmission of DICOM content from client to server, the parsing and extraction of alphanumerical tags, and the insertion of the extracted tags into the underlying database. For the search engine, in turn, the C-STORE operation performs only the transmission of DICOM content from client to server – parsing, extraction, and indexing are performed by conversion operations and HTTP POST operations. In numbers, converting from XML to JSON correspond to 1.65% of the total storage time, followed by the conversion from DICOM to XML (2.76%) and HTTP POST executions (4.55%). Together with the C-STORE operation, conversions and HTTP POSTs contribute to classify the search engine as the worst option for storage, being six times slower than the originally configured RDB instance and approximately 2% slower than the extended configured RDB instance.

### 4.3. Query time

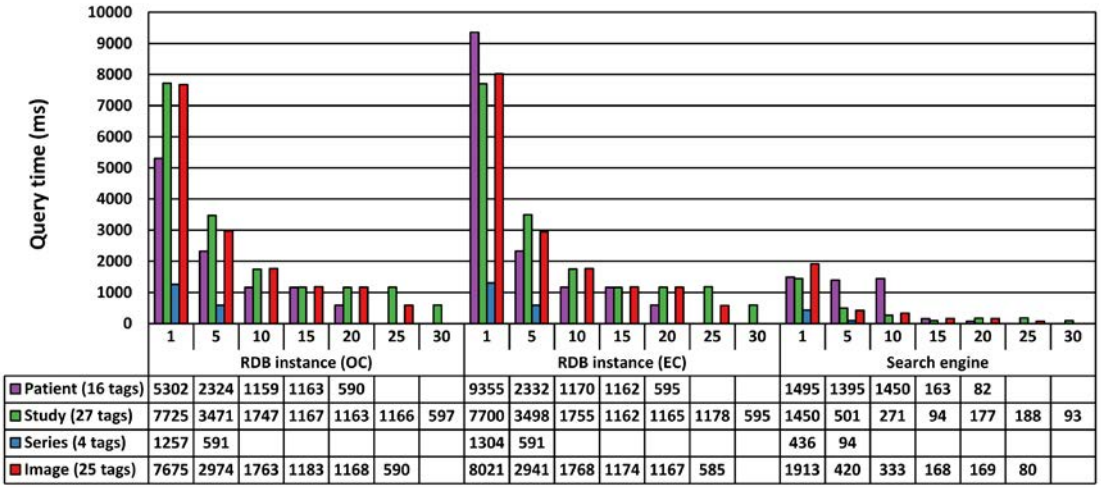
The time spent on searching and retrieving single or multiple tags at once is a sum of individual times for the following steps: a *query*, sent from a client to the server specifying filtering parameters and tags to be retrieved; the *processing*, including the access to the underlying databases and/or indexes to effectively perform the search; and the *response*, sent from the server to the client with the resulting dataset. These steps are equivalent to a DICOM C-FIND operation [National Electrical Manufacturers Association 2015a].

Fig. 3 depicts the behavior of the evaluated implementations in searching for tags related to the four hierarchical levels defined by the DICOM standard (*patient*, *study*, *series*, and *image*). According to the chart, the greater the number of tags per level, the greater the time needed to search and retrieve such tags. This behavior is consistent in both RDB instances, not considering some exceptions (e.g., individual tags, extended RDB instance). Being approximately 10% slower than the originally configured RDB instance, the extended configuration is penalized by the overhead on managing tag values stored into BLOB fields. Despite its flexibility in storing heterogeneous sets of tags, this approach compromises the search performance due to the parsing step needed to identify and extract tags that are not persisted as individual, schema-defined fields.

When compared to both RDB instances, the search engine performs better; it is approximately 4.1 times faster than the original configuration and 4.5 times faster than the extended configuration, summarizing all results. Executing a level-by-level comparison, performance gains on adopting the search engine vary from 2.3 times (related to the originally RDB instance, querying tags for the patient level) to 6.1 times (querying tags for the study level, in both RDB instances). The search engine surpasses both RDB instances, also, in comparing results for queries searching for individual tags or groups of tags: reductions in time vary from 2.3 times (querying groups of 10 tags, in both RDB instances) to 8.3 times (querying groups of 15 tags, in the originally configured RDB instance).

## 5. Discussion

When dealing with DICOM indexing, it is paramount to observe the final objectives to be accomplished. An uncontextualized analysis of the chart presented in Fig. 2 (right) indicates a clear advantage in adopting the originally configured RDB instance, due to the reduced time spent on storing content. For scenarios with limited query possibilities, restricted to the available physical structures, this assumption is valid; however, the referred instance does not provide a direct solution for search and retrieval of tags that are




| | 1 | 5 | 10 | 15 | 20 | 25 | 30 | 1 | 5 | 10 | 15 | 20 | 25 | 30 | 1 | 5 | 10 | 15 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Patient (16 tags) | 5302 | 2324 | 1159 | 1163 | 590 | | | 9355 | 2332 | 1170 | 1162 | 595 | | | 1495 | 1395 | 1450 | 163 | 82 | | |
| Study (27 tags) | 7725 | 3471 | 1747 | 1167 | 1163 | 1166 | 597 | 7700 | 3498 | 1755 | 1162 | 1165 | 1178 | 595 | 1450 | 501 | 271 | 94 | 177 | 188 | 93 |
| Series (4 tags) | 1257 | 591 | | | | | | 1304 | 591 | | | | | | 436 | 94 | | | | | |
| Image (25 tags) | 7675 | 2974 | 1763 | 1183 | 1168 | 590 | | 8021 | 2941 | 1768 | 1174 | 1167 | 585 | | 1913 | 420 | 333 | 168 | 169 | 80 | |

**Figure 3. Query time, in milliseconds, used to search and retrieve tags. Queries were executed for each DICOM hierarchical level, retrieving individual tags and groups of tags.**

not stored as individual database fields. The economy in storage space and in storage time may be insufficient to justify the lack of query flexibility.

Built over the same underlying database schema than the originally configured RDB instance, the extended configured RDB instance uses the available BLOB fields as a workaround for the physical limitations on storing DICOM tags. It is effective in the sense of guaranteeing full-content storage; however, the approach does not reduce search and retrieval times due to the lack in indexing (as can be seen in Fig. 2 - left, when comparing the size of the index volumes between RDB instances). Assuming that the storage of a tag is performed once, and the search and retrieval of this same tag is performed whenever necessary, performance boosts for queries are at least desirable.

The adoption of schema-free or schema-flexible solutions is an alternative to the limitations of both RDB instances. Using an approach based on JSON documents, the search engine provides flexibility for both storage and query operations. Metadata acquired from radiological images are persisted and indexed in full, requiring a fraction of the space for storage and reducing significantly the time needed for content search. Another characteristic of such solutions, not explored in this work, is its native scalability; in scenarios like federated PACSs, high-demanding by nature, the adoption of extensible infrastructures – malleable enough to be reconfigured and to grow as needed – is differential. Here, the replacement of RDBMSs by less-constrained search engines turns out to be a strong option.

## 6. Conclusion

This work compares a search engine to relational database instances in the context of DICOM metadata management, observing its behavior in terms of storage space, storage time, and query time. RDB instances are configured to store partial- and full-content metadata extracted from a DICOM dataset; the search engine, in turn, indexes the same full-content metadata as JSON documents. The storage and query operations are executed similarly to DICOM C-STORE and C-FIND.

The acquired results attest the viability in adopting search engines as alternatives to relational databases in support for PACSs. Requiring less storage space and demanding a similar amount of time to store all alphanumerical tags when compared to an RDB instance, the evaluated search engine performed better in searching and retrieving individual tags and groups of tags for all DICOM hierarchical levels. Demanding 69% less storage space to a gain of 8.3 times in query response times, in the best case, the search engine outperforms the RDB instances in the majority of the experiments.

As future work, it is suggested to extend queries aiming to verify the index responses, both in relational database instances and in the search engine, to predicates involving intervals, lists of values, and patterns. It is suggested, also, the evaluation of the search engine deployed in a cluster, measuring the impact of distribution in storing and querying DICOM tags.

## Acknowledgment

This work was supported by Santa Catarina's State Health Department (SES/SC), CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), and FAPESC (Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina).

## References

Baeza-Yates, R. and Ribeiro-Neto, B. (2011). *Modern Information Retrieval - the concepts and technology behind search*. Pearson.

Bastiao Silva, L. A., Beroud, L., Costa, C., and Oliveira, J. L. (2014). Medical imaging archiving: A comparison between several nosql solutions. In *Biomedical and Health Informatics (BHI), 2014 IEEE-EMBS International Conference on*, pages 65–68.

Bidgood Jr, W. D., Horii, S. C., Prior, F. W., and Van Syckle, D. E. (1997). Understanding and using dicom, the data interchange standard for biomedical imaging. *Journal of the American Medical Informatics Association*, 4(3):199–212.

Clark, K. et al. (2013). The cancer imaging archive (tcia): Maintaining and operating a public information repository. *Journal of Digital Imaging*, 26:1045–1057.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Croft, B., Metzler, D., and Strohman, T. (2009). *Search Engines: Information Retrieval in Practice*. Pearson.

Elmasri, R. and Navathe, S. B. (2010). *Fundamentals of Database Systems*. Pearson.

Faggioni, L., Neri, E., Castellana, C., Caramella, D., and Bartolozzi, C. (2011). The future of pacs in healthcare enterprises. *European Journal of Radiology*, 78:253–258.

Huang, H. K. (2011). Short history of pacs. part i: Usa. *European Journal of Radiology*, 78:163–176.

Lemke, H. U. (2011). Short history of pacs. part ii: Europe. *European Journal of Radiology*, 78:177–183.

Mansoori, B., Erhard, K. K., and Sunshine, J. L. (2012). Picture archiving and communication system (pacs) - implementation, integration and benefits in an integrated health system. *Academic Radiology*, 19(2):229–235.

Mildenberger, P., Eichelberg, M., and Martin, E. (2002). Introduction to the dicom standard. *European Radiology*, 12(4):920–927.

National Electrical Manufacturers Association (2015a). Dicom ps3.4 2015c - service class specifications. Date last accessed 2016-01-31.

National Electrical Manufacturers Association (2015b). Dicom ps3.5 2015c - data structures and encoding. Date last accessed 2016-02-03.

Pianykh, O. S. (2012). *Digital Imaging and Communications in Medicine (DICOM) – A Practical Introduction and Survival Guide*. Springer.

Rascovsky, S. J., Delgado, J. A., Sanz, A., Calvo, V. D., and Castrillón, G. (2012). Informatics in radiology: Use of couchdb for document-based storage of dicom objects. *RadioGraphics*, 32(3):913–927.

Savaris, A., Härder, T., and Wangenheim, A. v. (2014). Dcmdsm: a dicom decomposed storage model. *Journal of the American Medical Informatics Association*, 21(5):917–924.

Singh, R., Chubb, L., Pantanowitz, L., and Parwani, A. (2011). Standardization in digital pathology: Supplement 145 of the dicom standards. *Journal of Pathology Informatics*, 2(1):23.

Sumathi, S. and Esakkirajan, S. (2007). *Fundamentals of Relational Database Management Systems*. Springer.

The Cancer Imaging Archive Team (2015). Lidc-idri. Date last accessed 2016-02-02.