DICOM Data Warehouse: Part 2

Steve G. Langer¹

Published online: 30 October 2015
© Society for Imaging Informatics in Medicine 2015

Abstract In 2010, the DICOM Data Warehouse (DDW) was launched as a data warehouse for DICOM meta-data. Its chief design goals were to have a flexible database schema that enabled it to index standard patient and study information, modality specific tags (public and private), and create a framework to derive computable information (derived tags) from the former items. Furthermore, it was to map the above information to an internally standard lexicon that enables a non-DICOM savvy programmer to write standard SQL queries and retrieve the equivalent data from a cohort of scanners, regardless of what tag that data element was found in over the changing epochs of DICOM and ensuing migration of elements from private to public tags. After 5 years, the original design has scaled astonishingly well. Very little has changed in the database schema. The knowledge base is now fluent in over 90 device types. Also, additional stored procedures have been written to compute data that is derivable from standard or mapped tags. Finally, an early concern is that the system would not be able to address the variability DICOM-SR objects has been addressed. As of this writing the system is indexing 300 MR, 600 CT, and 2000 other (XA, DR, CR, MG) imaging studies per day. The only remaining issue to be solved is the case for tags that were not prospectively indexed—and indeed, this final challenge may lead to a noSQL, big data, approach in a subsequent version.

Keywords Data mining · Databases · Computer systems

✉ Steve G. Langer
langer.steve@mayo.edu

¹ Mayo Clinic, Rochester, MN, USA

Introduction

DICOM has brought a very high level of standardization to medical images, allowing interoperability in many cases. However, there are still challenges facing the informaticist attempting to data mine DICOM images. Images from different DICOM epochs will encompass different levels of the standard. For example, consider the example of CT; dose information has gone from private to public tags (DLP and CTDIvol) and now Dose-SR. These examples all use the DICOM standard, but based on the year of manufacture, each device may implement the same information in different “standard” ways.

The situation is not just limited to the movement of a number with a given meaning to different tags. There is also the case that different vendors may put a value on the same DICOM tag, but it means different things to the different vendors. For example, take the seemingly simple case of defining the start time of a MR series. Many would assume the series time stamp would indicate this. And we would in many cases be wrong; it may not indicate the actual onset of image acquisition—it may simply indicate the time the scan protocol for that series was selected. In at least one case, the actual onset of image acquisition was indicated by a shadow tag related to the activation of the MR magnetic field gradient.

In 2010, the DICOM Data Warehouse (DDW) addressed these ambiguities and “roving value” problems by creating a database schema that builds a knowledge base for a given software version of each scanner [1]. This knowledge base tracks the locations of ill-behaved values and maps them to a consistent lexicon. Thus, any software programmer familiar with SQL (Structured Query Language) can write a query that compares the MR pulse sequence used on various scanners even if such are stored in different tags for different vendors by using the familiar “select pulse_seq from table Series

where Series.modality ilike MR”. Furthermore, “derived” tags can be computed (e.g., if an MR series is T1 or T2 weighted) and stored alongside the mapped meta-data.

Methods

From inception, the DDW architecture was carefully considered to enable a freely available, yet industry friendly, open source project. As such, not only its code but also the underlying technologies are freely redistributable under industry friendly open-source licenses. This is in contrast to related works that require licensed software [2].

To understand how the DDW works, it is important to understand the problem it solves. The challenge of a data warehouse can be reduced to four steps: acquisition, transformation, indexing, and reporting. Taken one by one, the components solving these tasks are (Fig. 1):

- Acquisition: Clinical Trials Processor [3].
- Transformation: MIRTH Connect (Mirth Corporation, Irvine CA).
- Indexing: PostgreSQL database (PostgreSQL Association, Portland OR).
- Reporting: either simple SQL queries or more complex methods (such as Ruby on Rails) can be used to support a web report front-end for non-programmer users.

It is worthwhile to consider the above processing pipeline steps one by one.

- Acquisition: In our work, we use CTP (Clinical Trials Processor at <http://mirewiki.rsna.org/index.php?title=Downloads>) as a DICOM receiver and buffer. This is not absolutely necessary. MIRTH channels can provide their own DICOM receiver, but the performance is slow, around 5 MR images per second. With CTP, DICOM objects can be written to the pipeline at more than 25 images/s. This allows sending scanners to dump their

objects quickly and disconnect their association, freeing their resources for another job (like sending to PACS).

- Transformation: At this point in the process, a MIRTH channel converts the binary DICOM object header into an XML tree. Universal tags (i.e., patient and study meta-data, common to all modalities) are immediately harvested and written to the database. The great innovation of DDW occurs at this point. The MIRTH channel knows nothing special about the modality; it does not know MR tags from DR tags. It gets this data by querying the database on the software version of the object it is currently processing. The database in turn has a knowledge base for that software version and returns to MIRTH what special modality tags (public or private) it should also harvest. We shall see below how this knowledge base is created. The power of this concept is that MIRTH can run continuously with one channel, which processes any type of object. Adding a new scanner consists only of adding new rows (not tables or columns) to the database that instructs the channel what to map at the study, series, instance, and (if needed) acquisition level. These new items are picked up by the channel automatically the next time it processes an object. Neither the MIRTH channel nor the DDW schema has been changed in several years.
- Indexing: The normal study hierarchy of DICOM is evident in the DDW schema. A study has one or more series which have one or more instance objects. Since there are also mapped and derived values, there are actually three tables at each level. Hence, there is a “study, study_mapped” and “study_derived” triplet of tables. The same pattern is repeated for series and instance. As the MIRTH channel runs, it populates all the base and mapped tables. The values for the derived tables are deferred until the associated stored procedures for a given scanner run and compute them.
- Reporting: To date, a library of over 30 SQL queries has been developed to mine DDW data. Some examples were as follows: scanner software versions, detecting a scanner version change, reports of what exams (accessions) were

Fig. 1 The four conceptual steps in a functional data warehouse. DICOM objects (images, SR (Structured Reports), MPPS (Modality Performance Procedure Steps), etc.) are received by CTP (Clinical Trials Processor) and spooled to a processing folder. That folder is watched by a MIRTH pipeline that intakes the object and converts the header to an XML

(eXtensible Markup Language) tree. In turn, that tree is harvested for common and mapped tags, which are then inserted into PostgreSQL. Reports from external SQL (Structured Query Language) queries then mine results

Table 1 The common base of tags that are processed for every scanner. They include information about the study itself (the various UIDs), patient information, and basics about the study (i.e., its description and accession)

DICOM tag	DDW name	DICOM tag	DDW name
00100010	patientsName	00080033	contentType
00100020	patientId	00080090	referringPhysiciansName
00100030	patientsBirthDate	00080080	institutionName
00100040	patientsSex	00081070	operatorsName
00180015	bodyPartExamined	00081060	nameOfPhysiciansReadingStudy
00101010	patientsAgeAtStudy	0020000D	studyInstanceId
00101020	patientsSize	00080018	sopInstanceId
00101030	patientsWeight	0020000E	seriesInstanceId
001021C0	pregnancyStatus	00080016	sopClassUid
00080060	modality	00081030	studyDescription
00080070	manufacturer	0008103E	seriesDescription
00081090	manufacturersModelName	00200010	studyId
00181020	softwareVersions	00200011	seriesNumber
00020016	sourceApplicationEntityTitle	00200012	acquisitionNumber
00081010	stationName	00200013	instanceNumber
00080050	accessionNumber	00181030	protocolName
00080020	studyDate	00080008	imageType
00080030	studyTime	00185101	viewPosition
00080031	seriesTime		
00080032	acquisitionTime		

done with experimental protocols, modality counts by day/device/location, CTDIvol by device, and so on. One conversant in Ruby on Rails could readily develop a web front-end to expose these and other reports to non-programmers, but, to date, this has not been done.

We alluded above to the process by which the knowledge base is created per software version. The full details of the database schema can be found in the earlier work or the source code [4]. For our purposes, it suffices to know that there exists

a common list of tags that are the same for every scanner (Table 1).

When one encounters a hitherto unknown scanner, the following process is followed to create a knowledge base entry for that scanner software version. First, one checks if a new term is needed in the table “dict_standard_terms” (e.g., CTDIvol) to define a new term. Second, one manually adds modality specific tags to the “mapped_tags” table for that version. Third, one defines computable tags for the “derived_tags” table and writes stored procedures that will

Table 2a For the case of MR scanners, this table shows the additional header tags that the MIRTH channel will be told to map

DICOM tag	DDW name	DICOM tag	DDW name
0019109c (00291020)	Pulse_seq	00180087	Field_strength
00101030	Weight	00181314	Flip_angle
001910f9 (0019105a)	Transmit_gain	00200037	Image_orientation
00180050	Slice_thickness	00180088	Interslice_space
00181316	SAR	00180082	Inversion_time
00180080	Repetition_time		
0019108a	Receive_gain		
0019105a	Acquisition_duration		
00181250 (00291020)	coil		
00180010	Contrast_agent		
00180081	Echo_time		

Sometimes, all vendors agree on the same tag, other times they do not, as evidenced by the multiple tag entries for that row (i.e., an additional tag in () indicates the variations among vendors)

Table 2b For the case of CT and MR scanners, this table shows the additional tags that are derived from the prior information (common or mapped tags)

Source data for stored procedure	DDW name to derived data tables
0040a730	Dlp, ctdi_vol
00180080, 00180081	Weighting (T1 or T2)

For example, CTDIvol and DLP (Dose Linear Product) values for CT scanners (at our institution) are parsed out of the mapped tag “0040a730”

compute these values. By joining the common tags (seen above) with the mapped and derived tags (seen below) for a given scanner type, one can define the knowledge base entry for the scanner.

It may not be obvious from above, but in the case of Table 2a, values will be stored into those rows after MIRTH processes a given DICOM header because the existing header value is mapped to it. In the case of Table 2b, those rows will be empty until the stored procedure that computes that value is run. If a new scanner version comes along that is known to use the same mappings as an earlier version, a stored procedure exists to clone the settings of the old scanner to the new one.

As an example of how the DDW schema enables a single query to render a report for CT scanners whose CTDIvol and DLP information is stored in different locations, consider the following SQL statement:

```
select distinct
accession,
exam_descrip,
station_id,
vendor,
model,
std_name,
value,
unit
from
exams_derived_values,
exams,
series
```

where

```
(exams_derived_values.std_name ilike
'dlp%' OR exams_derived_values.std_name
ilike 'ctdi%') AND
exams.exam_uid =
exams_derived_values.exam_uid AND
exams.exam_uid=series.exam_uid
order by station_id, accession
```

The associated output is shown in Table 3. The important point to note is that there is no vendor specific “secret sauce” in the above query. Yet, it is able to create a report that spans three vended CTs using different Dose-SR representations (the true vendor names have been obscured for privacy reasons). The data for this was obtained in two steps: first, the vendor’s Dose-SR was stored in the database by the mapping operations of MIRTH. Then, stored procedures in PostgreSQL parsed the Dose-SR of each vendor to find the values and map them to the proper “derived” table under the standard name “dlp” or “ctdi_vol.”. And for the interested reader who wishes to peruse the MIRTH channel at github, there is no vendor “secret sauce” in it either. The knowledge base within DDW tells the MIRTH channel how to map non-universal tags, and the stored procedures in DDW convert mapped values into derived ones; all of this is done under the standard internal lexicon of DDW. Thus, the report author need only concern themselves with SQL queries based on that lexicon. The mechanics of DICOM are utterly invisible to them.

Discussion

At this point in time, DDW at our institution is indexing about 3000 studies every weekday. Of those, 300–400 are MR, 600 are CT, and the balances are CR/DR, XA, MG, and some NucMed (US and some NucMed studies are reachable, yet). Furthermore, not all modalities (e.g., MG) have had their stored procedures fully developed, yet. This load amounts to about two million DICOM objects per day. A single MIRTH instance was insufficient to index a day’s load in one day, so a

Table 3 The output of a DDW query for the CTDIvol and DLP of CTs at the author’s institution

accession	exam_descrip	station_id	vendor	model	std_name	value	unit
16257639-1	CT abdomen w+pelvis w	CT3R	A	A1	ctdi_vol	64.63	mGy
16257639-1	CT abdomen w+pelvis w	CT3R	A	A1	dlp	2435.97	mGy*cm
16642473-1	CT abdomen w+pelvis w	CT3Y	B	B1	ctdi_vol	39.97	mGy
16642473-1	CT abdomen w+pelvis w	CT3Y	B	B1	dlp	1334.63	mGy*cm
16568946-1	CT abdomen w+pelvis w	CT3Z	C	C1	ctdi_vol	20.63	mGy
16568946-1	CT abdomen w+pelvis w	CT3Z	C	C1	dlp	1023.16	mGy*cm

This sub-sample of the report compares outputs for a “CT Abdomen and Pelvis” on each of three different vended scanners of different software vintages. The CTDIvol and DLP for the same study on a given scanner are shown together as demonstrated by the same study accession. The actual names of the vendor/model have been aliased for privacy

load balanced cluster has been implemented consisting of 12 cores and 24 GB of RAM. The knowledge base knows 98 software versions comprising: eight modalities (CT, MR, PET, XA, MG, CR, DR, PET-CT), eight vendors, and over 20 models of equipment. Over 100 terms are defined to map modality specific or derived tags to a standard lexicon for query building.

There are several audiences for DDW reports. Researchers use it to track the use IRB approved research scanner protocols or coils (in the case of MR). They also use to find candidate studies that match the inclusion criteria for retrospective IRB work. For example, if a researcher desires to compare the diagnostic efficacy of competing coils or pulse sequences, DDW can list out the accessions of studies using those devices, which then can be used on billing/reporting systems for ICD9 codes to determine if there is a differential trend in reporting the disease of interest. Administrators like it for determining room usage patterns. Bioengineers like it to see the software versions of the deployed equipment. And physicists use DDW to compare/chart safety metrics like CTDIvol or SAR. To date, these reports are run manually by the developer, but since the SQL reports already exist, it would be a moderate exercise to expose them via a web site.

Where DDW falls short is if a report is desired for data tags that were not indexed, prospectively. There are two ways to address this in a relational database like PostgreSQL. The first method is to add new tags to index and re-run studies through DDW, and the new tags will index to the mapped and derived tables as they are processed. The second method is to archive the XML transform of the entire header and then if required, augment a stored procedure to parse out the new tags and re-run the stored procedure. This option was actually trialed, but seeing the DDW grows 40 GB every night gives one pause [5]. Thus, as this is written, a week's worth of XML transformed headers are being sent to a big data appliance (TeraData Corp., Miamisburg, OH). The advantages are

several: the system is built to scale, retrospective map-reduce functions are supported, and grander data patterns can be mined via cross-queries to the EMR data. Others have trialed this path for MR [6]. The value we hope to add is the power of a multi-modality lexicon for simplified report development.

Addendum

Over the past couple of years, people who have downloaded the code from the github site have commented that there is a very steep learning curve to installing and configuring all the DDW components. In an effort to address this, the author has built a turnkey DDW database and DDW MIRTH gateway in the form of Dockers. Look here "<https://hub.docker.com/u/sglanger/>" for updates on this project.

References

1. Langer SG: A flexible database architecture for mining DICOM objects: the DICOM data warehouse". *JDI* 25(2):206–212, 2012
2. Wang S, Pavlicek W, Roberts CC, Langer SG, Zhang M, Hu M, Morin RL, Schueler BA, Wellnitz CV, Wu T: An automated DICOM database capable of arbitrary data mining (including radiation dose indicators) for quality monitoring. *JDI* 24(2):223–33, 2011
3. Langer SG, Tellis W, Carr C, Daly M, Erickson BJ, Mendelson D, Moore S, Perry J, Shastri K, Warnock M, Zhu W: The RSNA image sharing network. *JDI* 28(1):53–61, 2015. doi:10.1007/s10278-014-9714-z
4. DDW at <https://github.com/sglanger/ddw> . last viewed June 2015.
5. Langer SG: Challenges for data storage in medical imaging research. *J Digit Imaging* 24(2):203–7, 2011. doi:10.1007/s10278-010-9311-8
6. Rascovsky SJ, Delgado JA, Sanz A, Calvo VD: Castrillón G. Informatics in radiology: use of CouchDB for document-based storage of DICOM objects. *Radiographics* 32(3):913–27, 2012. doi:10.1148/rg.323115049