ImgDW Generator[1]: A tool for generating data for medical image data warehouses

Guilherme Muzzi da Rocha[1], Cristina Dutra de Aguiar Ciferri[1]

[1]Departamento de Ciências de Computação – Universidade de São Paulo
Caixa Postal 668 – 13.560-970 – São Carlos – SP – Brasil

guilherme.muzzi.rocha@usp.br, cdac@icmc.usp.br

Abstract. *In this paper, we introduce ImgDW Generator, a tool that generates synthetic data for populating medical image data warehouses designed according to the relational technology. The tool supports different star schemas for the image data warehouse and offers a graphical interface that assists users to manipulate these schemas. ImgDW Generator can be used to generate data aiming at different scenarios of performance evaluation.*

1. Introduction

Data warehousing environments (DWEs) play a key role in decision-making [Chaudhuri et al. 2011]. They support an ETL (extract-transform-load) process aimed to extract, transform, clean, integrate, and load data from heterogeneous sources into large databases called data warehouses (DWs). In addition to these characteristics, data in the DWs are subject-oriented, non-volatile, and referring to large periods of time. In relational implementations of DWs, data are usually modeled through a star schema, where a central fact table is linked to several satellite dimension tables. DWEs also support OLAP (on-line analytical processing) queries, which are complex analytical queries aimed to discover useful trends and patterns. Conventional DWEs only manage conventional data, such as alphanumeric and numeric types.

Image DWEs extend conventional DWEs to deal with images. Because there is no agreement in the literature about the definition of these environments, we use the principles introduced by Teixeira et al. (2015). Instead of managing images as matrices of pixels or files in the DICOM (Digital Imaging and Communications in Medicine) format, images are represented through their intrinsic features, i.e. *feature vectors* and *attributes for similarity search*. As a result, image DWEs should support an extended ETL process that also performs the extraction of features from images, an extended DW (i.e. an *image DW*) that also stores images features in fact or dimension tables, and an extended query processing that enables OLAP similarity queries.

Image DWEs empower decision-making by enabling users to issue a new range of queries, which integrate the conventional OLAP and the similarity search processes. For instance, consider an application related to the medical field that stores images of exams in addition to conventional data related to patients and years. Using these environments, specialists are able to issue queries such as "How many images are similar to a given cancer image, considering patients with ages between 40 and 50, and years from 1992 to 2018?". Assessing the performance of this new range of queries is

[1] https://www.lsec.icmc.usp.br/medicalimage. The authors thank the financial support of the following Brazilian agencies: CAPES, CNPq, FAPESP and FINEP.

not an easy task. It is very difficult to generate image data and to relate them with conventional data.

This paper introduces ImgDW Generator, a tool for generating data for medical image DWs implemented according to the relational technology. We have designed ImgDW Generator to provide the major characteristics as follows.

- It offers a graphical and interactive interface.
- It supports four different star schemas to model the medical image DW.
- It generates conventional and image data related to the medical field to populate the medical image DW.

To the best of our knowledge, there is no other tool that generates intrinsic features of images and relates them with conventional DW data. This paper is organized as follows. Section 2 describes background, Section 3 details ImgDW Generator, Section 4 shows configurations for performance evaluation, and Section 5 concludes the paper.

2. Background

The computational management of images requires the construction of an image descriptor, which is described by two aspects [Traina et al. 2007]. The first one is an extraction algorithm that encodes image features into feature vectors. Common image features are attributes of color and texture. Well-known image extractors for these attributes are Color Histograms [Gonzalez and Woods 2006] and Haralick descriptors [Haralick 1979]. The feature vectors generated by these extractors contain the numeric representations of the images, and are usually represented in the metric space. Thus, the second aspect refers to a distance function, which calculates the dissimilarity between two images based on their feature vectors. A distance function becomes smaller as the images become more similar, thus enabling the execution of similarity search.

Environments that manage large volumes of image data should improve similarity search performance by pruning portions of the database where a given query image cannot be found. To this end, the principles introduced by Teixeira et al. (2015) use the Omni-technique to select strategically positioned images from the dataset, called representative images [Traina et al. 2007]. The number of representative images depends on the intrinsic dimensionality of the dataset according to the applied image extractor. For instance, Color Histograms may require three representative images.

Figure 1 depicts four different star schemas that may be modeled to store medical images in an image DW, according to the results described by Annibal (2011). The fact table (e.g. Exam) contains foreign keys to the dimension tables and numeric measures; conventional dimension tables (e.g., Age) store a primary key and several descriptive conventional data; and image dimension tables (e.g., Histogram) store a primary key and distances to representative images. The schemas differ on how they manage the storage of the feature vectors, as described as follows: (i) *attributes of a joint dimension table* (Figure 1a): feature vectors are stored as attributes in a unique dimension table, which may contain feature vectors related to distinct image dimension tables; (ii) *attributes of a single dimension table* (Figure 1b): feature vectors are stored as attributes in a single dimension table, which cannot contain feature vectors related to distinct image dimension tables; (iii) *facts in the fact table* (Figure 1c): feature vectors are stored as numeric measures in the fact table, which may contain feature vectors related to distinct image dimension tables; and (iv) *attributes of an image dimension table* (Figure 1d): features vectors are stored together with their respective image dimension table.

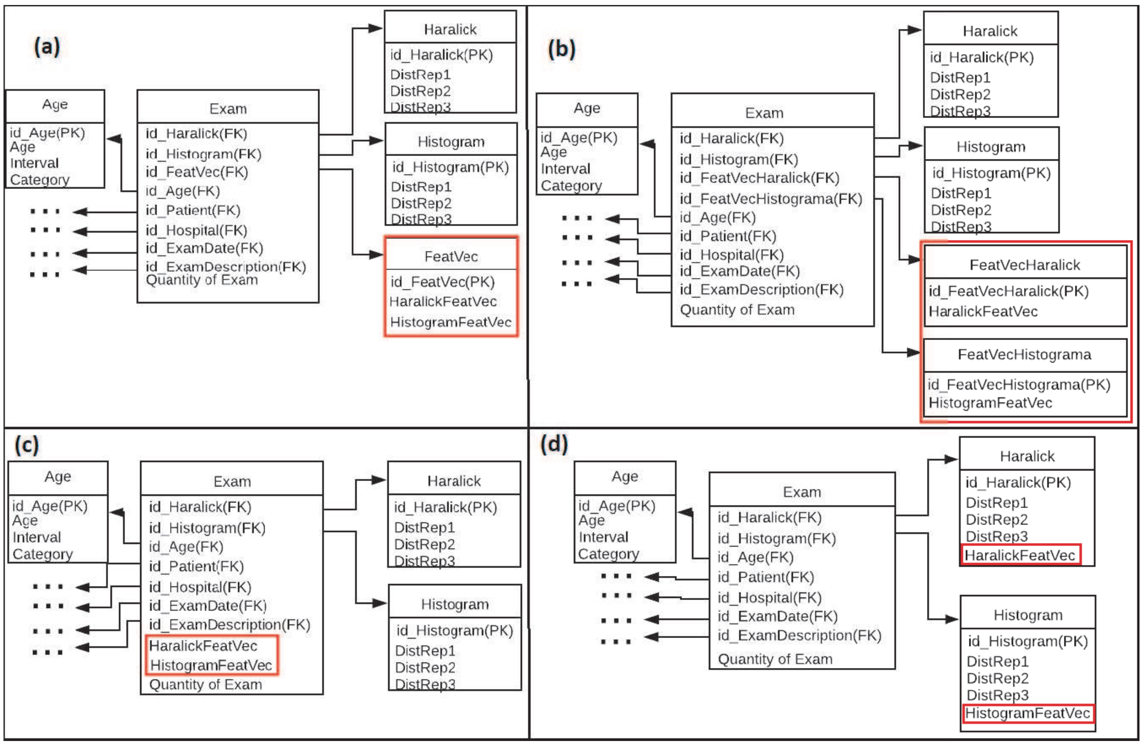


Figure 1. Four different star schemas to model a medical image DW [Annibal 2011]: (a) *attributes of a joint dimension table*; (b) *attributes of a single dimension table*; (c) *facts in the fact table*; and (d) *attributes of an image dimension table.*

3. ImgDW Generator

In this section, we introduce ImgDW Generator, a tool for generating data for medical image DWs implemented according to the star schemas depicted in Figure 1. The tool considers as **default application** a medical application that models the fact *Quantity of Exams*, considering the conventional dimensions *Age*, *Patient*, *Hospital*, *ExamDate*, and *ExamDescription*, and the image dimension tables *Histogram*, *Haralick1*, *Haralick2*, *Haralick3*, and *Haralick4*. Different Haralick descriptors allow analyzing different texture features of images [Haralick 1979].

Using the initial interface of ImgDW Generator, the user can set **configurations parameters**, such as the directory where data will be saved, and choose the star schema to be used. Because the tool works similarly for each star schema, here we consider that the user has chosen the schema *attributes of a joint dimension table* (Figure 1a) to show the major characteristics of ImgDW Generator.

Based on the user's option, ImgDW Generator displays the interface shown in Figure 2. The fact table is visually drawn in the center, conventional dimension tables are drawn in green on the left, and image dimension tables are drawn in blue on the right. The dimension table that stores the feature vectors is represented in red on the right. The user may iterate with the tool by choosing one of the following tags: generate data (Section 3.1) and generate create/insert script (Section 3.2).

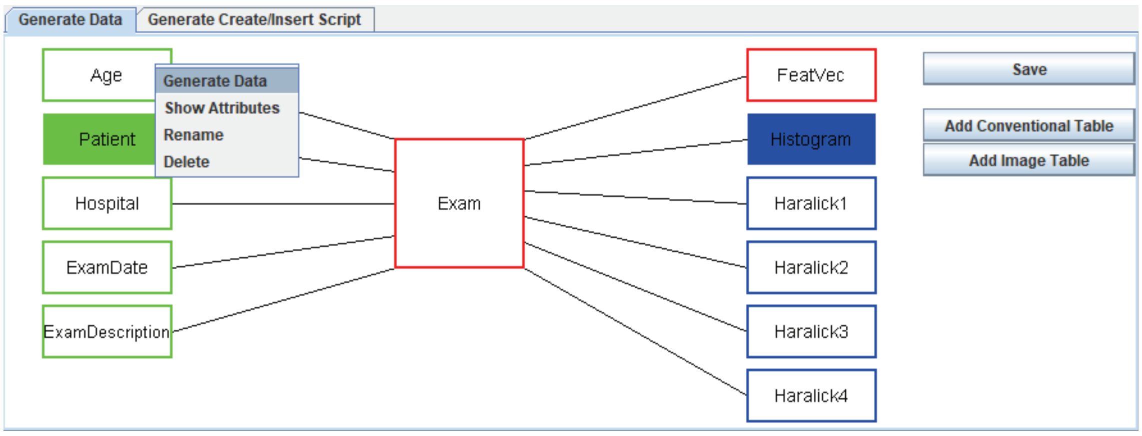


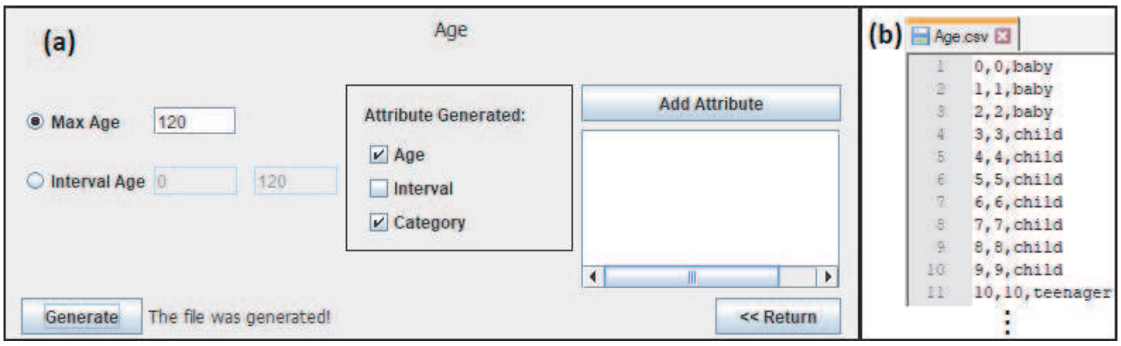
Figure 2. Interface of the tag Generate Data, considering the schema *attributes of a joint dimension table.*

3.1. Tag Generate Data

Tag Generate Data is used to generate synthetic data to populate the fact table, the conventional dimension tables, and the image dimension tables.

When the user chooses a given table, she can generate data, show its attributes, rename the table, and delete the table (Figure 2). Figure 3a depicts data generation for the conventional dimension table Age. The user can set the number of tuples to be generated by defining a maximum value or an interval of values, select the attributes to be generated, and add new attributes as needed. For each new attribute, the user should name it and select an external text file that contains data values for its domain. Figure 3b shows data generated for the table Age, which are stored in a CSV file. Similarly, Figure 4 shows an example of data generation for the image dimension table Histogram.

Data generation should follow an order: *first*, populate the conventional and image dimension tables; *second*, populate the feature vectors table; *third*, populate the fact table. ImgDW Generator visually illustrates that a table has been populated as follows. Before being populated, only the border of the table is colored (see the dimension tables Age and Haralick1 in Figure 2). After being populated, the table is fully colored (see the dimension tables Patient and Histogram in Figure 2).



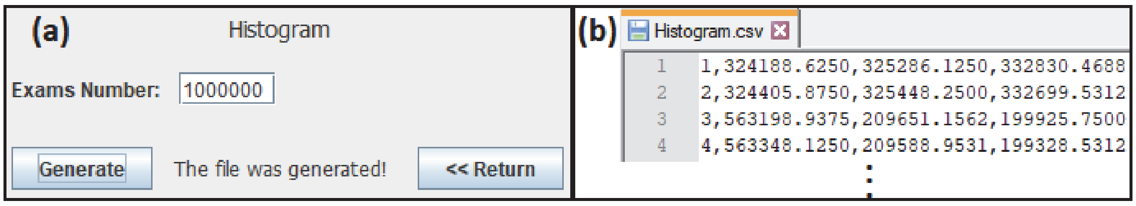
[✓ Max Age] 120
Interval Age 0 120

Attribute Generated:
[✓ Age]
Interval
[✓ Category]

The file was generated!

```
0,0,baby
1,1,baby
2,2,baby
3,3,child
4,4,child
5,5,child
6,6,child
7,7,child
8,8,child
9,9,child
10,10,teenager
⋮
```

Figure 3. Example of data generation for the conventional dimension table Age: (a) interface with options; (b) generated CSV file.



Exams Number: 1000000

The file was generated!

Histogram.csv

```
1,324188.6250,325286.1250,332830.4688
2,324405.8750,325448.2500,332699.5312
3,563198.9375,209651.1562,199925.7500
4,563348.1250,209588.9531,199328.5312
          ⋮
```

Figure 4. Example of data generation for the image dimension table Histogram: (a) interface with options; (b) generated CSV file.

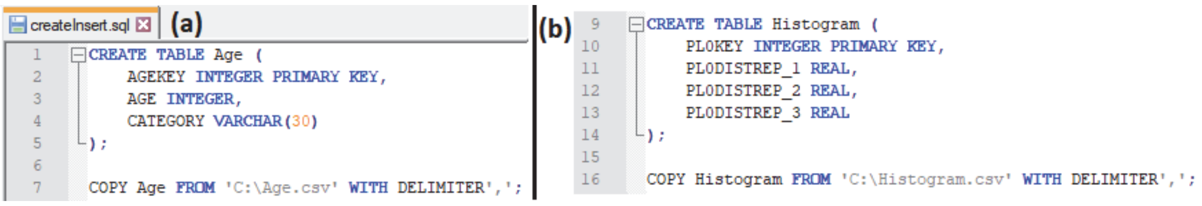
At any time, the user can add new conventional and image dimension tables (Figure 2). To this end, she should define the type of the table, its name and its attributes. For each attribute, she also should select an external text file that contains the domain of values of the attribute. That is, the user can use specific data sets to populate tables. Another available functionality is saving the work.

For the default application, ImgDW Generator already encompasses external text files that contain data to populate each table. Although the tool generates synthetic data, it provides real data when possible. Data that populate the conventional dimension table Hospital are real and were obtained from the Brazilian health care system available at www2.datasus.gov.br/datasus/index.php. Data that populate the conventional dimension table ExamDate are also real, and refer to days from 1992 to 2018. Furthermore, image data were generated using feature vectors obtained from 1,000,000 real images from a Brazilian public hospital. The remaining dimension tables are populated with synthetic data mainly due to privacy issues. Regarding the fact table, it associates each image to its conventional data. It is a factless fact table since it contains as numeric measure the artificial attribute *quantity of exams*, which is always populated with the value of 1.

3.2 Tag Generate Create/Insert Script

Tag Generate Create/Insert Script is used to generate SQL commands to create the dimension and fact tables and to insert data into these tables. This tag is only available to the user when all tables of the star schema are populated.

The definition of the CREATE TABLE commands follows the characteristics of the star schema defined in the tag Generate Data, respecting the tables and attributes selected by the user. The insertion of data is performed using the COPY command and the generated CSV files. Figure 5 depicts the create/insert scripts for the conventional dimension table Age and the image dimension table Histograms. The directory and the delimiter specified in the COPY command are set in the configuration parameters. In its current version, ImgDW Generator is set to generate SQL scripts for PostgreSQL®.



```
CREATE TABLE Age (
    AGEKEY INTEGER PRIMARY KEY,
    AGE INTEGER,
    CATEGORY VARCHAR(30)
);

COPY Age FROM 'C:\Age.csv' WITH DELIMITER',';
```

```
CREATE TABLE Histogram (
    PL0KEY INTEGER PRIMARY KEY,
    PL0DISTREP_1 REAL,
    PL0DISTREP_2 REAL,
    PL0DISTREP_3 REAL
);

COPY Histogram FROM 'C:\Histogram.csv' WITH DELIMITER',';
```

Figure 5. Create/insert scripts: (a) conventional dimension table Age; (b) image dimension table Histogram.

Table 1. Number of tuples generated by ImgDW Generator for scenarios 1 and 2.

Scenario 1				Scenario 2	
Tables	# tuples	Tables	# tuples	Tables	# tuples
Age	121	Histogram	1,000,000	Age	121
Patient	100,000	Haralick1	1,000,000	ExamDate	4,934
Hospital	645	Haralick2	1,000,000	Exam	500,000
ExamDate	9,868	Haralick3	1,000,000	Histogram	500,000
ExamDescription	1,000,000	Haralick4	1,000,000	Haralick1	500,000
Exam	1,000,000	FeatVec	1,000,000	FeatVec	500,000

4. Configurations for Performance Evaluation

ImgDW Generator can be used to generate data aiming at different scenarios of performance evaluation. For instance, consider the following two scenarios: (i) *scenario 1*, which is composed of all tables shown in Figure 2; and (ii) *scenario 2*, which is composed of only some of these tables. Table 1 depicts the tables and their respective number of instances (i.e. # tuples) for each scenario. Performance evaluation tests should use these medical image data warehouses to issue SOLAP similarity queries over them considering different dimensionalities and data volumes. Performance evaluation considering these scenarios can be found in Annibal (2011) and Teixeira et al. (2015).

5. Conclusions

In this paper, we introduce ImgDW Generator, a tool for generating data for medical image data warehouses implemented according to the relational technology. The tool offers a graphical and interactive interface through which users can work with different star schemas that model the medical image data warehouse; set conventional and image dimension tables, attributes and values for these attributes; and generate SQL scripts containing commands for creating and populating the data warehouse. ImgDW Generator was implemented in Java using NetBeans version 8.2, thus it is portable. We are currently extending the tool to support different database management systems. We are also developing new functionalities to assist users to define SQL commands that perform OLAP similarity queries over medical image data warehouses.

References

Annibal, L.P. (2011) Istar: Um Esquema Estrela Otimizado para Image Data Warehouses Baseado em Similaridade. Dissertação de Mestrado, UFSCar.

Chaudhuri, S., Dayal, U., Narasayya, V.R. (2011). An Overview of Business Intelligence Technology. Communications of the ACM, v. 54, n. 8, p. 88-98.

Gonzalez, R.C., Woods, R.E. (2006). Digital Image Processing. Prentice-Hall, 3rd ed.

Haralick, R.M. (1979). Statistical and Structural Approaches to Texture. Proceedings of the IEEE, v. 67, n. 5, p. 786-804.

Teixeira, J.W., Annibal, L.P., Felipe, J.C.; Ciferri, R.R.; Ciferri, C.D.A. (2015) A Similarity-based Data Warehousing Environment for Medical Images. Computers in Biology and Medicine, v. 66, p. 190-208.

Traina Jr., C., Santos Filho, R.F., Traina, A.J.M., Vieira, M.R., Faloutsos, C. (2007) The Omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient, The VLDB Journal. v. 16, n. 4, p. 483-505.