Published in final edited form as:
*Decis Support Syst*. 2013 June 1; 55(3): 728–739. doi:10.1016/j.dss.2013.02.008.

# Methodological Guidelines for Reducing the Complexity of Data Warehouse Development for Transactional Blood Bank Systems

**Pedro L. Takecian**[a,1,*], **Marcio K. Oikawa**[b], **Kelly R. Braghetto**[a], **Paulo Rocha**[d], **Fred Lucena**[e], **Katherine Kavounis**[g], **Karen S. Schlumpf**[g], **Susan Acker**[g], **Anna B. F. Carneiro-Proietti**[d], **Ester C. Sabino**[c], **Brian Custer**[f], **Michael P. Busch**[f], and **João E. Ferreira**[a]

[a]University of São Paulo, Inst. of Math. and Statistics, São Paulo-SP, Brazil

[b]Center of Mathematics, Computing and Cognition, Federal University of ABC

[c]Pró-Sangue Foundation, São Paulo Blood Center, São Paulo-SP, Brazil

[d]Hemominas Foundation, Minas Gerais, Brazil

[e]Hemope Foundation, Recife, PE, Brazil

[f]Blood Systems Research Institute, San Francisco, CA, USA

[g]Westat, Rockville, MD, USA

## Abstract

Over time, data warehouse (DW) systems have become more difficult to develop because of the growing heterogeneity of data sources. Despite advances in research and technology, DW projects are still too slow for pragmatic results to be generated. Here, we address the following question: *how can the complexity of DW development for integration of heterogeneous transactional information systems be reduced?* To answer this, we proposed methodological guidelines based on cycles of conceptual modeling and data analysis, to drive construction of a modular DW system. These guidelines were applied to the blood donation domain, successfully reducing the complexity of DW development.

### Keywords

modular architecture; conceptual modeling; heterogeneous systems; blood donation process; data warehouse; data analysis

## 1. Introduction

Data warehouse (DW) projects have become increasingly large and difficult to manage because of the growing use of analytical systems in which the data come from heterogeneous transactional systems. Usually, DW projects involve staff with different profiles, technological backgrounds and internal assignments within their institutions.



[*]Corresponding Author: plt@ime.usp.br (Pedro L. Takecian).
[1]Correspondence: Instituto de Matemática e Estatística - Universidade de São Paulo. Rua do Matão, 1010, Cidade Universitária, São Paulo - SP. CEP: 05508-090 - Brazil. PHONE#: +55 11 30916172. FAX#: +55 11 30916102.

Because of these factors, many DW projects are slow to deliver effective results and often are not completed. According to a study by the Gartner Group in 2005 [12], about 50% of DW projects tend to fail due to problems during DW design and construction. One of the most important causes is the long development time, which leads to delays in delivery of functional features to the end user. Often, when DW systems are finally available, some of the features implemented are already obsolete, while newer needs end up being postponed until future phases of development.

According to Johnson [20], even when these large projects are successful, their usage rate is low, with less than 40% of implemented functions in use. This occurs because of two main problems. The first one is the lack of synchronization between the development team and end users of a DW (e.g., physicians, researchers or businessman). Due to the extended development time on large projects, the alignment between user needs and the final solution implemented is adversely affected. The other problem is that the traditional process for DW construction does not allow rapid and partial deliveries of functional features. Usually, projects will only be available to end users after they have been fully implemented, which can take months or years. After these lengths of time, it is likely that their data will already be out of date and no longer relevant.

This paper addresses the following question: *how can the complexity of DW development for integration of heterogeneous transactional information systems be reduced?* To answer this question, we proposed methodological guidelines for construction of DW based on cycles of conceptual modeling and data analysis, to drive creation of a modular DW system. Trying to improve these guidelines through a real usage experience, we have applied them in the *Retrovirus Epidemiology Donor Study - II* (REDS-II) project, reducing the DW development complexity.

The REDS-II project is sponsored by the National Heart, Lung and Blood Institute (NHLBI) of the United States and consists of a network of blood centers in that country, formed with the purpose of developing research projects focusing on blood safety. In 2005, the United States National Institutes of Health (NIH) opened a request for proposal (RFP) to support foreign blood center participation in this network. Three large Brazilian blood centers were included: Pró-Sangue Foundation / Hemocentro de São Paulo, Hemominas Foundation / Hemocentro de Minas Gerais, in the Southeastern region of Brazil, and Hemope / Hemocentro de Pernambuco, in the Northeast. In this project, our work was to develop a DW to compile blood centers' routinely collected data from their transactional systems. Although these blood centers were able to store data, they were unable to analyze their data because of lack of appropriate information systems with the capacity to analyze large datasets. These centers generate around 400,000 screenings per year from 220,000 candidates with 350,000 donations, and 2.5 million test results.

This paper is organized as follows. Section 2 presents some fundamentals and approaches within the field of DW system development. Section 3 describes the concepts and practices used in constructing the REDS-II DW, and presents our methodological guidelines for reducing development complexity. Section 4 presents some discussion about the proposed guidelines, and Section 5 gives our conclusion and describes our ongoing steps.

## 2. Fundamental Concepts and Related Works

Starting in the late 1970s, the growing success of database management systems (DBMSs) was responsible for popularizing the use of databases in organizations around the world. This success was mainly due to the introduction of the SQL declarative language in DBMSs, which facilitated handling, maintenance and recovery of stored data, and to the use of relational databases. In DBMSs, the databases were (and still are) designed to store data

coming from companies' routine transactions, being optimized for this purpose. Transactional databases, as they are known, are widely used in various business sectors, and the use of DBMSs for this purpose has become a *de facto* standard. Over the same period, with increasing competition between companies and the constant search for improvements in production processes, the need to obtain analytical and managerial views of the data stored in databases has been intensified. However, transactional databases are not, on their own, suitable for providing views that assist in decision-making, because they are not modeled for this purpose. Therefore, this period of time was marked by successive and progressive attempts by companies and researchers to find appropriate solutions for developing analytical databases. In the early 1990s, the term *data warehouse* (DW) was coined by William (Bill) Inmon [18]. He defined it as "a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision-making process".

Currently, in the available literature, there is no precision regarding DW concept. In some more recent works, it is noticeable that the scope of the term has been increasing, such that it now includes not only data collections, but also support systems for extraction and preparation of data that will compose these collections. It is this broader sense of DW that will be used throughout the present work. Thus, a DW can be interpreted as a system that is designed with the purpose of supporting "efficient (data) extraction, processing and presentation for analytic and decision-making purposes" [11]. With the same scope, Rainardi [24] defined DW as "a system that retrieves and consolidates data periodically from the source systems into a dimensional or normalized data store".

There are several approaches towards building a DW, which result from combinations, of differing degrees, of the approaches advocated by Inmon and Kimball [21]. According to Inmon, an organizational DW should have a single data repository, in which the data should be stored in a normalized manner. A normalized database [11] is organized so as to avoid data redundancy as much as possible. This data repository is called a NDS (*Normalized Data Store*). From the NDS, data subsets specialized in specific subjects can be extracted, denormalized (data redundancy is introduced into databases to improve the speed of data retrieval) and indexed, giving rise to analytical databases (known as *data marts*) that will serve the analytical needs of the organization. According to Kimball, however, the DW of an organization should be assembled as a conglomerate of data marts, developed to meet specific requirements. In this approach, the data are always stored in analytical denormalized databases, using a dimensional model.

In database modeling, a multidimensional model is developed when efficiency in retrieving data from analytical queries is sought. However, for efficiency to be achieved, the model must introduce data redundancy into the database, and this process culminates in a need for more space for data storage and increased difficulty in ensuring database integrity. The database multidimensional modeling is basically composed of two kinds of concepts: facts and dimensions. A fact is a measure of interest for analysis. This measure should be stored at an appropriate scale of observation for the intended purposes. In a blood center, for example, a fact of interest could be the number of blood donations within one day. A dimension is a context of interest related to a fact. As an example, the months in which the donations occurred can be considered to be a dimension related to the number of donations. Since facts will usually be analyzed from the perspectives of different contexts, the model can be called multidimensional. A multidimensional model can be implemented using a relational DBMS. The two most widespread types of implementation are the *star schema* model and the *snowflake schema* model. In both models, the facts of interest are stored in fact tables, containing measured attributes and links to dimension tables. For any given

dimension, the possible values are stored in one (or more) dimension table(s), which can be joined to fact tables as required by the analyses.

In the current scenario, there are opportunities for the use of all these approaches, depending on the goal to be reached through construction of the DW. Many organizations are closer to the idea developed by Kimball, because DWs often arise from the union of several dimensional databases developed by departments in independent efforts. Additionally, some database writers say that they prefer a fully dimensional approach because this model is simpler for end users to understand and interpret. However, this claim has been questioned by some researchers [28]. The use of an approach closer to that proposed by Inmon has been advocated because of the ease of maintenance of a central and historical database when it is normalized. Data consistency is more easily ensured in this manner. Small data marts, always fed by the NDS, can then be constructed as needed.

In the present work, one of our major concerns was to integrate data coming from several heterogeneous blood center systems, while bearing in mind their quality during the integration process. Therefore, we decided to adopt the approach proposed by Inmon, which facilitates data maintenance and consistency.

Just as important as the DW databases are the components that manipulate the data in order to populate these databases. One very common example of components for this purpose is the ETL (Extract, Transform and Load) tools [22], which have the primary function of facilitating the following processes: extracting data from various systems; transforming, validating and correcting extracted data as necessary; adapting them to the needs of the DW; and loading the data into analytical or normalized data repositories. There are many uses for such tools, even outside of the DW context.

In the DW context, in addition to the components for data manipulation, data visualization tools are also of great importance. Among visualization tools, those for manipulation of OLAP (Online Analytical Processing) cubes stand out because of their versatility and ease of use by the end user. OLAP is a computer-based approach designed for answering multidimensional analytical queries. An OLAP cube is one of these approaches, consisting of a dataset that is stored in a different-from-usual format, in order to facilitate implementation of dynamic queries addressed to summarized information. These cubes are constructed such that each combination of dimensions with measures is precalculated or calculated very quickly. The data that feed the cubes can come either from a dimensional model implemented in a relational database, or from multidimensional databases that already use the cube format to store information. Several graphical tools for visualizing data cubes are available, such as the cube viewer offered by Pentaho Mondrian [4], or even well-known applications such as Microsoft Office Excel [3]. OLAP tools have been used in several domains. Some of them have been adapted to biological environments [16, 19]. “Scientific OLAP” has been proposed [17] for analyzing data in controlled scientific experiments. One important scientific OLAP initiative is called BIOLAP [1]. This approach has sought to solve some of the limitations of classical OLAP through providing a new way to summarize biological data for non-numerical domains. Another approach is called BIOSTAR, and aims to move away from the fact schema to a quadruple schema [29].

From a system development point of view, traditional methodologies that are used to develop software (e.g., waterfall, spiral) are being used to build DWs. More recently, some agile methods have also been introduced in this area. Traditional methodologies have not shown themselves to be capable of efficiently handling the evolution of DW systems, which are very dynamic and require constant addition of requisites and fast error correction. Since 2000, works like the one by Ang & Teo [8] have been showing the inherent difficulty in

constructing DW systems using the available techniques. Such difficulties suggest that an incremental approach and end user participation must form part of the development process, but these works have not shown how this can be done. Given this scenario, DW development has started to make use of agile methodologies. However, in these approaches, initial planning is usually not contemplated. This also contributes towards system evolution problems, because the conceptual essence behind the requirements may not be captured by these methodologies. More recently, Giorgini et al. [13] provided an excellent contribution to requirement analysis, with suggestions for ways to gather requirements until they evolve to the conceptual design. They also compared the existing works using a classification scheme for different development methodologies according to the approach adopted for analyzing the requirements. They divided the existing works into two approaches: *supply-driven* and *demand-driven*. In the supply-driven approach, the requirements come from detailed analysis on the data sources. In the demand-driven approach, the requirements are gathered from the DW end users. However, their work did not consider the need for system evolution using development cycles.

The approach that we have taken has attempted to comprehend both the supply and the demand-driven ways of gathering requirements. Through this approach, our intention was to capture the conceptual essence by using a supply-driven approach, thus developing an initial conceptual model that would guide system development. We then intended to iteratively capture the user's requirements so as to gradually improve the system with features that would really be useful for the end users, with adaptation of the initial conceptual model to reflect those requirements. Hence, we sought to embrace the best of each technique in an effective manner for building up a DW. More concretely, our approach has facilitated resolution of the heterogeneity problem through the integrated application of three good practices: 1) creation and maintenance of a conceptual modeling [11, 14]; 2) application of data analysis [10, 21] to improve and correct the DW; and 3) use of a modular architecture to drive pragmatic DW development [24].

## 3. The REDS-II Blood Donation System

The REDS-II DW system is responsible for performing collection, cleaning, standardization, storage, and report generation relating to Brazilian blood donation data. Each blood center has its own transactional system, which is suitable for local needs and requirements. The first challenge that we faced was how to obtain a global view of the data, since these systems are heterogeneous and do not always represent the data in the same way. As an example, there is the concept of "blood donor deferral". All donor candidates complete a screening questionnaire and, depending on their answers, they are deferred or accepted. At least one reason is associated with each deferral. Because there were no standardized deferral criteria, and there were discrepancies in the scale of observation used to describe these criteria, direct mapping between the different deferral reasons used at the centers became impossible. For example, one deferral reason outlined as "use of testosterone or injected anabolic steroid" by one center could be described more generally as "use of medication" in another.

During the development of the REDS-II system, we tried to identify the best way to apply some important practices in order to achieve successful implementation. In this section, we will present these practices and indicate how they can be applied in a complex domain.

### 3.1. Data Warehouse Modular Architecture for REDS-II

Breaking a system into modules not only improves the system organization, but also facilitates its maintenance, since this practice draws the boundaries between each of its parts, thereby separating their roles and functional features and establishing well-defined

interfaces for interactions with them. This practice also facilitates reuse of system components and enables system growth. Therefore, we used a modular architecture based on one of the architectures proposed by Rainardi [24], divided into four distinct software modules (and three database levels) that are illustrated in Figure 1. These four modules are as follows:

1. Module 1 is responsible for receiving and temporarily storing transactional data coming from each blood center donation system in a stage area[2] (the first database level). However, although blood centers deal with basically the same concepts, each center has its own particular formats for recording them. Therefore, this stage must have separate databases to accommodate different formats. We had one database per blood center. At this level, no data corrections and adjustments were made.
2. Module 2 detects and corrects possible mistakes and homogenizes the data. As expected, input data from heterogeneous sources often use different values or encodings for the same attributes, which sometimes require a certain amount of uniformization. For example, one center can represent a positive test result as 'P' and the other center can represent it as 'Y'. Furthermore, several types of errors can be found in these data. Such errors must be corrected when possible or properly marked when there is no possibility of correction. These adjustments are also made in the stage area.
3. Module 3 is responsible for getting data from the stage and inserting it (or updating it) into a normalized historical database (the second database level). Once corrected and standardized, the data need to be definitely stored and integrated with the data already existing in the system. For this purpose, we created a normalized historical database (Normalized Data Store, NDS). This NDS was based on a conceptual model (see Section 3.2) and was created for storing data from all the centers. In this module, some integrity verifications between new data and the data that already exist in the NDS need to be run in order to ensure NDS consistency.
4. Module 4 focuses on data visualization, analysis and export. Here, we built analytical databases (the third database level) to facilitate generation of OLAP cubes and processing of more elaborate queries. Our analytical databases were composed of fact tables and dimension tables. The fact tables contain numerical measures used for donation analysis. The dimension tables represented a set of perspectives relating to the system that we want to study, such as test results, gender, date of birth, date of donation, and other factors. From these databases, some cubes can be extracted. By using cubes, non-expert users can discover relationships between different attributes using software tools that are familiar to them (like Excel). This module also contains components for constructing relational views of the NDS, spreadsheet generators, and processes that select data for mining programs.

### 3.2. Normalized Data Store (REDS-II NDS): Conceptual Modeling

Although the blood centers participating in this study deal with the same blood donation process, each one of them designs and operates the process in its own way. This circumstance leads to several different models. However, by analyzing the data from the centers, we concluded that, despite these differences, the main concepts were always the same. The conceptual model [11, 14] helped us to extract these key concepts, through

[2]According to Rainardi [24], a stage is "an internal data store used for transforming and preparing the data obtained from the source systems, before the data is loaded to other data stores in a data warehouse".

allowing us to focus on what really mattered within a process. In addition to identifying essential entities involved in the donation process, it also helped in identifying the relationships between those entities. The model, once developed, should be used to build the data warehouse structure. Furthermore, it is essential to keep the model updated, so that structural changes in the DW are a reflection of changes made primarily to the conceptual model. Thus, our normalized database, which stores historical data from all the blood centers, was developed based on our blood donation conceptual model (Figure 2). It is important to notice that the model described here was the result of several iterations that involve data analyses and, based on these, subsequent improvements to the model initially developed.

*Screening* is one of the identified entities. It is the first contact between the *donor candidate* and the blood center. Basically, it consists of data from interviews, questionnaires, and clinical examinations (like blood pressure, height and pulse) obtained at the beginning of each visit. Depending on the blood center rules, each screening can evolve to blood *collection* (if the candidate is approved) or to *deferral* (if the candidate is rejected). It is important to notice that screening will never produce both situations, since they are mutually exclusive. If a deferral is produced, it will be linked to the reason(s) that led to the rejection (*deferral reason*). On the other hand, if a collection is produced, two products should be obtained: one or more *blood unit*s and one *sample* for blood testing. To the aggregation of relationships between a collection and its related blood units, the name *donation* is assigned. The sample is the basis for several investigative *searches*, whose results will determine the destination given to blood units relating to the same collection. A search, in turn, is also composed of *tests*, which will be evaluated individually. The sample result will be determined based on these test results. One example that can be envisaged is HIV search, which is composed of two different tests. This search will only be considered positive if both tests are reactive.

The REDS-II NDS is our storage element for data coming from several years of donations at all the participating blood centers. It is our organized data source for all analytical questions and research, and so it is important to note that data insertions in this database must be carefully made. *Ad hoc* inclusion of new entries in the NDS can lead it to an inconsistent or even unusable state. To avoid such situations, before allowing a definitive change in our database, we ensure that hundreds of integrity constraints are applied, thereby maintaining the consistency and validity of our database. To illustrate the types of constraints that are used in our validation scripts, we have chosen some examples depicted in Figure 3.

### 3.3. Data Analyses

After finishing DW construction, we had a representative tool for evaluating managerial information that enabled us to perform data analyses [10, 21]. The DW provides a global view of the data transformation process, from the transactional origin to the analytical final result. In particular, the DW provides important features in two contexts: consistency verification and decision support system. Before exploring these, we present a possible visualization structure for the blood center DW.

Figure 4 illustrates a query extracted from a cube generated from one of our analytical databases. This figure is divided into two parts: a list of variables and a data manipulation area. The list of variables indicates different views that blood center information can assume. The data manipulation area indicates a region where the user can freely manipulate the variables and create many classifications from the original data. Variables can be combined in order to generate views of greater complexity. The current version of this tool provides the following variables: *ABO group*, *age*, *birth date*, *blood center*, *country*, *declared race*, *donation procedure*, *donation date*, *donor type*, *education*, *previous screening*

*history*, *fixed or mobile donation indication*, *gender* and *pregnancy*. In addition, the DW is able to manage tests results, like *Chagas disease*, *hepatitis B* (*HBSAG* and *HBCAB*), *hepatitis C* (*HCV*), *HIV*, *syphilis*, *human T-lymphotropic virus* (*HTLV*) and their respective supplemental results.

Our analysis strategy includes offering cubes as described above to the end users. Using a cube (Section 2), the user can quickly combine these variables, because in this kind of structure, the values for every possible combination of dimensions were precalculated during cube construction. This characteristic provides agility for end users, who may try many query options to determine the most appropriate one for their analyses.

Figure 4 and Figure 5 are examples of queries considering the variables *blood center* and *HCV test*. The DW builds an automatic table in which all data are classified considering the blood center and, internally, the HCV test. The values for blood centers are Pró-Sangue Foundation, Hemominas and Hemope. The values for the HCV test are *negative*, *positive*, *not tested*, *undefined* and *unit not collected*.

Figure 6 shows another investigative option that illustrates the power of flexibility offered by this kind of tool. The number of donations at the Hemope blood center over a one-year period is plotted on a graph according to blood type. This graph can easily be adapted to see the details of daily collection for a specific month (Figure 7). For example, by using this graph, it is easier to visualize the impact of Carnival (February, 13–16 in the year illustrated) on blood collections in Brazil.

A good way to detect the presence of both conceptual and implementation errors is by analyzing the available data from a global point of view. Several verifications can be conducted in this manner, such as consistency verifi-cation. The contribution of data analysis towards consistency verification appears during some linked analyses. One simple example of verification is identification of errors in the *Numpreg* (number of pregnancies) field, by associating it with the *Pregnant* field, which indicates whether the donor has been pregnant. The presence of a ‘No’ value in the latter field must be followed by a ‘Not Applicable’ value in the former field. Like in the example above, experts may identify many other complex associations that probably indicate transactional or transformation mistakes, like typing errors, updating difficulties, system problems and so on.

Data analyses can also facilitate informed decision-making. In many cases, analyses using the DW may point towards unexpected, but valid observations such as greater HIV prevalence among family replacement donors than among blood donors from the community. In this situation, analyses of greater precision may indicate some special behavior of a particular population, or a possible data problem. In both cases, the DW helps experts to make more rigorous investigations using information from their own databases.

### 3.4. Methodological Guidelines

In the previous sections, we presented the following practices: use of a modular architecture, development of a conceptual model and generation of data analyses. It is important to note that these topics, when considered separately, are well known within the field of database systems and have already been widely used in several projects. However, the methodological guidelines that form the main contribution of the present paper are based on their combined use. The use of these practices from a cyclic and interactive perspective is innovative, thus enabling reduction of the complexity of DW development.

When working in a complex domain that includes many variables (such as the blood donation domain), it is important to firstly concentrate on the essential parts of the process,

instead of trying to build a final and complete system at this stage. Only after obtaining a stable version, the system should be slowly and progressively expanded to include the remaining variables and the concepts of minor importance. Otherwise, we would be prone to mistakes. This strategy is detailed through the guidelines presented below, and constitutes a feasible way to effectively develop a DW. These guidelines are presented in an algorithmic manner (Algorithm 1), in order to highlight the order of the steps to be followed. The ideas presented in this algorithm are explained in text format to complement and facilitate their understanding.

There are typically two scenarios within which there is a need to build DWs that compile data coming from heterogeneous sources. The first of these occurs when there are several distinct systems that implement the same process. The blood center transactional systems are an example of this situation, in which the process under implementation is blood donation. The second scenario is constituted by heterogeneous systems that implement different but related processes. In any of the scenarios considered, it is important to focus on the processes implemented by the systems. Through looking at the processes, it can clearly be seen what the systems have in common and how they are connected to each other. Likewise, distraction factors that have been introduced during the implementation can be neutralized: for example, different interfaces for implementing the same concepts. The purpose in concentrating on processes that are hidden behind system implementations is to achieve a feasible way of identifying the main entities that need to be modeled in the DW, how they are related and their attributes. This identification constitutes line 1 of the guidelines. Although some approaches towards identifying these entities have been devised, it is not an easy task to conceive all the details of the processes involved at the initial phase of development. It is therefore important that the system evolution should be gradual.

After the main entities and relationships have been established, it is possible to construct a conceptual model that will be used to develop the normalized database, thus giving rise to the first version of the DW (line 2). The conceptual model will be essential for representing the data (entities) that will be integrated and the relationships between them. In addition, this helps with regard to another important point in building a DW system: identification of constraints to be placed on the data (line 3). The constraints give information regarding the validations that need to be implemented. In turn, the validations are responsible for ensuring the quality of the data in the DW. However, because of semantic limitations, not all constraints can be inferred by conceptual model. These limitations are the reason why data analysis (line 21) also has an essential role for increasing the data quality. At the first time, lines 5 to 13 will not be executed, because their conditions are not initially satisfied, so they will be explained further ahead. The existence of the repeat/until loop (lines 4 and 23) will also be explained later on.

Once a conceptual model has been produced, the DW system can be implemented. At this point, the first thing to do is to build the NDS following the schema that has been developed (line 14). There are some widely used tools [5, 2] that automate this job, to generate a database construction script from a given model. Now that the NDS has been constructed, it is known which data will have to be extracted from the source systems. Therefore, it is time to build the stage databases to receive these data (line 15). Based on the identified restrictions, the validation (and correction) routines for improving the quality of the data that will be stored can be implemented (line 16). Based on the stage databases and the NDS, routines for populating the NDS can be developed (line 17). If new data are available for populating the NDS, this can be done using the routines that have been developed (line 18). After a working NDS has been achieved, the analytical databases can be constructed (line 19) and the routines for populating them can be implemented (line 20). The populated analytical databases can be used to generate views and reports (line 21). The analyses on

these views will be crucial for checking for errors in the system and for expanding it. In integration systems, data quality and system development are closely related concepts. The more efficient the generation of partial system deliveries is, the faster the end user can perform data analyses. From these analyses, users are able to find inconsistencies that indicate the existence of possible errors (line 22). If there is an error in the implementation of the system or in the conceptual model, it can be corrected by the developers prior to the next delivery. However, the error may be in the data. This means that such information has mistakenly been entered in the system, thus signaling a failure in the validation process. Such failures reveal either that there are errors in the existing validation tasks or that new validations must be implemented. Thus, data analysis contributes towards increasing the data quality. Looking at this from another perspective, it is also possible to say that the more reliable the data in the system is, the greater the confidence that can be placed in the analyses. In other words, data quality improves the analyses which, in turn, improve data quality.

In addition to detecting inconsistencies in the system, analyses are important for finding what entities and attributes should be included in order to expand the system (line 22). From end users' real needs, they are able to say what the next steps in developing the DW system should be, which ensures that hours will not be spent in developing parts of the system that will not be used in practice.

From the analyses, it is possible to build a new version of the conceptual model, containing corrections (lines 5 – 7) of the errors found. After several cycles of analysis and development, a reasonably stable version of the DW will eventually be achieved (line 8). At this stage, expansion of the conceptual model can be envisaged, in order to cover more entities and attributes (lines 9 – 12), thus starting the cycle of analyses and corrections again. In addition to system evolution, the data already stored in the NDS needs to be considered, since these data have to evolve in the same way. However, this task has not been difficult with regard to our data, since all the components that access our central data repository are under our control, which has allowed us to make corrective scripts to be used in the NDS without significant consequences. Several works [7, 27] have previously reported using data evolution techniques when changes to the schemas occurred. Such studies have proven to be functional and therefore this subject will not be dealt with in the present work.

The cyclic characteristic of the guidelines is represented through the "repeat/until" loop from line 4 to line 23. It is important to notice that, in this loop, correction of errors takes precedence over the system growth, and the system will not be expanded until a stable version has been achieved. Another important point is that after the first cycle, all the routines and databases will need just a few adaptations for them to work in the new scenarios, and implementation of these adaptations will be facilitated when a modular architecture is used.

In fact, considering a context in which development takes place in several cycles, the various stages of refactoring, correction and expansion to which the system may be subjected require both high flexibility and loose coupling. With regard to these characteristics, it is important to adopt well-defined modular architecture that helps in system organization. In a large-scale system, it would be easy to get lost in a great number of code lines with hundreds of validations. Modularizing the system in such a way that its parts are grouped according to their functional features was a way that we found for efficiently accelerating system development and facilitating its maintenance.

## 4. Discussion

In dealing with DW systems in which the data come from heterogeneous systems, “we cannot boil the ocean”. It is a hard task to understand all the details of the available data that come from multiple sources at once. To facilitate development of such a system, it is important to focus on entities that are common among all the sources. Through these entities, a starting point for building a conceptual model is reached and, from this, for building the DW. After the first version has been developed, it can be used to populate the databases and generate some views and reports. The analyses on these views will then be very useful in verifying whether any errors exist in the concepts that have been modeled so far. It will then be possible to develop a new version of the conceptual model, containing corrections for detected errors. After completing enough cycles of analyses and development for a reasonably stable version of the DW to be achieved, expansion of the conceptual model can be envisaged, such that it will cover more entities or attributes, thus restarting the cycle of analyses and corrections. All these changes will be easily accomplished if modular architecture with high flexibility and loose coupling is used.

Combined use of the practices that we have discussed here (modular architecture, conceptual modeling and data analyses) from a cyclic and interactive perspective allowed us to reduce the complexity of DW development. In other words, by following these good practices, we quickly and efficiently achieved a DW that contained the most interesting entities stored and with the possibility of generating managerial views and reports from these data. We were therefore able to solve the main cause of DW development failure: the delays in delivering a functional system. However, the following question can be raised: what would happen if only a subset of these practices were to be used in a project? To attempt to answer this question, let us analyze what would happen if each of the following practices were to be separately excluded from the whole scenario:

- Without modular architecture: the evolution of the conceptual model through evaluation of the analyses generated by the end users would lead us to constant changes in system implementation. These changes would be necessary in order to correct errors found or to expand the system. Lack of system modularization would make these changes hard to implement, and the development iterations would take more time to accomplish. This would lead to loss of the agility of our method, with great amounts of time spent on deliveries.
- Without a conceptual model: it is well known that absence of a conceptual model in a database project is as catastrophic as the lack of a floor plan in constructing a building. Complex databases need to be carefully thought through before being implemented. Conceptual models also provide a way of thinking about and documenting what is going to be built. Without them, it would be extremely difficult to implement the system or to know what changes would have to be made to the system in order to correct it or to expand it.
- Without the analyses: the analyses are the way that end users have for giving the developers feedback about the system quality. Today, the importance of end user participation in the development process is well known. If this feedback is unavailable, the system status will remain unknown. It will only be possible to answer the question of “What must be corrected?” and “Which aspects must be implemented as the next step?” through generation of analytical structures, with end users’ analyses.

In the REDS-II project, these guidelines have proved to be highly functional, with excellent results. Currently, our REDS-II DW stores over 1.7 million screenings from about 950,000 candidates, with more than 1.3 million donations and 380,000 deferrals. From the blood

samples, more than 10.5 million test results have been recorded so far. All these data came from heterogeneous transactional blood donation systems and are now part of our DW, which was built using the practices discussed in this work. Several analytical results were obtained, and can be found in the related references [9, 6, 26, 23, 25, 15].

## 5. Conclusions

Working with a decision support system like a DW whose data come from heterogeneous sources brings several problems for developers. One of them is to deliver a useful version of the system on time.

In this article, we have shown that the combined use of three good practices was very convenient for developing a complex decision support system such as a DW. These practices were the following: structuring the code according to modular architecture; creation and maintenance of a conceptual model that guided DW construction; and data analysis as a way of finding possible errors in the system. These practices should be applied cyclically with corrections focusing initially on the main entities of the system, leaving the less important entities to be incorporated when a working version of the system already exists.

These three good practices have been successfully applied in constructing the REDS-II DW project and our plan is to expand it to integrate four additional centers, in order to gradually cover all blood collections in Brazil.

With regard to future directions, we intend to work on two fronts: a framework for automated adaptive statistical data validation in DW systems and advanced statistical analyses. In today's DW systems, there is a gap in validation automation. It would be important to have a framework that could offer validations that, based on statistical measures of input data batches and the data already validated and in the DW repository, could make decisions about the validity of new batches. Our intended contribution is to provide a framework that facilitates the use of statistical validations in integration tools. Implemented validators need to be flexible enough to adapt to the data that they process. This adaptive mechanism will be developed using machine learning techniques and, more specifically, classifiers. Through improving the validations, we intend to improve user analyses and, in this manner, the development process as a whole. Furthermore, we intend to expand the statistical analyses that we do today, and increase their complexity, so that end users will be able to see the data from other perspectives, thus facilitating the search for new requirements and the correction of errors.

## Acknowledgments

This work was supported by the National Institutes of Health (NIH-USA), project entitled "Recipient Epidemiology and Donor Evaluation Study II", grant number nHHSN268200417175C. The authors also thank Prof. Julio M. Singer for helping in the review for final version.

## References

1. [Accessed on 14-Feb-2013.] BIOLAP. 2012. http://biolap.sourceforge.net/)
2. [Accessed on 14-Feb-2013.] CA ERwin Data Modeling. 2012. http://erwin.com/
3. [Accessed on 14-Feb-2013.] Microsoft Excel. 2012. http://office.microsoft.com/en-us/excel/
4. [Accessed on 14-Feb-2013.] Pentaho Mondrian Project. 2012. http://mondrian.pentaho.com/
5. [Accessed on 14-Feb-2013.] SAP Sybase PowerDesigner. 2012. http://www.sybase.com/products/modelingdevelopment/powerdesigner/

6. de Almeida-Neto C, Liu J, Wright DJ, Mendrone A Junior, Takecian PL, Sun Y, Ferreira JE, de Alencar Fischer Chamone D, Busch MP, Sabino EC. Demographic characteristics and prevalence of serologic markers among blood donors who use confidential unit exclusion (CUE) in São Paulo, Brazil: implications for modification of cue polices in Brazil. Transfusion. 2011; 51:191–197. [PubMed: 20663108]
7. Ambler, SW.; Sadalage, PJ. Refactoring Databases: Evolutionary Database Design. Addison-Wesley Professional; 2006.
8. Ang J, Teo TSH. Management issues in data warehousing: insights from the Housing and Development Board. Decision Support Systems. 2000; 29:11–20.
9. Carneiro-Proietti AB, Sabino EC, Sampaio D, Proietti FA, Gonçalez TT, Oliveira CD, Ferreira JE, Liu J, Custer B, Schreiber GB, Murphy EL, Busch MP. Demographic profile of blood donors at three major Brazilian blood centers: results from the International REDS-II study, 2007 to 2008. Transfusion. 2010; 50:918–925. [PubMed: 20003051]
10. Chaudhuri S, Dayal U. An overview of data warehousing and OLAP technology. SIGMOD Rec. 1997; 26:65–74.
11. Elmasri, R.; Navathe, SB. Fundamentals of Database Systems. 4. Addison-Wesley; Boston, MA, USA: 2003.
12. Gartner. [Accessed on 14-Feb-2013.] Gartner says more than 50 percent of data warehouse projects will have limited acceptance or will be failures through 2007. 2005. http://www.gartner.com/it/page.jsp?id=492112
13. Giorgini P, Rizzi S, Garzetti M. GRAnD: A goal-oriented approach to requirement analysis in data warehouses. Decision Support Systems. 2008; 45:4–21.
14. Golfarelli, M.; Maio, D.; Rizzi, S. Conceptual design of data warehouses from E/R schema. Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences, HICSS'98; Kona, HI, USA. p. 334
15. Gonçalez TT, Sabino EC, Schlumpf KS, Wright DJ, Silvana DS, Leao C, Takecian PL, Carneiro-Proietti AB, Murphy EL, Busch MP, Custer B. Vasovagal reactions in whole blood donors at 3 REDS-II blood centers in Brazil. Transfusion. 2012; 52:1070–1078. [PubMed: 22073941]
16. Han, J.; Pei, J.; Dong, G.; Wang, K. Efficient computation of iceberg cubes with complex measures. Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, SIGMOD'01; Santa Barbara, CA, USA. p. 1-12.
17. Huyn, N. Scientific OLAP for the biotech domain. Proceedings of the 27th International Conference on Very Large Data Bases, VLDB'01; Rome, Italy. p. 645-648.
18. Inmon WH. What is a data warehouse? Prism Tech Topic. 1992; 1
19. Jensen CS, Kligys A, Pedersen TB, Timko I. Multidimensional data modeling for location-based services. VLDB J. 2004; 13:1–21.
20. Johnson, JH. Keynote speech, 2002. Presented at The XP 2002 Conference; Sardinia, Italy.
21. Kimball, R. The data warehouse toolkit: practical techniques for building dimensional data warehouses. John Wiley & Sons; New York, USA: 1996.
22. Kimball, R.; Caserta, J. The data warehouse ETL toolkit: Practical techniques for extracting, cleaning, conforming, and delivering data. Wiley; 2004.
23. Patavino GM, de Almeida-Neto C, Liu J, Wright DJ, Mendrone-Junior A, Ferreira MIL, de Freitas Carneiro AB, Custer B, Ferreira JE, Busch MP, Sabino EC. NHLBI Retrovirus Epidemiology Study-II (REDS-II) International Component, Number of recent sexual partners among blood donors in Brazil: associations with donor demographics, donation characteristics, and infectious disease markers. Transfusion. 2012; 52:151–159. [PubMed: 21756264]
24. Rainardi, V. Building a Data Warehouse: With Examples in SQL Server. Apress; 2008.
25. Sabino EC, Gonçalez TT, Carneiro-Proietti AB, Sarr M, Ferreira JE, Sampaio D, Salles N, Wright DJ, Custer B, Busch MP. Human immunodeficiency virus prevalence, incidence, and residual risk of transmission by transfusions at Retrovirus Epidemiology Donor Study-II blood centers in Brazil. Transfusion. 2012; 52:870–879. [PubMed: 21981109]
26. Sabino EC, Salles NA, Sarr M, Barreto AM, Oikawa M, Oliveira CD, Leao SC, Carneiro-Proietti AB, Custer B, Busch MP. NHLBI Retrovirus Epidemiology Donor Study-II (REDS-II) International Component, Enhanced classification of Chagas serologic results and epidemiologic

characteristics of seropositive donors at three large blood centers in Brazil. Transfusion. 2010; 50:2628–2637. [PubMed: 20576017]

27. Sadalage, PJ. Recipes for Continuous Database Integration. Addison-Wesley; 2007.
28. Schuff D, Corral K, Turetken O. Comparing the understandability of alternative data warehouse schemas: an empirical study. Decision Support Systems. 2011; 52:9–20.
29. Wang L, Zhang A, Ramanathan M. BioStar models of clinical and genomic data for biomedical data warehouse design. Int J Bioinformatics Res Appl. 2005; 1:63–80.

## Biography

Pedro Losco Takecian is a PhD Candidate in the Department of Computer Science at the University of São Paulo, Brazil. He received his Computer Science MSc and BSc degrees also from the University of São Paulo. His previous work experiences include business process formal modeling, database modeling and finance industry software development. His current research interests include business intelligence, integration of heterogeneous databases and their applications in health domain.

## Research Highlights

- Appliance of some good practices to facilitate data warehouse system development.
- Modularizing a data warehouse system makes its maintenance easier.
- A conceptual model guides data warehouse system construction.
- Data analyses are important for data warehouse system debugging.
- Combined use of specific practices promotes agile data warehouse development.

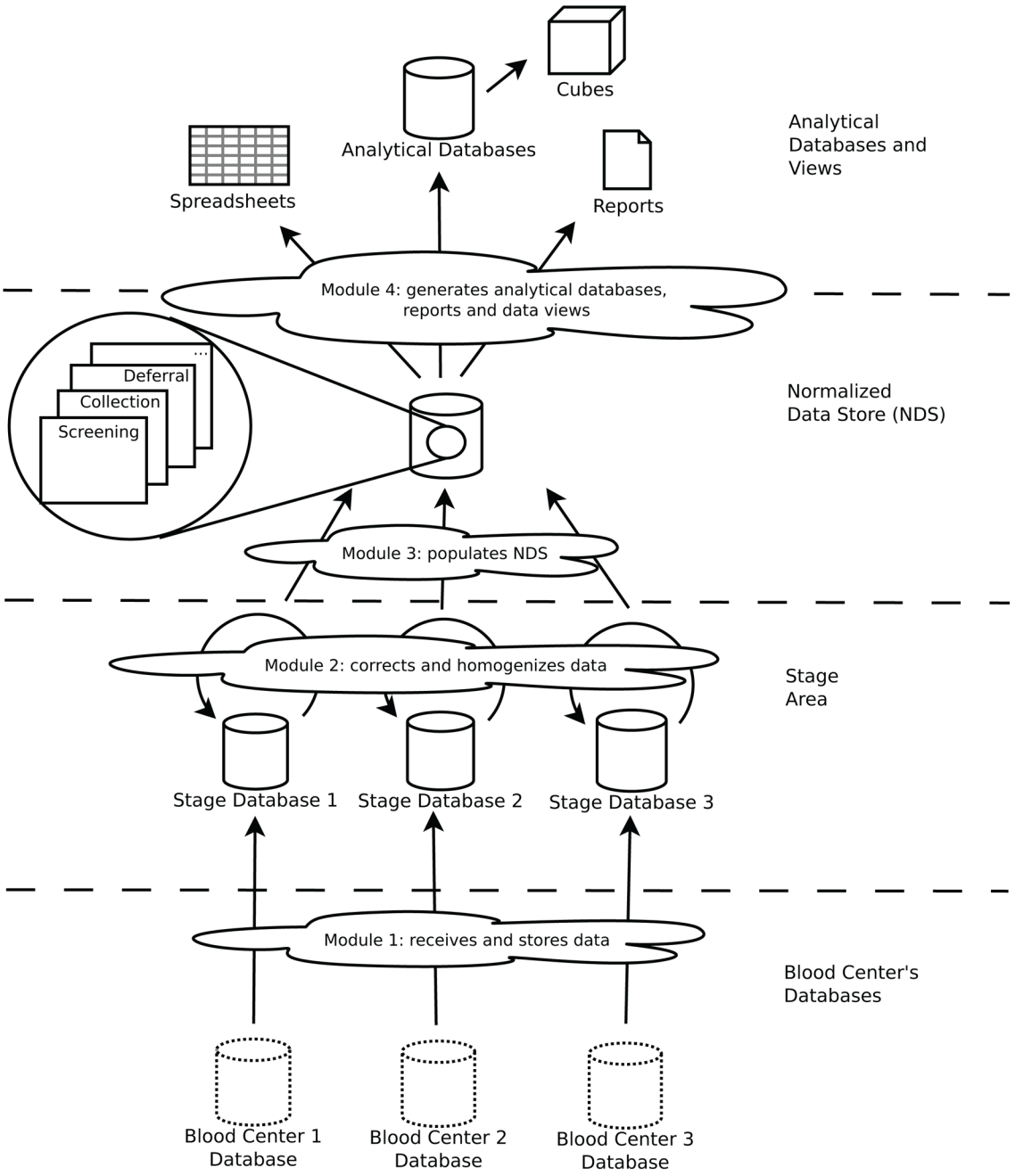



**Figure 1.**
Data Warehouse Architecture for REDS-II.

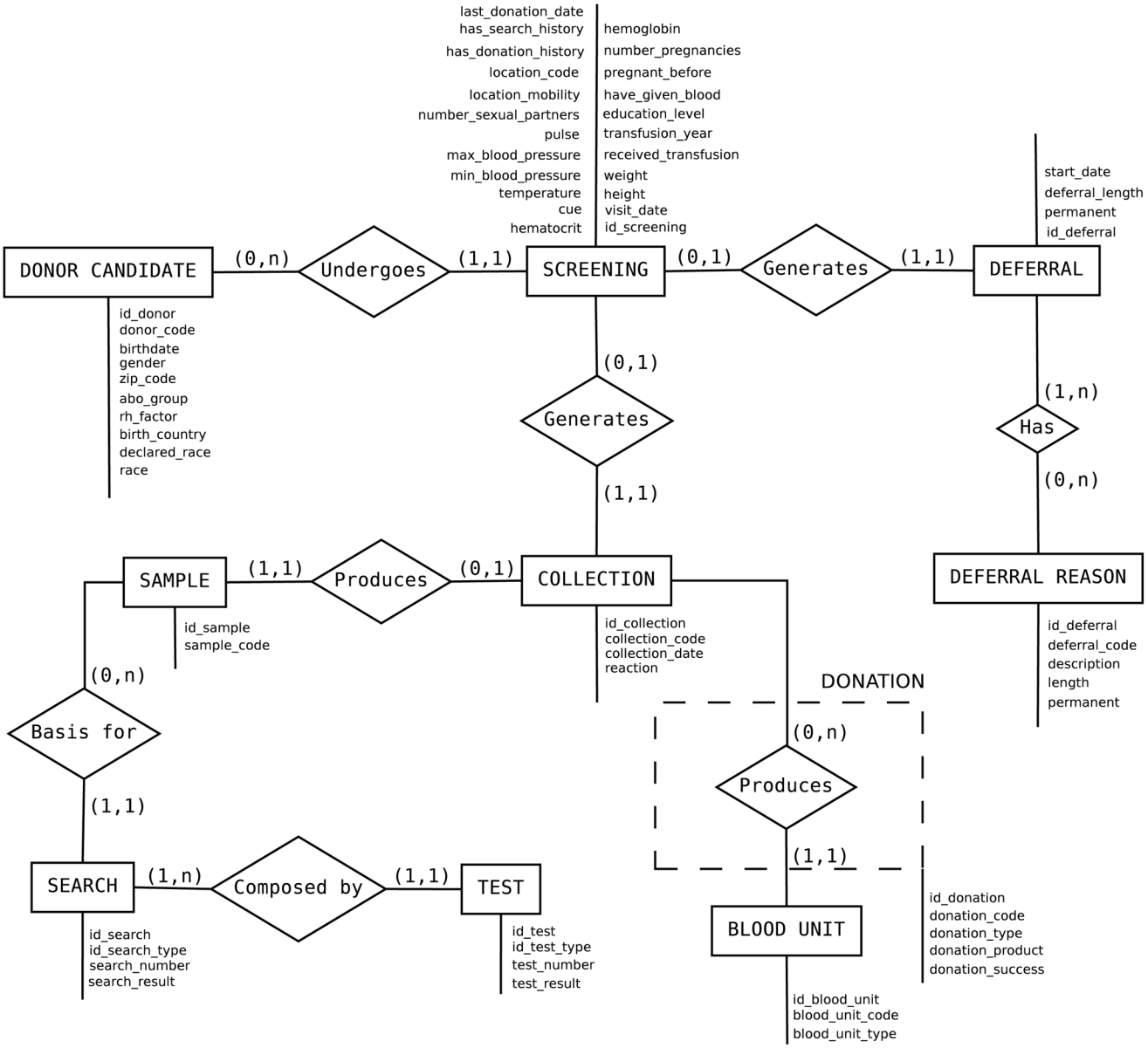
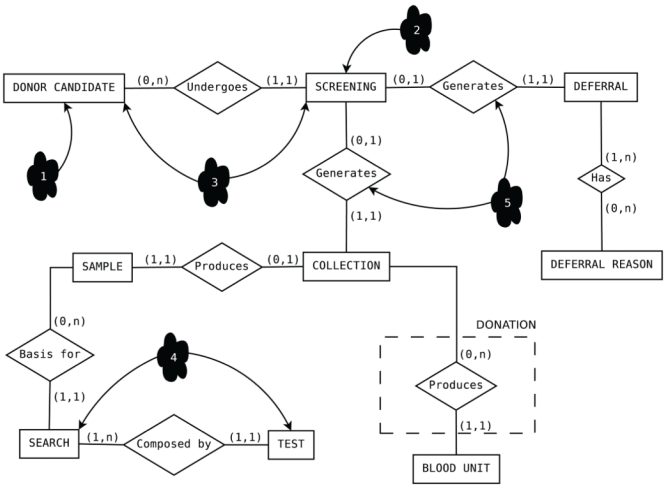



**Figure 2.**
Conceptual model of normalized database (Extended Entity-Relationship diagram).




| # | Integrity Type | Example |
|---|---|---|
| ❶ | Field integrity | 'Gender' field only allows values 'M' (male) and 'F' (female) |
| ❷ | Integrity between fields of the same entity | If 'Pregnant' has value 'N' (donor candidate is not pregnant), we check the presence of value 'Not applicable' at 'Numpreg' (number of pregnancies) field. |
| ❸ | Integrity between fields of different entities | If 'Gender' has value 'M', we check if 'Pregnant' field has value 'Not applicable'. |
| ❹ | Referential integrity | Every test must be linked to one and only one search. |
| ❺ | Integrity of disjunction | A screening can generate a deferral or a collection or even none of them, but not both. |

**Figure 3.**
Examples of integrity constraints for the donation process.

| Row Labels | Count |
|---|---|
| **Fundacao Pro-Sangue - SP** | **404,901** |
| Negative | 400,713 |
| Not tested | 2,390 |
| Positive | 966 |
| Undefined | 404 |
| Unit not collected | 428 |
| **Hemominas - MG** | **211,769** |
| Negative | 208,151 |
| Not tested | 3,295 |
| Positive | 211 |
| Undefined | 112 |
| **Hemope - PE** | **313,040** |
| Negative | 307,625 |
| Not tested | 3,285 |
| Positive | 571 |
| Undefined | 713 |
| Unit not collected | 846 |
| **Grand Total** | **929,710** |

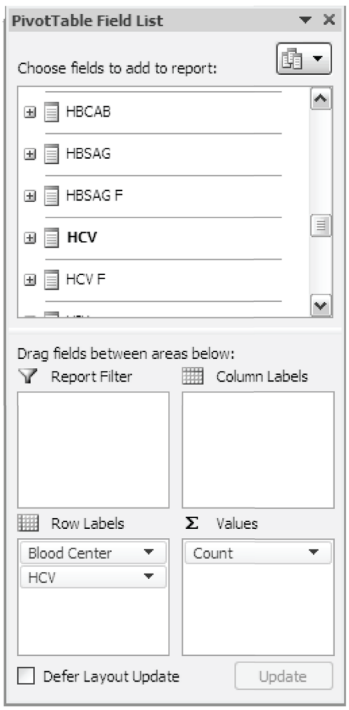
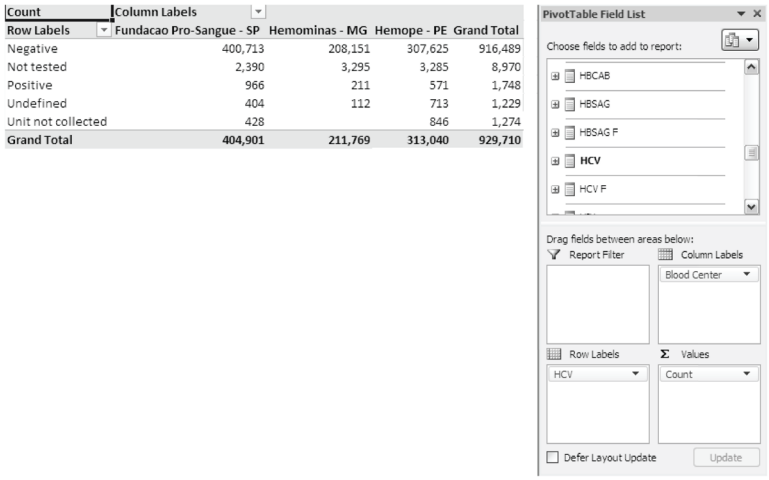



**Figure 4.**
Data warehouse visualization of blood center versus HCV analysis



| Count | Column Labels | | | |
|---|---|---|---|---|
| Row Labels | Fundacao Pro-Sangue - SP | Hemominas - MG | Hemope - PE | Grand Total |
| Negative | 400,713 | 208,151 | 307,625 | 916,489 |
| Not tested | 2,390 | 3,295 | 3,285 | 8,970 |
| Positive | 966 | 211 | 571 | 1,748 |
| Undefined | 404 | 112 | 713 | 1,229 |
| Unit not collected | 428 | | 846 | 1,274 |
| **Grand Total** | **404,901** | **211,769** | **313,040** | **929,710** |



**Figure 5.**
Another visualization for blood center values HCV analysis

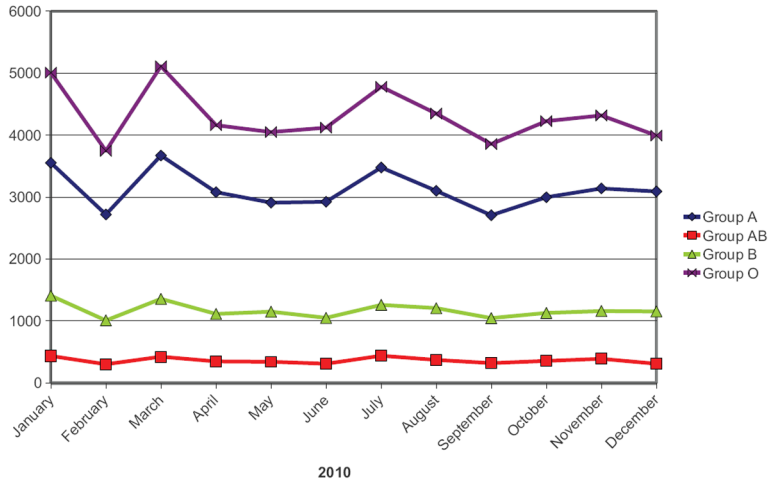



**Figure 6.**
Number of donations at Hemope blood center per month according to blood type (2010)

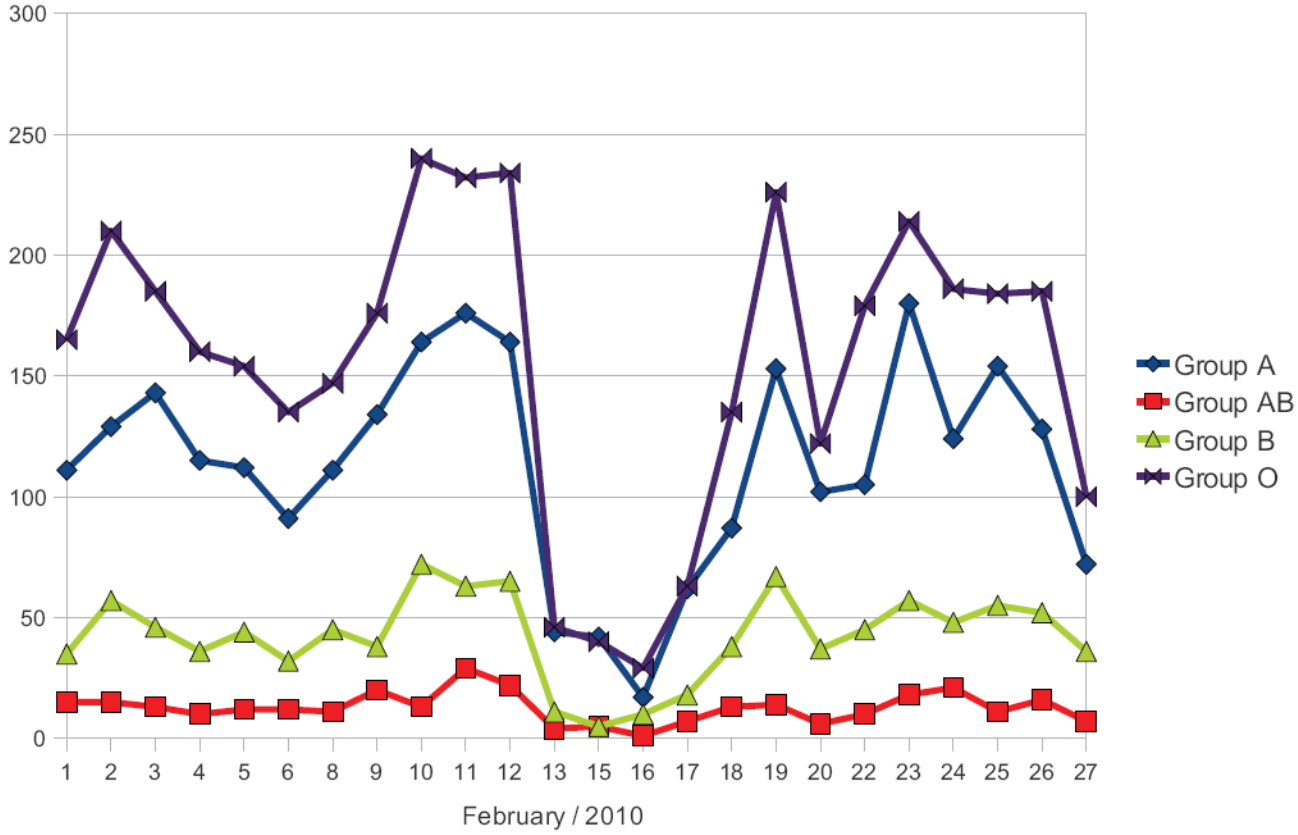



**Figure 7.**
Number of donations at Hemope blood center per day according to blood type (Feb/2010)

**Algorithm 1**

Guidelines for building an effective DW.

```
Identify the main entities, relationships and attributes involved
Build an initial conceptual model
Identify constraints
// Hypothesis: at this point, there are no identified errors
// and no new requirements
repeat
    if there are errors then
        Correct the conceptual model
        Correct constraints or identify new ones
    else // here, we have a DW stable version
        if there are new requirements then
            Change the conceptual model to embrace new requirements
            Identify new constraints
        end if
    end if
    (Re)construct the NDS based on the conceptual model, migrating existing NDS data to new scenario
    (Re)build stage databases to store the necessary data
    (Re)implement validation routines based on identified constraints
    (Re)implement routines to populate NDS
    Populate the NDS
    (Re)build analytical databases
    (Re)implement routines to populate the analytical databases
    Generate analyses and reports to end users
    Identify possible errors and new requirements
until there are no identified errors or no new requirements
```