PolyEHR: A Framework for Polyglot Persistence of the Electronic Health Record

André Magno Costa de Araújo[1], Valéria Cesário Times[1], Marcus Urbano da Silva[1]
[1]Center for Informatics, Federal University of Pernambuco, Recife, Brazil

Abstract - *Building data schemas for storage in the Electronic Health Record (EHR) has traditionally been done using a single data model. This practice increases the complexity in application development due to the heterogeneity of data in the health care sector, which consequently makes the data schema rigid. In addition, an approach that stores EHR from applications built from heterogeneous database archetypes is lacking or unknown. This article presents a framework that builds application templates for the health case sector using archetypes and storing EHR data in heterogeneous databases through polyglot persistence. To generate templates and data schemas, we extract three elements from archetypes: data attributes that define the EHR, terminologies and vocabulary which give the clinical data a semantic meaning as well as constraints specified on data attributes. Finally, we demonstrate the creation of health applications templates using archetypes and EHR storage in heterogeneous databases*.

Keywords: Applications in healthcare, frameworks for Big Data, Archetypes, Health Information Systems.

1 Introduction

The creation of data schemas for storage in the Electronic Health Record (EHR) is a relevant theme when considering the life cycle of a Health Information System (HIS) and is the object of study in several research papers [1,2]. According to ISO/TS 18308 [3], an EHR data schema must store all relevant clinical events that shall be used in patient's care, including textual descriptions, numeric values, logical values, date and time expressions and hierarchical data structures. Furthermore, it must store data in tables in a way that the relationships between columns and rows are preserved and allow data storage by pairs of attributes (e.g. key-value).

As indicated in [4], the concept of archetypes and templates proposed in openEHR represent an important standard in health care. It minimizes the problems of modeling heterogeneity of EHR data and facilitates the standardization of terminologies and constraints for a given health care sector. An archetype may be defined as a computer expression represented by domain-specific constraints that model and add semantic meaning to the EHR, while templates are graphical user interfaces (GUI) automatically generated and based on archetype specifications [5].

Traditionally, a single data model has been used to represent the different types of EHR data. This practice is present in data schemas designed with or without the use of archetypes [6, 7]. Due to the variety of data types (i.e. structured and non-structured) found in health care sectors, the use of a single storage model might increase the complexity in HIS development, resulting in rigidity in the data schema (i.e. an alteration in the data schema depends on a software development team) and compromises the application performance in data processing.

In the health sector, storage in heterogeneous databases is one alternative to represent the data diversity of several HIS applications (e.g. diagnosis and therapy, clinical and assistance, supplies and revenue management). In that sense, the concept of polyglot persistence proposes a mixed approach for combined storage – the consistent characteristic of relational data models, alongside the flexibility of NoSQL data models (i.e. key-value, document and graph). The core idea is to store structured data using a relational approach, while semi-structured or non-structured data are stored in NoSQL data models.

Polyglot persistence has been applied in a variety of applications, such as IBM's auto scaling PaaS architecture [8], source-code-based data schemas identifications tools [9] and the re-engineering of legacy systems for heterogeneous databases [10]. As pointed out in [11], polyglot persistence is new, promising and evolving. Thus, one may notice that some research projects are dedicated to the creation of new storage architectures for HIS [11, 12], adapting to a new reality in heterogeneous database storage.

Although polyglot persistence has been debated and applied in the health care sector, an approach that stores EHR from applications built from heterogeneous database archetypes is lacking or unknown. There are three main advantages in the use of polyglot persistence for EHR storage created from archetypes. First of all, the data schema is created from a health standard that makes data attributes, terminologies and constraints uniform. Secondly, heterogeneous data can be stored in different data models, for example, the hierarchical data of a lab exam result can be organized as a document set in a NoSQL database, while the patient's demographic data can be stored in a relational data schema. Finally, it minimizes problems caused by the constant changes in data schema. Here, the idea is to store the clinical EHR data that undergo the most changes in flexible data models (e.g. key-value, document and graph), while other data that suffer less alterations are stored in relational data schemas.

This paper proposes a framework named PolyEHR, which builds health application templates using archetypes and store EHR data in heterogeneous databases by means of polyglot persistence. To generate templates and data schema, PolyEHR extracts the following elements from archetypes; i) EHR data attributes; ii) terminologies and vocabulary that give semantic meaning to clinical data, and iii) constraints specified over data attributes. As the main contributions of the present work, we highlight; i) the specification of an architecture that demonstrates how PolyEHR creates templates for health applications and persists data in heterogeneous databases; ii) the creation of a data schema from archetypes attributes, terminologies and constraints; iii) the development of a mechanism for automatic generation of templates using archetypes (i.e. Graphical user interfaces); iv) implementation of a REST API to make the data manipulation process transparent in the multi-model storage architecture; and finally, v) the demonstration of template generation based on public domain archetypes from the openEHR repository.

The remaining sections of this work are organized as follows: Section 2 contextualizes the concept of archetypes and polyglot persistence used in this paper, and surveys related works. Section 3 describes the characteristics and functionalities of the framework hereby proposed and demonstrates the construction of templates from extracted archetypes. Finally, the conclusion is found in Section 4.

2 Background and related work

This section contextualizes the basic concepts of archetypes (Section 2.1) and polyglot persistence (Section 2.2) and presents a comparative analysis of related works.

2.1 Archetypes and templates

The specification of archetype-based EHR consists in organizing its components through a two-level modeling approach [5]. The first, called information, represents the stable level of the model from which systems and software components can be built (e.g. XML schema, UML or OWL). At this level, data have no semantic characteristics and only the data types that are chosen to represent them are known. The second level called knowledge consists of domain-driven definitions represented as archetypes and templates.

An archetype consists of a computational expression based on a reference model and represented by domain constraints and terminologies [4] (e.g. data attributes of a blood test). A template is a structure used to group archetypes and allow their use in a particular context of application. It is often associated with a graphical user interface (e.g. a GUI used by a professional for defining the elements of a leukogram list, such as leukocytes and neutrophils). Dual modeling is the separation between information and knowledge of health care system architectures. In this approach, the components responsible for modeling the clinical and demographic data of EHR are specified through generic data structures, which are composed of data types, constraints and terminologies.

In an archetype, the specification of attributes is achieved through data entry builders named generic data structures. Such structures allow the representation of EHR data heterogeneity through the following types: ITEM_SINGLE, ITEM_LIST, ITEM_TREE and ITEM_TABLE.

ITEM_SINGLE models a single data attribute such as a patient's weight, height and age. ITEM_LIST groups a set of attributes in a list. A patient's address containing number, street and zip code for example. ITEM_TREE specifies a hierarchical data structure that is logically represented as a tree. It can be used, for instance, to model a patient's physical or neurological evaluations. Finally, ITEM_TABLE models data elements by using columns for field definition and rows for field value respectively. Each attribute of a data structure is characterized by a type of data and can have a related set of associated domain restrictions and terminologies. The terminologies give semantic meaning to clinical data and can be represented as a set of health terms defined by a professional.

2.2 Polyglot persistence

Polyglot Persistence defines the use of different data storage technologies to deal with different storage needs [10]. Different types of data are better represented by different storage approaches. The core idea in using polyglot persistence is the storage of structured data through a relational approach, while semi-structured or non-structured data is stored in NoSQL data models. Figure 1 shows how the different types of data from a health care sector can be stored using polyglot persistence.

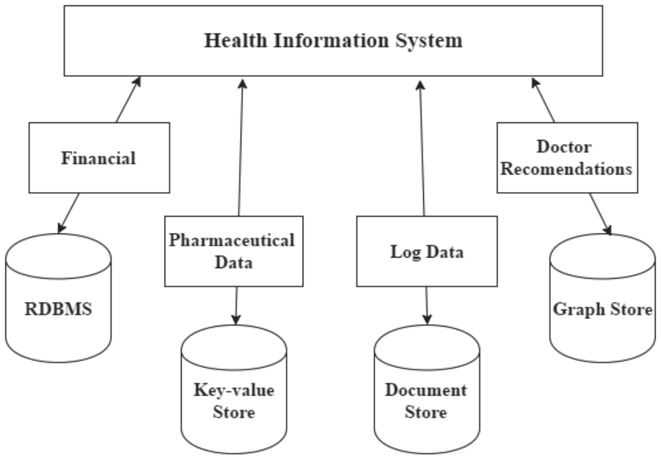


Figure 1. Example of multi-model storage in the health care sector

As shown in Figure 1, the NoSQL approach offers different data models. Among the most common are: Key-value store, Document store e Graph Store.

Key-value store: Stores information in a table, with rows, keys and value. The key field displays the information description while the value field denotes the information itself.

Document store: Stores document sets. A set consists of a collection of fields that can be displayed as a single element, a list, or nested documents.

Graph store: It makes use of nodes, relationships and properties to store information. The node represents the vertices in a graph, the relationships, its edges and the properties, the attributes.

The NoSQL data models have characteristics that differentiate them from more traditional approaches. Such characteristics offer a more adequate support for non-conventional data storage and more flexibility when creating or altering a data schema.

2.3 Related work

Based on the state-of-the-art works reviewed, we present an analysis of the main related works in the fields of i) archetype mapping and persistence in databases, ii) generation of templates from archetypes, and iii) the use of polyglot persistence in health applications.

One of the pioneering works in the field of archetype mapping was developed by Späth and Grimson (2010) [13]. They mapped a set of archetypes for a legacy database and exposed the lack of tools and methodologies that would have helped in modeling archetypes in a database. Using an automatic approach, Georg et al. [14] specified a set of rules to map archetype data attributes in a relational database, generating templates in a specific problem domain. However, the proposed solution could not map archetypes with hierarchical data structures (i.e. ITEM_TREE).

The openEHR foundation offers a free solution in which the data attributes of an XML archetype are serialized in a Database Management System (DBMS). Node+Path [15] uses Entity- attribute-value (EAV) approach to store the path (i.e. address) of an attribute in the first column, while the value of such attribute is stored in the second column. Yet, this solution requires the creation of complex logical sentences for data manipulation that may compromise the performance of the application. The persistence solution proposed by Wang et al. (2015) [16], specifies a set of mapping rules, extracts the archetypes attributes and stores them in tables of a relational data schema. However, the terminologies and constraints specified in those archetypes are not considered.

The EHRScape[1] and EHRGen[2] frameworks support the health application development process by using specifications from openEHR and generate templates and data schema from archetype mapping. However, they use a single data model to organize EHR data.

To minimize the rigidity caused by relational data schema and provide support to the continuous data requirement changes which commonly occur in legacy HIS, Prasad and Sha (2013) [12] specify an architecture and a HIS prototype that allows polyglot persistence and improves health data management in a legacy application. Similarly, Kaur and Rani (2015) [10] specify a polyglot storage architecture to store structured data in a relational database (i.e. PostgreSQL), while two NoSQL databases (i.e. MongoDB e Neo4j) store semi-structured data, such as laboratorial exams and medicine prescriptions. Nevertheless, neither solutions of polyglot persistence use archetypes to standardize EHR data attributes and terminologies.

Table 1 depicts the main characteristics of related works and the framework we are proposing. In this assessment, we have compared the following aspects A1) Reads an archetype and outputs an open standard; A2) generation of data schema from archetypes; A3) polyglot persistence of archetypes; A4) template generation; and A5) HIS architecture for heterogeneous databases.

Table 1. Comparative analysis of related work

Work/Criterion	A1	A2	A3	A4	A5
Späth and Grimson (2010)	✗	✗	✗	✗	✗
Georg et al.(2013)	✗	✗	✗	✓	✗
openEHR Node+Path	✓	✗	✗	✗	✗
Wang et al.(2015)	✗	✗	✗	✗	✗
EHRScape Framework	✓	✓	✗	✓	✗
EHRGen Framework	✗	✓	✗	✓	✗
Prasad and Sha (2013)	✗	✗	✗	✗	✓
Kaur and Rani (2015)	✗	✗	✗	✗	✓
PolyEHR Framework	✗	✓	✓	✓	✓

Only the Node+Path solution [15] and the EHRScape framework offer resources that allow mapped archetype attributes to persist in other solutions, whereas data schema creation from archetypes is only present in EHRScape, EHRGen and PolyEHR frameworks. The EHR polyglot persistence resource built from archetypes is possible only in PolyEHR framework, while template generation is possible in EHRScape, EHRGen, PolyEHR and in the solution proposed by Georg et al. Finally, PolyEHR and the works of Prasad and Sha (2013) [12] and Kaur and Rani (2015) [10] specify HIS architectures that exemplify polyglot persistence.

As shown in Table 1, the main motivation for the proposed solution is to develop a framework capable of building health application templates, generating data schemas in heterogeneous databases using archetypes and storing EHR data through polyglot persistence.

3 The PolyEHR framework

This section describes the proposed archetype-based framework and is organized as follows: Section 3.1 details the storage architecture in heterogeneous databases, while the main features of PolyEHR are presented in Section 3.2. Section 3.3 describes how the archetypes are displayed in heterogeneous data models. Finally, Section 3.4 shows the generation of templates and exemplifies the multi-model storage through polyglot persistence.

3.1 Architecture and overview

The framework proposed in this paper is built for health care applications and consists in a computing environment dedicated to the construction of templates from the specifications existent in EHR archetypes. The core feature consists in extracting the elements from archetypes for persistence in different data approaches, as well as the automatic generation of GUIs. Figure 2 illustrates the components and architecture relationships designed for PolyEHR.

1 https://www.ehrscape.com/
2 https://code.google.com/p/open-ehr-gen-framework/

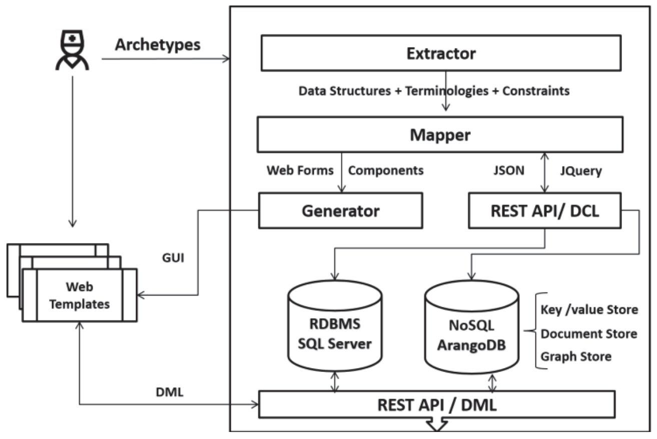


Figure 2. Framework architecture for multi-model storage.

One of the main features of the framework is to help health professionals build templates specific to their sector. As shown in Figure 2, the framework generates data schemas and GUI by reading an archetype imported by the user.

The *Extractor* component shown in Figure 2 extracts from the informed archetype the attributes that define the EHR, the health care terminologies and vocabulary that give a semantic meaning to the clinical data as well as constraints specified in the attributes. With the extracted elements, the *Mapper* component performs the following operations: i) transforms the attributes into data entry fields in the GUI; ii) uses the constraints extracted from archetypes as data entry validation mechanisms (e.g. range of values, data type restrictions); and iii) provides the terminologies extracted from archetypes to give a semantic meaning to their respective GUI fields. Afterwards, the *Generator* component groups and organizes the created elements, subsequently providing a usable template.

After the *Extractor* component obtains the attributes, terminologies and constraints, the REST API / DCL dynamically creates data schemas in the databases, both relational and NoSQL. Every GUI created by the framework has data manipulation capabilities (i.e. insert, update, query, and delete). Since the framework storage is based on heterogeneous data models, we specify a REST API that makes the process of data manipulation transparent to the user. Thus, all data persistence processes carried out in the template generated by the framework are managed by the REST API / DML, as shown in Figure 2.

3.2 Main functionalities

In addition to the data schemas generation feature in heterogeneous databases, we provide a host of features to help a health care professional use the templates generated by our framework. Such features are shown in Figure 3 and detailed below:

Demographic information management:

In order to reduce the demographic information redundancy in the generated data schema, we provide the management of health care organizations, patients, doctors, nursing staff and system users. Based on the type of organization (e.g. hospital, clinic, clinical laboratory, basic health unit), the framework generates instances of applications, i.e. an instance of a dedicated application for the admission of patients at a given hospital, or an instance of the application focused on outpatient and emergency care. In addition, information from patients, doctors and nursing staff can be integrated with features generated from the archetypes.

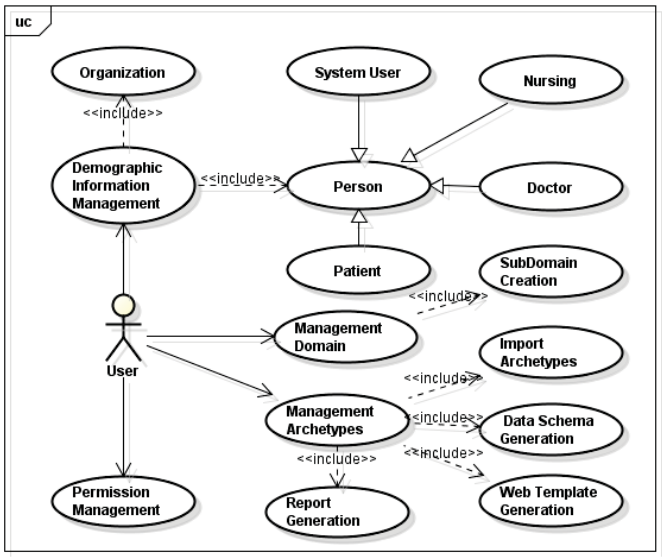


Figure 3. Main functionalities of the PolyEHR framework

Health care sector management: An organization can offer different types of health services. For example, a hospital may perform laboratory testing services, diagnostic imaging, emergency care, hospitalization, etc. Considering that, we have made it possible to create and configure domains and sub-domains that represent the services offered by each organization. In order to facilitate its use, the features generated by the framework are grouped into themes. Additionally, these features can also create a sequence of activities that shall be performed by health professionals; for example, an application may have a domain called *Nursing*, where professionals in this area have a determined drill to carry out, including assessment of patients' vital signs, water balance and physical examination.

Archetype management: This functionality allows one to import and map the archetypes that will be used to generate templates for a health application. When importing an archetype, the framework allows the professional to choose which elements will be part of the data schema and manage the GUI elements by adding, removing or disabling fields. Such elements can be modified at a later time. In this case, the framework will automatically extend the database schema created. After this activity, the user can then associate the generated GUI to a domain or sub-domain application and manage user and organization access permissions for each functionality. In addition to the GUI automatic generation, a mechanism allows users to create their own reports from the information stored in the data schema.

3.3 Persistence of archetypes in heterogeneous data models

The polyglote persistence proposed is this framework is performed as follows; data of a structured nature such as demographic information is stored in a relational database (i.e. SQL Server), while non-structured data such as a laboratory exam is stored in a NoSQL database. The generation of NoSQL data schemas is based on the type of data structures found in archetypes, such as ITEM_SINGLE, ITEM_LIST, ITEM_TREE and ITEM_TABLE.

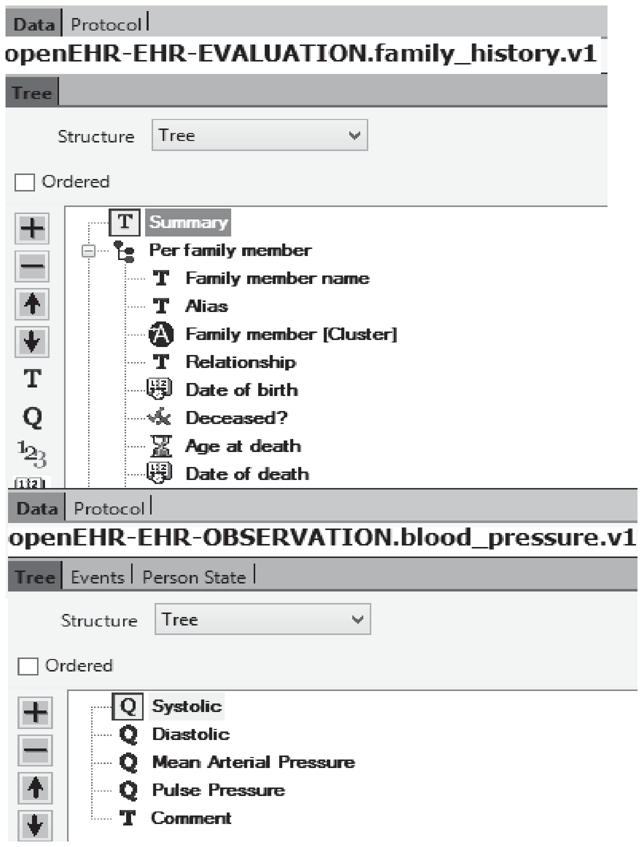
As the name suggest, ITEM_SINGLE displays a single attribute, while ITEM_LIST groups a set of attributes into a vertical list. Due to the nature of these items, we propose the key-value data model to generate data schemas. ITEM_TREE displays data attributes in a hierarchical structure containing the several levels of an archetype. In order to maintain such data organization, the framework uses document data model (i.e. collections). Finally, ITEM_TABLE organizes the data attributes in a table made of rows and columns. For this type of storage, we propose the graph data model, where data found in rows are represented as properties, data from columns as vertices, and attribute description as the relationship between the two.

3.4 Generating templates for health applications

In this section, we demonstrate how the framework generates templates and exemplify how the data is persisted in heterogeneous databases through polyglot persistence. Here, we will use the family history and blood pressure archetypes to generate templates and data schemas. The blood pressure archetype was chosen to demonstrate that we obtained the same results using EHRScape and EHRGen, as well as the approach proposed by Georg et al. [14]. The family history archetype was in turn chosen because it contains a significant amount of attributes represented by a multilevel hierarchical structure. Both these archetypes are shown in Figure 4 and are available in the openEHR repository [17].

We initially registered a fictitious health facility named *Model Hospital* along with some patients, doctors, nursing staff and system users to manipulate data in the templates. Before importing the family history and blood pressure archetypes, we created a domain named *Ambulatory* and two sub-domains named *Family Background* and *Patient assessment*, which will then be linked to their respective GUIs.

Afterwards, the user can choose to import the archetypes that will be used for the generation of data schemas and GUIs. To customize importation, we have developed a feature that allows the user to choose archetypal elements, and visualize how they were mapped by the framework with their respective data types. Furthermore, data input for each field can be made mandatory or not.



openEHR-EHR-EVALUATION.family_history.v1

Structure: Tree

Ordered

- Summary
- Per family member
  - Family member name
  - Alias
  - Family member [Cluster]
  - Relationship
  - Date of birth
  - Deceased?
  - Age at death
  - Date of death

openEHR-EHR-OBSERVATION.blood_pressure.v1

Structure: Tree

Ordered

- Systolic
- Diastolic
- Mean Arterial Pressure
- Pulse Pressure
- Comment

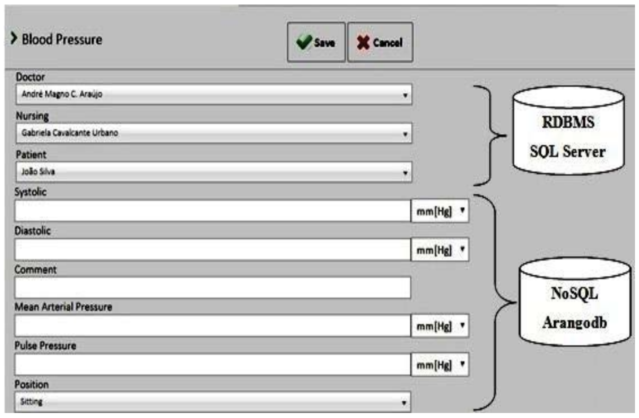
Figure 4. Data attributes of family history and blood pressure archetypes [17]

Once the user inputs all information, the framework dynamically generates the data schemas from the elements extracted from the archetypes (i.e. attributes, terminologies and constraints) following the criteria described in Section 3.3. To enable the functionality which was just generated, the user simply needs to associate the GUI to a domain and sub-domain previously configured in the framework.

To the GUI generated from the blood pressure archetype, we added demographic information managed by the framework. Therefore, beyond the chosen archetypal elements, the GUI will have the following additional fields: the name of the doctor responsible for clinical care, the nurse responsible for the exams and the actual patient.

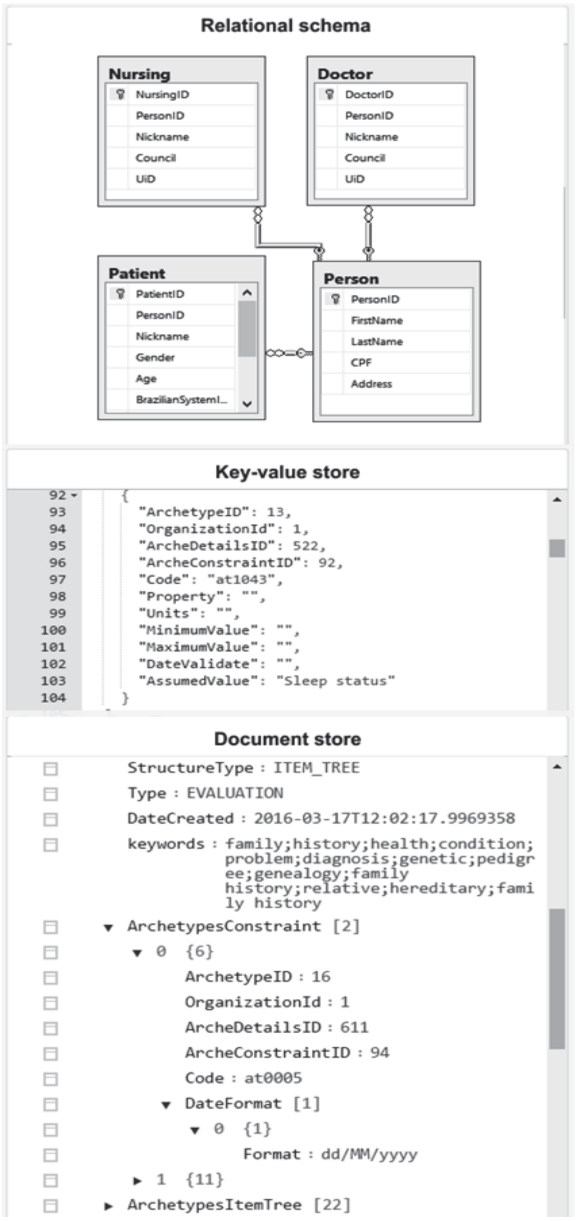
Figure 5 depicts the GUI created from the blood pressure archetype. One may observe that the demographic information (i.e. Doctor, Nursing and Patient) is stored in a relational database, while the other archetype fields are stored in a NoSQL database. Since the structure that defines the attributes of the blood pressure archetype is an ITEM_LIST type, the framework persists the data in a key-value database.

Figure 6 shows parts of the relational and NoSQL data schema (key-value and document) used to store the data templates generated from the blood pressure and family history archetypes.



Doctor: André Magno C. Araújo
Nursing: Gabriela Cavalcante Urbano
Patient: João Silva
Systolic: mm[Hg]
Diastolic: mm[Hg]
Comment:
Mean Arterial Pressure: mm[Hg]
Pulse Pressure: mm[Hg]
Position: Sitting

Figure 5. Example of a template generated by our framework based on the blood pressure archetype



Nursing	Doctor	Patient	Person
NursingID	DoctorID	PatientID	PersonID
PersonID	PersonID	PersonID	FirstName
Nickname	Nickname	Nickname	LastName
Council	Council	Gender	CPF
UiD	UiD	Age	Address
		BrazilianSystemI...	

```
{
  "ArchetypeID": 13,
  "OrganizationId": 1,
  "ArcheDetailsID": 522,
  "ArcheConstraintID": 92,
  "Code": "at1043",
  "Property": "",
  "Units": "",
  "MinimumValue": "",
  "MaximumValue": "",
  "DateValidate": "",
  "AssumedValue": "Sleep status"
}
```

```
StructureType : ITEM_TREE
Type : EVALUATION
DateCreated : 2016-03-17T12:02:17.9969358
keywords : family;history;health;condition;
           problem;diagnosis;genetic;pedigr
           ee;genealogy;family
           history;relative;hereditary;fami
           ly history
▼ ArchetypesConstraint [2]
  ▼ 0  {6}
      ArchetypeID : 16
      OrganizationId : 1
      ArcheDetailsID : 611
      ArcheConstraintID : 94
      Code : at0005
    ▼ DateFormat [1]
      ▼ 0  {1}
          Format : dd/MM/yyyy
  ► 1  {11}
► ArchetypesItemTree [22]
```

Figure 6. Heterogeneous database storage

As the framework generates templates that store data in a heterogeneous database, we had to develop a mechanism to dynamically create the data manipulation resources for each GUI generated. Such resources are input, edit, print, delete and retrieve information from the databases. Here, the REST API encapsulates the heterogeneity of manipulation languages used by the adopted data models (i.e. SQL and AQL.) The template shown in Figure 5 and all other features described in this article are available at archeweb.dotdesign.com.br.

4 Conclusion

In this paper we presented a framework capable of building application templates from archetypes to be used in the health care sector, generating archetype-based data schemas in heterogeneous databases and storing EHR data through polyglot persistence.

We developed an approach to generate graphical user interfaces with data manipulation capabilities and specified a mechanism to extract attributes, terminologies and constraints from archetypes and allow the dynamic persistence of data in heterogeneous databases. To help a health care professional use the templates generated, PolyEHR provides a host of features such as the management of health care sector, archetypes and demographic information. In addition, all data persistence processes carried out in the template generated by PolyEHR are managed by a REST API. Finally, we demonstrated the generation of health application templates based on public domain archetypes from the openEHR repository.

Acknowledgment

This work was partially supported by Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE), under the grants APQ-0173-1.03/15 and IBPG-0809-1.03/13.

5 References

[1] K. k. Lee, W. Tangb, K. Choia. “Alternatives to relational database: Comparison of NoSQL and XML approaches for clinical data storage”; Computer Methods and Programs in Biomedicine, pp. 99-109, 2013.

[2] V. Dinu, P. Nadkarni. “Guidelines for the Effective Use of Entity-Attribute-Value Modeling for Biomedical Databases”; International Journal of Medical Informatics. pp. 769-779, 2007.

[3] International Organization for Standardization: ISO/TS 18308 health informatics - requirements for an electronic health record architecture, available at http:// www.iso.org/iso/iso_catalogue.htm.

[4] Marco E., Thomas A., Jorg. R, Asuman D., Gokce L. “A Survey and Analysis of Electronic Healthcare Record Standards”; ACM Computing Surveys, pp. 277–315, 2005.

[5] D. Lloyd, T. Beale, S. Heard. “openEHR Architecture: Architecture Overview”; available at http://www.openehr.org/releases/1.0.2/architecture/overview.pdf, last accessed September 2015.

[6] Jumaa H, Rubel P, Fayn J. “An XML-based framework for automating data exchange in healthcare”; IEEE International Conference on e-Health Networking Applications and Services (Healthcom), pp.264 – 269, 2010.

[7] Bernstein K, Bruun R. M, Vingtoft S, Andersen S. K, Nøhr C. “Modelling and implementing electronic health records in Denmark”; International Journal of Medical Informatic, pp. 213-220, 2005.

[8] Seetharami R. Seelam, Paolo Dettori, Peter Westerink, Ben Bo Yang. “Polyglot Application Auto Scaling Service for Platform As A Service Cloud”; IEEE International Conference on Cloud Engineering, 2015.

[9] Martyn Ellison, Radu Calinescu. “Richard Paige, Re-engineering the Database Layer of Legacy Applications for Scalable Cloud Deployment”; IEEE/ACM 7th International Conference on Utility and Cloud Computing, 2014.

[10] Karamjit Kaur, Rinkle Rani. “Managing Data in Healthcare Information Systems: Many Models, One Solution”; IEEE Computer Society, 2015.

[11] Rami Sellami, Sami Bhiri, Bruno Defude. “Supporting multi data stores applications in cloud environments”; IEEE Transactions on Services Computing, 2015.

[12] Srikrishna Prasad, Nunifar Sha, “NextGen Data Persistence Pattern in Healthcare: Polyglot Persistence”; Fourth International Conference on Computing, Communications and Networking Technologies, 2013.

[13] Späth M. B., Grimson J. “Applying the archetype approach to the database of a biobank information management system”; International Journal of Medical Informatics, pp. 1-22, 2010.

[14] D. Georg, C. Judith, R. Christoph. “Towards plug-and-play integration of archetypes into legacy electronic health record systems: the ArchiMed experience”; BMC Medical Informatics and Decision Making, pp. 1-12, 2013.

[15] Node + Path Persistence, available at https://openehr.atlassian.net/wiki/pages/viewpage.action?pageId=6553626, last accessed April 2016.

[16] Wang L, Min L, Lu X, Duan H. “Archetype relational mapping - a practical openEHR persistence solution”; BMC Medical Informatics and Decision Making, pp. 1-18, 2015.

[17] Archetype Editor, available at www.openehr.org, last accessed April 2016.