SUMMARY
Large-scale adoption of electronic healthcare applications requires semantic interoperability. The new proposals propose an advanced (multi-level) DBMS architecture for repository services for health records of patients. These also require query interfaces at multiple levels and at the level of semi-skilled users. In this regard, a high-level user interface for querying the new form of standardized Electronic Health Records system has been examined in this study. It proposes a step-by-step graphical query interface to allow semi-skilled users to write queries. Its aim is to decrease user effort and communication ambiguities, and increase user friendliness.

key words: query interfaces, database query languages, electronic health records, healthcare, archetype-based EHR

1. Introduction

Health care industry is adopting use of information technologies on a large scale. It aims to generate mechanisms and standards for sharing health information. The openEHR foundation, CEN/TC251 (European Committee of standardization) and HL7 aim to generate mechanisms and standards for sharing health information worldwide [9], [22], [23].

1.1 Complexity of EHR Components

Electronic Health Records (EHRs) have a complex structure that may include data of about 100-200 parameters, such as temperature, blood-pressure and body mass index. Individual parameters have their own values/variables (Fig. 1).

In order to provide an information interchange platform for EHRs, it has been proposed to use archetypes [8]. For different parameters in EHR, the archetypes are distinct, structured models of domain content, such as ‘data’ (e.g., captured for a blood pressure observation). It contains complete knowledge about a clinical context, i.e., attributes of data, ‘state’ (context for interpretation of data), and ‘protocol’ (information regarding gathering of data) as shown in Fig. 1.

Traditional clinical software development is led by professional IT developers, who represent the clinical concepts within the database structure itself. The new approach to EHR development is known as two-level modeling [8], which involves the separation of clinical knowledge (archetypes) from the information or recording model (reference model). It has been approved as a standard by ISO [2]. According to the modelling, the Reference Model (RM) [1] represents the generic structures of components of health record information, at storage level. The primary purpose of archetypes is to provide a reusable, interoperable way of managing generic data so that it conforms to specified structures and semantic constraints. Every archetype is written with respect to a particular ‘RM’. Thus, archetypes are expressed in the form of constraints on data whose instances conform to a RM. These are described by using Archetype Definition Language (ADL) [21]. It provides a solid formalism for expressing, building and using these entities, computationally. The formalism is adapted between internal layers of knowledge resources in a computing environment (such as, clinical terminologies and ontology) to facilitate interoperability and use of actual data in production systems. This reflects the stable characteristics of the design of Electronic Health Records (EHRs) and their components.

1.2 Requirement of Graphical Query Language

Our work deals with querying these standardized EHRs, where there is complete segregation of information from knowledge. Research to date has shown that there is no easy-to-use Query Language (QL) interface for healthcare users. The study examines the use of graphical query language interface for these semi-skilled users.

Query-by-example (QBE) is the first application as a graphical query language [10]. QBE is both the language name and the system name of the early application by IBM
2. Problem Background

2.1 Clinical Data Sets in the OpenEHR System

The openEHR foundation [9] is pioneering the field for maintaining semantic interoperable EHRs. Recently, Microsoft’s Connected Health Framework included openEHR (ISO 13606-2) archetypes as part of its domain knowledge architecture for interoperability [6]. The similarity of DBMS architecture [10] and the openEHR architecture [8] has been explored (Appendix A). There are three levels. First is physical level. It is the lowest level of abstraction that describe details of reference model such as identification, access to knowledge resources, data types and structures, version control semantics, support for archetyping and semantics of enterprise level health information types. Second is logical level, which contains logical descriptions. These can be represented in the form of archetypes and templates, through archetype model (AM) [3]. The implementation of clinical concepts may involve physical-level structures (user of logical level does not need to be aware of this complexity). Clinical domain experts use logical level. The third is view level, which is the highest level of abstraction. It describes user view for a part, or the entire EHR architecture. This corresponds to the service model. Several views can be defined. The users access these views. The clinical domain concepts can be expressed independent of how the data is actually stored in a relational database. Thus, it is possible to modify the healthcare concepts, i.e., the archetypes and their instances without changing the underlying relational tables.

2.2 Problem Statement

The openEHR specifications have been designed to support services in a service-oriented architecture (SOA) [8]. These services include the ‘EHR’ and ‘Demographics’ at present with means of accessing other key resources such as Terminology. Further development of the platform API is necessary for many initiatives. A service level specification is also being sought. The role of a high-level query language interface in such a platform approach is a powerful and useful requirement.

The RM and AM are well-defined and implemented, but the service model layer is under implementation. Although EHR data would be stored in standardized model, querying data has not been well standardized or defined yet. Archetype Query Language [4] is not useful to end-user (clinicians). The current research focuses on providing high-level support implemented at service layer as shown in Fig. 2.

The openEHR provides a Clinical Knowledge Manager (CKM) [11]. It is a domain knowledge governance tool. As per CKM, there are 279 numbers of archetypes. With CKM, the users interested in modeling clinical content can participate in the creation and/or enhancement of an international set of archetypes. Thus, these archetypes (knowledge layer) will be used globally among EHR systems and ensure semantic interoperability.

The authors use the standardized representation of an international electronic health record, the abstract specifications of which are defined using the UML notation and formal textual class specifications [8]. The information layer is shown in Fig. 2.

3. Need for AQBE Approach

The standardized EHRs database architecture may contain many entities. These include EHRs of patients. These may include non-clinical contents (relating to hospital information, including accounting, wards, and doctors.) There is a lot of complexity in structure of an EHR, and the components of an EHR system. At the same time, a query interface is required to support users at varying levels of query skills. These include semi-skilled users at clinics or hospitals.

3.1 Complexity in Modeling

EHR data may include many parameters about clinical status of a patient. Their contents may be structured, semi-structured or unstructured, or a mixture of all three. The data can be quantities (such as ratio and proportions, dimensioned quantity, reference ranges), date/times, plain and coded text, time specification, multimedia and uniform resource identifiers (URIs). The information is organized in hierarchical form. According to ISO 13606-1 Reference Model [2], the EHR extract (root object) contains EHR data as compositions (may be organized by folder hierarchy). Composition contains ENTRYs (may be organized by section hierarchy). ENTRYs contain ELEMENTS (may be contained within a cluster Hierarchy) (Figs. 3 and 4).

Service Layer

| High-Level interface AQBE (Query Language) |
|--------------------------------------------|
| Knowledge Layer (represented in ADL)       |
| Information Layer (available in UML)       |
| Data Base                                  |
| Terminology                                |

Fig. 2 High-level support at service layer.
3.2 Details of Schema

Complex Schema knowledge is required while querying EHR system using Archetype Query Language [4]. According to openEHR reference model [1], there are five sorts of entries (observation, evaluation, action, instruction, administrative) and four types of elements—single entities (e.g., weight, height), lists (e.g., blood test results), tables (e.g., visual acuity results) and trees (e.g., biochemistry results).

Thus, archetypes belong to any one among the following categories (composition, section, observation, evaluation, action, instruction, administrative, single, list, table and tree) (Fig. 3). Each category has a different structure. An EHR consists of many archetypes which belong to different category. The user is unaware of all these. He/she should be presented with an easy-to-use interface.

3.2.1 Archetype Definition Language (ADL)

ADL [21] (approved by ISO) is a formal language for expressing archetypes in EHRs. The Archetype Object Model (AOM) [8] describes the definitive semantic model of archetypes. The ADL approach makes use of existing UML semantics and existing terminologies, and adds a convenient syntax for expressing the required constraints. An archetype expressed in ADL is composed of four main parts: header, definition, ontology and revision history. The header section contains the archetype metadata. In the definition section, the modeled clinical concept is represented in terms of a particular RM class. This description is built by constraining properties of classes and attributes, such as existence, occurrences or cardinality or by constraining the domain of atomic attributes. In the ontology section, the entities defined in the definition section are described and bound to terminologies. Finally, the revision history section contains the audit of changes to the archetype.

Thus, modeling, detailed knowledge of schema and ADL hinder the medical user from querying.

4. Querying EHR System

The key challenges in querying EHRs are:

- Complex and domain-specific semantics,
- Frequent references to external information sources such as dictionaries and ontologies, and
- Special treatment of time and location attributes.

Archetypes can be used for the purpose of intelligent querying [8]. They ensure semantic interoperability. They contain domain-specific knowledge. They contain a ‘term binding’ section for mapping the internal terms to the external terminologies (such as SNOMED [34]). The internal structure of archetypes take care of time attributes. Therefore, the present study proposed to make efficient use of archetypes for querying.

It is important for healthcare professionals to examine data from a variety of perspectives. The aim of the proposed research activity is:

1. To provide content independent querying capability to clinicians; and
2. To design high-level interface for users, who are not skilled in use of database query languages.

4.1 Archetype Query Language (AQL)

The standardized EHR system supports a domain-specific query language which has been proposed for standardization [4]. The Archetype Query Language (AQL) is a declarative query language developed specifically for expressing queries used for searching data from archetype-based EHRs. It has neutral expression syntax, i.e., neutral to EHRs, programming languages and system environments. It depends on the openEHR archetype model and semantics. The openEHR specifications for AQL and EHR are based on an object-oriented framework. It was developed on the basis of many observations, namely, a set of clinical query scenarios, the study of the current available query language syntaxes (including XQuery, SQL and Object Query Language), and the study of the archetypes technology, openEHR RM and openEHR path mechanisms.

AQL was earlier named as EQL (EHR Query Language) [5] which has been enhanced with the following two innovations:

i) utilizing the openEHR path mechanism to represent the query criteria and the response or results; and

ii) using a ‘containment’ mechanism to indicate the data hierarchy and to constrain the source data to which the query is applied.

OpenEHR path mechanism enables any node within a top level structure to be specified from the top of the structure using a path compatible with semantic level and X-path.

The syntax of AQL is illustrated by the help of example. It uses SELECT, FROM and WHERE clauses. EHR uses a hierarchical structure. AQL has a containment constraint which specifies the hierarchical relationships between parent and child archetypes involved in query. It
Fig. 4 Concepts in hierarchical EHR structure.

Fig. 5 Syntax of AQL.

makes use of the path expression, naming retrieved results, the class expression and archetype predicate, as shown for the query example in Fig. 5. The class expression syntax consists of openEHR RM class name (mandatory), followed by variable name (optional), followed by an archetype predicate (optional). The archetype predicate is used to scope the data source from which the query expected data is to be retrieved.

AQL is used by developer level users (skilled users). Its syntax is complex. We explored in our earlier study that it requires more skills than SQL and XQuery, which are at application level [18].

Query Example: Find all Blood Pressure (BP) values encountered in a health encounter for a patient, having the systolic BP and diastolic BP, (where systolic BP \( \geq \) 140 or diastolic BP \( \geq \) 90).

4.2 AQBE approach for EHR System

At the present moment, there is no easy-to-use query language interface available for EHRs system.

Query-By-Example (QBE) [14] is a graphical user interface (GUI) based query language for non-professionals with little or no computer/mathematical background. It operates on relational data model. In contrast to linear type languages such as SQL, QBE offers to its users multi-degrees of freedom. For example, the sequence of filling in the tables and the rows within the tables is immaterial. It can handle complicated queries without relinquishing its simplicity. Similarly, a new AQBE interface has been proposed for meeting the querying need for the standardized EHR system.

The proposed AQBE interface depends on two former innovations i) and ii) in previous section (Sect. 4.1). The semi-skilled user may be provided with the AQBE approach for querying EHR data. They do not need to learn or use the AQL syntax. This approach takes into account the EHR structure (Figs. 4 and 6) and parameter (archetype) structure (Fig. 1). It presents these in the form of QBE, to enable the semi-skilled users to query the EHR system. The proposed AQBE approach involves two steps:

1. Determine the node(s) involved in the query from EHR structure tree. (For example, see BMI or BP in the following query scenarios 1 and 2).

2. For the selected node, a graphical QBE interface will be depicted with the help of multiple-level model (Appendix A). User can compose algebraic query expressions using the appropriate variables (operands) in condition box. User can put a “P” command in appropriate column(s) to specify the fields that should be shown in result.

In the proposed Multi-step QBE (AQBE) for the EHR system, XML data forms an intermediate layer of support between ADL and QBE interface. Different clinical documents may contain same or different set of concepts. A sample of clinical document is shown in Appendix B. The concepts present in the clinical document are shown in bold. Consider the following examples.

4.2.1 Projection and Selection Predicate Query

Query Scenario 1. Get all blood pressure (BP) values, where systolic value is greater than or equal to 140, or diastolic value is greater than or equals to 90, which are recorded in a health encounter for a specific patient.

AQBE approach: The sample of clinical document for the above query is shown in Appendix B. The proposed AQBE system presents an EHR in the form of tabular structure consisting of all the concepts present in clinical document (Fig. 7). In this case, the user selects “Blood Pressure”.

In step 2, for the selected node (Blood Pressure), Fig. 8 depicts QBE interface. The querying process is simplified by giving a QBE view for the complex structure of chosen archetype (Appendix C). The blood pressure archetype consists of data, state, protocol and event information as shown in Fig. 1 (see Fig. 8 with selected data part). Further steps include,
Compose algebraic query expressions using the appropriate variables (operands) in condition box.

Put a ".P" command in appropriate column(s) to specify the fields shown in result.

In contrast, the AQL syntax is complex (Fig. 5). It cannot be used by semi-skilled clinical users.

**Query Scenario 2.** Get the body mass index (BMI) values which are more than 30 kg/meter-square for a specific patient.

AQBE approach:
Step 1: User selects EHR–Composition–Entry–BMI
In step 1, the proposed AQBE system presents EHR structure (Fig. 6). The EHR is populated by repository of archetypes (mentioned in Sect. 3). The user selects “BMI”.
In step 2, based on the stored archetype structure in ADL; a QBE interface for BMI is presented to the user. At user’s level, query conditions and results are expressed using conventional QBE notations (Fig. 9). Subsequently, the query response is presented to the user.

In contrast, the AQL syntax requires complex steps. The AQL syntax for above query is shown below.

SELECT ae1 FROM EHR e
CONTAINS ADMIN_ENTRY ae1 [openEHR-EHR-ADMIN_ENTRY.admission.v1]
WHERE ae1/encounter_id/value not in
(SELECT ae2/encounter_id/value FROM EHR e
CONTAINS ADMIN_ENTRY ae2 [openEHR-EHR-ADMIN_ENTRY.discharge.v1])

AQBE approach:
In step1, the user selects admission and discharge from the EHR structure. In step2, the following QBE (Figs. 10 and 11) are prompted to the user. The symbol ¬ denotes negation.

**4.2.3 Rename Query**

**Query Scenario 4.** Retrieve all BP values, retaining for each BP their systolic and diastolic blood pressure values; also change the tagname of systolic BP as Sys and Diastolic BP as Dias.

For step 1, the user selects blood pressure. Figure 12 depicts the interface prompted to user. Figure 13 is the result table showing renaming operation.

**4.2.4 Existential Query**

**Query Scenario 5.** Return all BP elements having a posi-
tion in which the BP was recorded.
In step1, the user selects blood pressure. Figure 14 shows the AQBE interface for existential operation on chosen concept.

4.3 Detailed Design and Comparison with QBE

The EHR or part of EHR (XML document shown in Appendix B) consists of multiple archetypes. These archetypes have a ‘containment’ relationship among them because of the hierarchical nature of EHR (Fig. 3). Also, for each archetype (clinical concept), it has a very complex structure (Appendix C).

So, the first step of AQBE helps in depicting which concepts are present in the EHR or part of EHR. This is a user-defined view presented by our system in the form of table-like structure to ease the querying. In actual, the database does not contain any such table. We wish to provide this view to the user based on hierarchical structure of EHR (explained in Sect. 3). The user selects the desired concept (user’s object).

In the second step, the querying process is simplified by giving a QBE view for the complex structure of chosen archetype. In this view, the user can query very easily with fine level of granularity and specify the conditions. Compare the QBE view given for concept ‘blood pressure’ in Fig. 8 with the complex structure of ‘blood_pressure.adl’ file shown in Appendix C.

The proposal forbids using multiple QBE interfaces. It proposes a multi-step QBE. The following points illustrate the reason.

- Step 1 presents table which are not picked from database, hence making it different from the existing QBE approach. Despite this, a tabular view is generated dynamically and presented to the user. It consists of the concepts present in EHR, i.e., clinical document.
- Step 2 presents QBE structure by parsing the archetype (e.g., BP.adl file) which is not same as a database table. For example, the demographics archetype stores information in two tables. In AQBE, the demographics concept is being presented to the user inspite of two database tables named ‘patient’ and ‘contact_details’.

Thus, the user is presented with a high-level view independent of how the data is stored in an underlying database. In multiple QBE, we just select the right tables and fill the predicates. In AQBE, it’s not just choosing the right table and filling out predicates. The user is dealing with the high-level view of user’s object (concepts). These objects have internal attribute mapping functions to the actual data stored in a relational database, which users need not have to worry about.

5. Technical Description: Mapping AQBE to AQL

Our analysis found that AQL has the basic structure based on the query block or subquery like SQL with few differences mentioned below, for the following expression.

\[
\text{SELECT target-list FROM relation WHERE predicate}
\]

i) The target-list is an identified-path expression of the data-item;
ii) The ‘FROM’ clause consists of ‘CONTAINS’ clause to specify the complete hierarchy of data source. E.g., for querying blood pressure, the user require to know the complete hierarchy

\[
\text{EHR \rightarrow COMPOSITION \rightarrow OBSERVATION}
\]

It also requires knowing the archetype predicate (Fig. 5); and
iii) The predicate is in the form of identified-path expression <condition><value>.

These findings lead to translation algorithm for AQBE to AQL through following steps. The input to the algorithm is AQBE and the output is AQL query.

1. **Naming transformation**: The chosen concept name is transformed to archetype predicate.
2. **Hierarchy generation of data source**: AQBE transformer automatically generates the containment mechanism. For implementation purpose, we store this as a schema in the form of look-up table. The first two steps help in the construction of ‘from’ clause of AQL.
3. **Identified-path generation**: AQBE generates internally the identified-path expression from the archetype (ADL file), for the chosen item (at a fine granularity of data) within any concept. For the implementation purpose, the ADL parser is used. This builds the ‘select’ clause of AQL.
4. **Process ‘predicates’**: The user is allowed to specify the conditions and value in the tabular structure. This is subsequently added to the ‘where’ clause.
5. **Process ‘relate’**: If the user requires relating two concepts, a recursive invocation of the translation algorithm takes place.

For simplicity, the user neither needs to write the archetype predicate/identified path expression, nor requires specifying the complete data source hierarchy. Thus, AQBE has a query level, which is nearly same as the AQL level. As illustrated, AQBE facilitates the task of user by selecting a clinical concept and prompting the user, instead of learning the AQL syntax.

The AQL syntax has on optional ‘group-by’ and ‘having’ clause. These would be incorporated in future in the translation algorithm.

6. Implementation Details and Related Work

The desired implementation can be realized (as shown by the block diagram in Fig. 15). It shows the high-level view of the end-to-end AQBE request processing. A GUI is pro-
provided to the client for AQBE input and output operations. The AQBE request is transformed and processed by the AQBE Transformation and AQL Processing Engine. The transformation of AQBE to AQL is done by the transformer. Subsequently, it is parsed and validated by the AQL Parser component and upon successful validation; the request is forwarded to the AQL Processing engine for the query execution. The application layer of the openEHR database is exposed through either hibernate (object relational mapping) or ODBC interfaces. This interface communication is handled by the AQL Processing Engine at the time of actual AQL execution. Therefore, the AQBE Transformation and AQL Processing Engine acts as a mediator between the novice client and openEHR database. This approach significantly abstracts the low-level AQL complexities from the end user.

As explained in Sect. 3, the RM specifies EHR Extract as the root object which contains the EHR data. For the step 1, in proposed query system, the EHR structure is built from the available UML description of the EHR and archetypes [13]. In step 2, for parameters (or archetypes) in EHR, the data available in ADL [12], is used to create individual QBE interfaces (Figs. 8 and 9). This QBE interface is presented for semi-skilled users.

6.1 AQBE Prototype System

AQBE prototype has been developed as a client-server application. The implementation uses the Java 2 Platform, Enterprise Edition (J2EE). The server part runs on an Apache Tomcat application server [25]. The server is developed with "SAStruts" (Framework for Java Servlet and JSP) [28]. Eclipse has been used as the integrated development environment (IDE) [26]. To get clinical knowledge (information) from archetype, the "ADL parser", implemented in the openEHR Java Reference Implementation Project is being used [27].

An experiment has been conducted for a sample of queries for various operations such as select, project, join and negation. QBE is a relationally complete language [14]. The study infers AQBE to be relationally complete language as it is built as an extension to QBE. The QBE approach is extended to be used for archetype-based EHR data. For querying, QBE gives archetype as a view which provides user-defined subset of a large database.

The feasibility of AQBE as complete approach can be enhanced by work in literature [29], [30]. Thus, AQBE incorporates the feature of a visual query language. It can be considered as expressive, complete and user-friendly. However, the usability studies have to be done in future for clinical users with no query language skills.

6.2 Technical Challenges and Solutions

The first challenge in querying archetype systems is the determination of archetypes that will be involved in querying. This is required for building of the ‘containment’ clause. The second challenge is that different archetypes (clinical concepts) can belong to different categories (Sect. 3). E.g., Querying on Blood Pressure (BP) requires the use of two archetypes namely: encounter archetype (belonging to ‘Composition’ category of RM) and BP archetype (belonging to ‘Observation’ category of RM). However, at the user level, for querying data regarding BP, the user knows BP as a parameter and will query only that parameter.

The solution to the first challenge is provided by giving a QBE like structure for the archetypes involved in querying. The second challenge is overcome as follows. For each archetype in the list, a different presentation model according to each category of archetype is being built. This is done using the XSLT excerpt for each category (eleven categories). Thus, all the concepts (279 developed till date) can be represented by using these small number of presentation models.

6.3 Related Works and Discussions

EHR systems are similar to GPS, in that they provide “locations” of patient information, as well as care pathways. At the present moment, little or no research has been done on the high-level QL for archetype-based EHR systems. It is necessary, that archetypes (in combination with terminologies) enable powerful possibilities for semantic querying of repository data (for individual usage/population research). Archetype-based querying enables true longitudinal processing of health data, regardless of the originating system. Thus, the study has presented a proposal for enhancing the query capability of EHR workers. For improvements in query interfaces, many possibilities have been examined.

An alternative approach proposed by Ocean informatics [17] suggests using a query builder tool, to construct AQL query. It requires - form related inputs and more skills on the part of the user.

The medical users are not skilled in the use of a QL. XQuery is complicated for hospital users. XQuery requires extensive knowledge of document structure (its XML schema) in order to formulate a query. The alternative can be forms and natural language query interface. Forms are simple, but limited and can only do a fixed kind of work. The other challenge is how to map the user intent to database schema. It is difficult to understand the user intent given an arbitrary natural language query interface.

The challenge in healthcare domain is the unknown
schema. The solution can be schema-free XQuery [31], but the end-users are still uncomfortable in using it because of the requirement of partial knowledge of schema.

Also, because of the rapid expansion of healthcare knowledge, the domain is becoming very complex. For example, the information model given by openEHR and HL7 has a very complex schema for representing the health record [1], [32]. One solution proposed may be schema summarization [33]. It represents the original complex schema with a smaller and conceptually simpler schema. It helps the users explore the schema, still schema knowledge is required.

Other high-level interface approaches, Query-by-Object [15], [16] and Query-by-Template [24] have been considered, before choosing the AQBE interface. Alternative approaches, such as use of mind maps of archetypes [11] are also examined. Natural Language interfaces help users avoid the burden of learning logic-based language. But, these are difficult to build. These also suffer from problems of linguistic variability and ambiguities. Similarly, the search engine can be used on the top of database approach, but these suffer from the lack of returning precise and accurate answers [19]. Thus, these approaches provide inadequate support to express a query.

It has also been examined to use XQBE [7] interface directly over the XML description of archetype [20]. It is found that XQBE interface requires some knowledge of trees or graphs on the part of users, whereas QBE interface is quite simple and intuitive to use. Also, it was found that some training is required to learn the graphical elements of XQBE, whereas AQBE provides multi degrees of freedom to its users. It facilitates query expressions for the users. It presents maximal content in order to include all types of requirements. For example, the archetype for blood pressure (Fig. 1) (It can be used in multiple places, wherever it is required within an EHR). The EHRs facilitate the archiving of data over the lifetime of patients. Although, depending upon the requirements, each medical application adopts different processes and procedures. Thus, AQBE QL interface provides useful insights for complex queries raised by medical researchers.

In simpler terms, the research hypothesis can be stated as, “there has been a quantum jump in the evolution of EHRs. It is possible to find QLs that are similar to database QL for querying archetype-based EHRs”. An experiment to use AQBE has been performed for a sample set of queries for various database operations. A comparative analysis of the existing and the proposed QL has been obtained.

6.4 Advantages of Proposed Approach

- It does not require learning the query language syntax.
- It gives a high-level view despite of how data stored in database.
- The user need not know about the knowledge of the complete hierarchy of archetypes, which are involved within the query. Such knowledge is required in the AQL (FROM clause). Thus, the user is provided with his own (user) level view of concepts.
- It provides an easy interface view (Figs. 8 and 9), in spite of the complex ADL file structure (Appendix C).
- AQBE can handle complicated queries without relinquishing its simplicity because it is based on notions of QBE.
- The user does not require making complex operations such as join or nesting.

Further, the question raised how to present the contents of selected concept (archetype) that form the result. For the sake of simplicity, a relational form of output has been adopted.

7. Experimental Results

7.1 Snapshots of Proposed Approach

The snapshots are shown for query scenario 1. The system presents an EHR (Fig. 16). In this case, the user selects Blood Pressure.

In the next step, an interface component (for BP) is presented to the user. The blood pressure consists of data, state and protocol information as shown in Fig. 17.

Based on choice of items above, the user is presented with the choice (Fig. 18) for specifying the conditions on the various items.

The system automatically generates AQL as shown in Fig. 18. Thus preventing the user to learn the query language syntax (AQL). The snapshots show the simple user query interfaces being presented to user.

7.2 Experimental Evaluation

A sample set of various queries has been used for checking the performance profile of AQBE in comparison to other query languages. Some of the queries have been presented...
The various query languages has been tested against the functionality, which includes project, select, rename, existential, nested, negation and join operations. All query languages can provide these basic functionality as depicted in Table 1. AQL is still under development, and grouping, difference, cartesian product and universal quantification have not been implemented yet. Similarly, the XQBE lacks for disjunction, union, leaf as a range of data, universal quantification, grouping, TOP operator. Also there is partial support for difference and nesting. The AQBE provides almost basic functionality required by healthcare workers.

Table 2 gives the qualitative evaluation of AQBE with respect to healthcare users. The query formulation effort and the language power are the evaluation parameters. The query (expression) formulation effort further includes thinking, input, probability of error and training. Input refers to the amount of clerical effort required to express the request. When the interaction is via a keyboard, this may be measured by the number of keystrokes. When pointing devices are used, a good measure of input is the number of pointed objects. The language power (query capability) is how much a user can do with a language. It considers application dependency, database dependency, functionality and selectivity. Table 2 shows that the query formulation effort required for AQBE is low and the functionality provided is sufficient.
according to the EHR user's need.

8. Conclusions

For the proposed EHR system, improvements in methods to access complex data have become inevitable. The new standard model proposed by openEHR foundation requires that the EHR will be accessed by skilled and semi-skilled users. This requires better interfaces by hospital users and researchers to access data. A query language to meet the new requirements has been proposed in this paper. An attempt has been made to incorporate modifications for a QBE type interface. The AQL is considered to be quite complicated for end-users to use, thus a visual-based alternative in the form of AQBE. A user-centric QL aims to improve quality by enhancing learnability, user satisfaction, and effectiveness of the EHR system.

The present study led to proposing AQBE (analogous to using QBE over relational data). It was chosen to use QBE over archetyped data because the archetypes are language neutral, dynamic, complete, accurate and overall play a vital role in semantic interoperability. The two-level modelling can be extended for any domain e.g., biomedical (genomics research). Thus, AQBE approach can be used with any system with archetypes.

References

[1] T. Beale, S. Heard, D. Kaiira, and D. Llyod, “The openEHR Reference Model: EHR Information Model,” The openEHR release 1.0.2., openEHR Foundation, 2008.
[2] ISO 13606-1., “Health informatics - Electronic health record communication - Part 1: RM.” 1st ed. 2008.
[3] T. Beale, The openEHR Archetype Model-Archetype Object Model., The openEHR release 1.0.2., openEHR Foundation, 2008.
[4] Archetype Query Language. http://www.openehr.org/wiki/display/spec/Archetype+Query+Language+Description (Accessed 2 October, 2011).
[5] M. Chunlan, F. Heath, B. Thomas, and H. Sam, “EHR query language (EQL)-A query language for archetype-based health records MEDINFO,” 2007.
[6] Microsoft Health framework. http://www.microsoft.com/industry/healthcare/technology/Health-Framework.mspx.
[7] D. Braga, A. Campi, and S. Ceri, “XQBE (XQuery by example): A visual interface to the standard XML query language,” ACM Trans. Database Systems, vol.30, no.2, pp.398–443, 2005.
[8] T. Beale, S. Heard, The openEHR Architecture: Architecture Overview, The openEHR release 1.0.2., openEHR Foundation, 2008.
[9] The openEHR foundation:.www.openehr.org.
[10] A. Silberschatz, H. Korth, and S. Sudershan, Database System Concepts, McGraw-Hill Book Company, Chapters on “Introduction, Data Models and Relational databases,” 5th ed. 2005.
[11] Clinical Knowledge Manager. http://www.openehr.org/knowledge/
[12] ADL for archetypes. http://www.openehr.org/svn/knowledge/archetypes/dev/html/index_en.html (Accessed 04/06/2010).
[13] Class Diagrams of RM classes. http://www.openehr.org/uml/release-1.0.1/Printable/Printable101.html#Diagrams (Accessed 08/05/2010).

[14] Moshe M. Zloof., Query-By-Example AIPS75: Proc. National Computer Conference and Exposition, 1975.
[15] S. Bhalla and M. Hasegawa, “Query Interface for Ubiquitous Access to Database Resources,” International Conference on Management of Data (COMAD), 2006.
[16] S.A. Rahman, S. Bhalla, and T. Hashimoto, “Query-by-object interface for information requirement elicitation in M-commerce,” Int. J. Human Computer Interaction (HCI), vol.20, no.2, pp.135–160, 2006.
[17] Query builder. Ocean Informatics. http://www.oceaninformatics.com/Solutions/ocean-products/Clinical-Modelling/Ocean-Query-Builder.html.
[18] S. Sachdeva and S. Bhalla, “Semantic Interoperability in Healthcare Informatics for EHR databases,” DNIS workshop, UoA, Japan, 2010.
[19] H.V. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi, and C. Yu, 2007. Making Database Systems Usable. SIGMOD, 2007.
[20] S. Sachdeva and S. Bhalla, “Implementing high-level query language interfaces for archetype-based electronic health records database,” International Conference on Management of Data (COMAD), 2009.
[21] T. Beale and S. Heard, “The openEHR Archetype Model-Archetype Definition Language ADL 1.4,” The openEHR release 1.0.2, (issue date 12 Dec 2008).
[22] European committee for Standardization, Technical committee on Health informatics, Standard for EHR communication. www.cen.eu.
[23] The HL7 organization. www.hl7.org.
[24] A. Sengupta and A. Dillon, “Query by templates: A generalized approach for visual query formulation for text dominated databases,” ADL, 1997.
[25] http://tomcat.apache.org/
[26] http://www.eclipse.org/
[27] http://www.openehr.org/projects/java.html
[28] http://www.seasar.org/en/
[29] T. Schuler, S. Garde, S. Heard, and T. Beale, “Towards automatic generation of GUIs from archetypes,” Studies in Health Technology and Informatics, vol.124, pp.221–226, 2006.
[30] L.M. Thurston, “Flexible and Extensible Display of Archetyped Data: The openEHR Presentation Challenge,” HIC 2006 Bridging the Digital Divide: Clinician, consumer and computer, 2006.
[31] Y. Li, C. Yu, and H.V. Jagadish, “Schema-free XQuery,” VLDB, 2004.
[32] HL7 Reference Information Model. [Online]. Available: http://www.hl7.org/v3ballot/html/infrastructure/rim/rim.htm.
[33] C. Yu and H.V. Jagadish, “Schema summarization,” VLDB, 2006.
[34] SNOMED (Systematized Nomenclature of Medicine) Clinical Terms. [Online]. Available: http://www.ihtsdo.org/snomed-ct/

Appendix A: Similarity of DBMS Architecture and EHR Architecture

Fig. A.1 Similarity of DBMS architecture and EHR architecture.
Appendix B: Sample of Clinical Document

Fig. A.2 Sample of clinical document.

Appendix C: Complex Structure of BP (tree form)

Fig. A.3 Complex structure of BP (tree form).

Shelly Sachdeva received the B.Tech. (Hons) in Computer Engineering from Maharishi Dayanand University, Rohtak, India and M.Tech. (Hons) degree in Computer Engineering from Guru Jambheshwar University, Hisar, India in 2001 and 2004, respectively. She has teaching experience of 8 years. She is currently pursuing PhD in Computer Science from University of Aizu, Japan.

Daigo Yaginuma is currently pursuing Masters in Computer and Information Systems from University of Aizu, Japan.

Wanming Chu received her B.E in computer science from Tsinghua University, Beijing and M.S. in software engineering at Beijing Institute of Software Training. She then developed database applications for six years and the system and application software for OMRs (Optical Mark Readers) for three years. Her research interest fields include Web query, Web Interface for Geographic Information System, General Web-based Database Management, Algorithm for Interconnection Network and Parallel Multithreaded Processor Performance Evaluation, and Simulation.

Subhash Bhalla received the B.Tech in Computer Science in 1978. He received the PhD degree in Computer Science in 1984 from Indian Institute of Technology, Delhi, India. His research interests include performance characterization of real-time dependable databases and the design of new databases to support multimedia information systems, data modeling and transactions and distributed algorithms. He is currently working as professor at University of Aizu, Japan. He has 24 years of experience as faculty, scientist and researcher.