Journal of Information Processing Vol.20 No.2 438–450 (Apr. 2012)

[DOI: 10.2197/ipsjjip.20.438]

Regular Paper

Visual Query Language for Archetype-Based Electronic Health Records Databases

Shelly Sachdeva[1,a)] Subhash Bhalla[1]

Received: July 2, 2011, Accepted: December 16, 2011

Abstract: The healthcare professionals have critical needs for general purpose query capabilities. These users increasingly require the use of information technology. The query needs cannot be met by form based user interfaces (or through the aids such as, the query builder). Further, the archetype-based Electronic Health Records (EHRs) databases are more complex, as compared with the traditional database systems. The present study examines a new way to support general purpose user level query language interface for querying EHR data. It presents the user with user's view of clinical concepts, without requiring any intricate knowledge of an object or stored structures. It enables clinicians and researchers to pose general purpose queries, over archetype-based Electronic Health Record systems.

Keywords: archetype query language, electronic health records, healthcare, high-level query interface, query languages, user interaction.

1. Introduction

The rise of the web has also led to the emergence of domain-specific information systems. These applications are often utilized by a large number of users. Furthermore, the large volume of data generates extensive querying needs. Thus, it has become necessary to focus on providing an ability to interact with data resources within EHR databases. In order to query the information, the user interface should provide suitable query language abilities.

1.1 New High-level Query Language

A query language (QL) is defined as a high-level computer language for the retrieval and the modification of data held in databases or files. It is usually interactive, on line, and able to support ad hoc queries. Query Languages have evolved in the context of database systems (repositories). Query-by-Example (QBE) [7] is a high-level QL interface for tabular data in relational databases. In contrast, the web contains the data in the form of documents (tree with hyperlinks). It is possible to download the documents from the web (in the form of XML) and use a QL interface for that. It can provide a user-friendly interface. Recent research in this field has led to the development of XQuery-By-Example (XQBE) [6]. It is a user-friendly, visual QL for expressing a large subset of XQuery for data in the XML form. Similarly for web data, Information Requirement Elicitation (IRE) has become essential to elicit information requirements through interactive choice prompts [16]. Currently, there is no usable QL interface, at the level of healthcare workers. This is a key requirement for information technology in the healthcare domain [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44]. Our study considers a database query interface as a QL for EHR databases. EHRs are complex and archetype-based (in contrast to table-based databases).

The study considers QBE as a model which can support many levels of user skills and many types of functional requirements. It can provide an interface that accepts the user's intent and communicates well-formed formulas (W.F.Fs) [13] for computations. The rest of the paper is organized as follows. The data modelling of EHRs and the role of archetypes are emphasized in Section 2. Section 3 presents querying archetype enabled EHR systems. Section 4 presents the visual QL interface for EHRs. It presents the Query-by-Concept approach, an example and implementation details. The advantages of the proposed high level QL are emphasized in Section 5. Section 6 describes the performance profile, the usability and the evaluation of the proposed approach. Section 7 gives related discussions and studies. Finally, Section 8 provides the summary and conclusions.

2. Model of EHR Systems

2.1 Data Model

Electronic Health Records (EHRs) have a complex structure that may include data from about 100–200 parameters, such as the temperature, the blood-pressure and the body mass index. Individual parameters will have their own contents. Individual parameters (concepts) are represented as archetypes. For example, each contains an item, such as 'data' (e.g., captured for a blood pressure observation). It offers the complete knowledge about a clinical context, (i.e., attributes of data), the 'state' (context for interpretation of data), and the 'protocol' (information regarding the gathering of data), as shown in **Fig. 1** (depicting complete-

[1] Graduate Department of Computer and Information Systems, University of Aizu, Aizu-wakamatsu, Fukushima 965–8580, Japan
[a)] sachdevashelly1@gmail.com

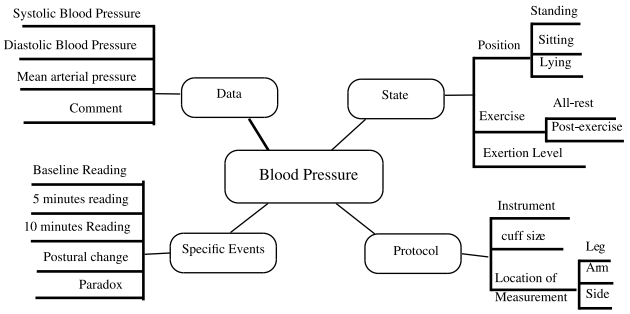

Fig. 1 Parameter blood pressure as a concept in the form of an archetype.

ness). Thus, the data modeling in healthcare is a time-consuming process. Most of the time is spent in learning the information system technologies.

2.2 Archetypes in Two-level Modelling

The medical concepts within EHRs (parameters) are represented in the form of an archetype. A two-level model is used to represent a concept and its storage. Under the two-level database paradigm, the core part of the system is based on the reference model and the archetype model (includes the generic logic for the storage, the querying and the caching). Both of the models are extremely stable, while domain semantics are mostly delegated to domain specialists who work building archetypes (reusable), templates (for local use) and the terminology (for general use) [33]. The two-level modelling approach, proposed initially by openEHR, includes the reference modeling (information modeling) and the content modeling. The Reference Model (RM) contains a stable and fundamental structure. It represents the generic structures of components of health record information, at the storage level. Content models, on the other hand, cover the informational aspects which are not so stable due to the variability and the high rate of change of the domain knowledge (i.e., the formal description of the physical examination or prescription). In simpler terms, in a two-level modelling, the RM provides a general framework for all EHRs and the archetypes define the rules for using EHR in different contexts. Consider an example. The first layer consisting of the reference model contains a generic class ACTOR. In the second layer, through the use of content models (archetypes), this generic class can be extended to be Doctor, Nurse or Insurer. An archetype is an agreed formal and interoperable specification of a re-usable clinical data set which underpins an electronic health record (EHR), capturing as much information about a particular and discrete clinical concept as possible. Thus, an archetype is "the prototype for the capture of clinical concepts - a machine readable specification of how to store patient data using the Reference Model."

Examples of a dual-model EHR architecture are CEN/TC251 EN13606 [9] (developed by the European committee for standardization) and the openEHR [10] (developed by the openEHR foundation). Recently, Microsoft has also adopted this approach. To incorporate this, Health Level 7 (HL7) [11] and Digital Imaging and Communications in Medicine (DICOM) are working on templating methodology, where templates are conceptually similar to archetypes.

An archetype contains rules for data entry into the system. For example, "the maximum and the minimum value," "allowed units," or whether a piece of data is required (or optional). Clinical concepts such as, the blood pressure, the health encounter, the diagnosis, the laboratory result are described by different archetypes. Thus, the archetypes specify the design of the clinical data that a health care professional needs to store. They are general-purpose, reusable and composable. These are clinically meaningful and interpretable by EHR systems, as they provide the structure and specify the content. These provide the ability of systems to reliably communicate with each other at the level of knowledge concepts. Thus, a level of abstraction has been added by storing the domain knowledge in the form of archetypes. Similarly, the physical data independence is achieved. As a result, with the expansion of the domain knowledge, the software does not need to be changed.

Only the first level i.e., the Reference Model (RM) [19], [20] is implemented in software. This part has the limitations imposed by application software and database schemas. RM specifies EHR_Extract as the root object which contains the EHR data. The other parts of the model universe (implemented in software) are in highly stable languages/models of representation (such as, programming languages, UML, XML Schema languages, OWL). With the use of two-level modelling, runtime data now conform semantically to archetypes as well as concretely to the reference model.

2.3 Research Motivation and Proposal

The present proposal proposes a visual query interface for EHRs. The archetypes model the character of clinical data attributes, and store this information, as it expresses data in the database (rather than in the database schema). The archetype instance is a maximally normalized object [33].

In a traditional setting, users express queries against the database schema. However the semantics of data can be understood by viewing the data in the context of the user interface (UI) (of the software tool used to enter the data). Archetypes have been used for the purpose of data entry and validation [33]. The study proposes a query interface from the conceptual model of archetype. The term 'concept' has one-to-one mapping with archetype and is a view of a user's object.

There are 279 archetypes developed up to now by openEHR and many more are expected in the near future [47]. The archetypes belong to different levels/categories according to the hierarchical levels [33]. The different categories have different structures. For example, 'COMPOSITION' is at the top-level of the hierarchy and is equivalent to a clinical document.

3. Querying EHR System

The existing QLs such as, SQL and XQuery are complicated for hospital users. For example, XQuery requires the extensive knowledge of a document structure (its XML schema) in order to formulate a query. The key challenges in querying EHRs are:

(1) Complex and domain specific semantics,
(2) Frequent references to external information sources such as dictionaries and ontologies, and
(3) Special treatment of time and location attributes.

```
                                  Identified path -systolic                    Naming retrieved results
SELECT obs/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value/magnitude AS Systolic,
       obs/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value/magnitude AS Diastolic
FROM EHR [ehr_id/value=$ehrUid]  Class Expression        Archetype Predicate
CONTAINS COMPOSITION [openEHR-EHR-COMPOSITION .encounter.v1]
CONTAINS OBSERVATION obs openEHR-EHR-OBSERVATION.blood_pressure.v1]
WHERE obs/data[at0001]/events[at0006]/data[at0003]/items[at0004]/value/magnitude>= 140
      OR
      obs/data[at0001]/events[at0006]/data[at0003]/items[at0005]/value/magnitude>=90
```

Fig. 2 Syntax of AQL.

(4) Archetypes need to be used as the basis for querying.
(5) Large slabs of data.

3.1 Archetype Query Language (AQL)

The standardized EHR system (openEHR) supports a standard domain-specific QL for archetype-based data [4], [5]. The syntax of AQL is illustrated by the help of an 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 the query. It makes use of the path expression, naming retrieved results, the class expression and the archetype predicate, as shown in the example in **Fig. 2**. The class expression syntax consists of openEHR RM class name (mandatory), followed by a variable name (optional) and 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. The '$' in Fig. 2 expresses the fact that the value of ehrUid will be substituted at run time.

Example: Find all the Blood Pressure (BP) values for a patient, having the systolic_*BP* and diastolic_*BP*, (where systolic_*BP* $\geq$ 140 or diastolic_*BP* $\geq$ 90).

3.2 Limitations of AQL

In a significant contrast to SQL and XQuery, AQL for EHRs is more complex due to the archetype-based structure. In this study, we adopt the notion of concept (archetype) from EHR, and propose a Graphical User Language (GUL) interface. The proposed GUL aims to emulate the QBE-style input forms for accepting user's intent. The system facilitates the formation of non-ambiguous expressions (W.F.F- Well Formed Formula) [13].

4. Visual Query Interfaces

The standard based EHR system has been implemented for hospitals (emergency department of Austin Health in Australia, and maternity care in a hospital in Cambodia) [26]. The opereffa project [40] is the real model of a practical EHR use (which is archetype-based). It uses PostgreSQL (object-relational database system). The programming interface may use XQuery/SQL. The Ocean informatics [43] makes use of XML-enabled relational databases. These use XQuery/SQL. Recently, AQL embedded in XQuery has been proposed by LiU-EEE [41] for querying archetype-based EHR databases. The openEHR provides AQL for querying EHR databases [4]. However, none of the systems discussed above provide high-level QL interface for archetype-based data.

The proposed graphic user language (GUL) interface (**Fig. 3**) is based on a top-down approach. In a separate study, we have proposed the detailed design specifications of a high-level query interface [46]. However, this approach requires the knowledge of the archetypes (concepts) present in the clinical document (EHR instance) on the part of users. The current approach has been evolved based on the above grounds. Here, the system internal mappings produce a list of concepts which are linked together in the clinical document.

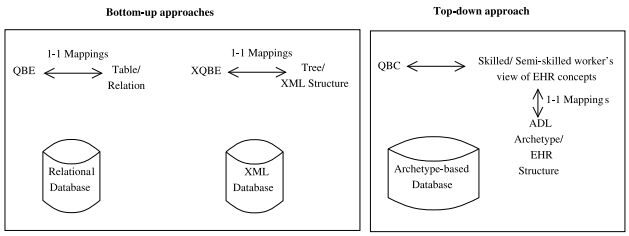

Fig. 3 Different visual query interfaces along with QBC for archetype-based data.

The EHR document consists of the details from the RM and ADL. **Figure 4** shows that RM (left, fragment of the openEHR RM in UML format) and the archetype (centre, in ADL format) are the blueprint for an EHR document (right, in XML format). The EHR structure consists of many clinical concepts arranged in a hierarchical fashion. It has a complex schema [33]. Each clinical concept has its own detailed structure consisting of many attributes (example in Fig. 1). The archetypes are defined using the Archetype Definition Language (ADL) [27] (approved by ISO [12]). ADL is composed of four main parts: the header, the definition, the ontology and the revision history. A sample and details of BP.adl is shown in Fig. 4. Thus, modeling along with detailed knowledge of schema and ADL, hinder the medical user from querying. In this light, the aim of the proposed research activity is:

1. To provide an independent querying capability to clinicians; and
2. To design a visual query interface for EHR users, who are not skilled in the use of AQL.

4.1 QBC Approach for EHRs

To explain the QBC approach we consider the following:

(1) User (skilled/semi-skilled hospital worker).
He/she is aware of the medical concepts which constitute the objects of query in the EHR repository.

(2) EHR Repository Contents: The repository contains EHRs for individual patients. Each EHR consists of archetypes at the composition level and the contents level. Thus, the target is a repository of clinical documents (a sample of a clinical document is shown in Fig. 4). It consists of multiple

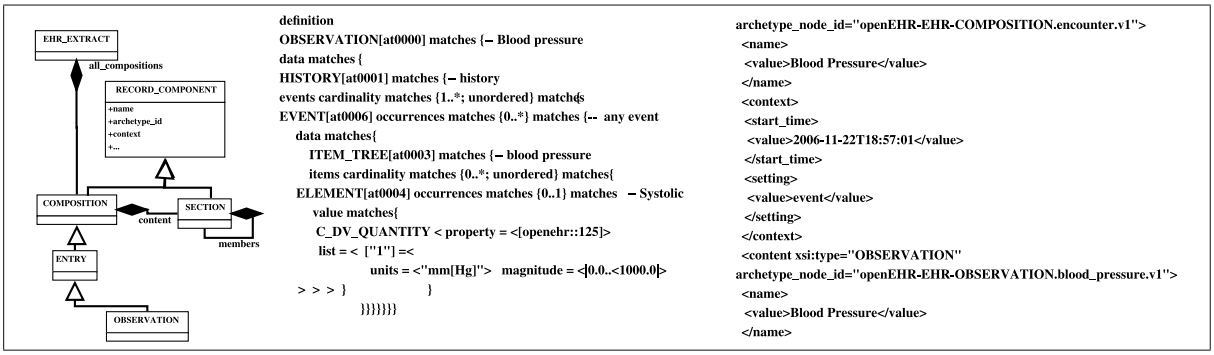

Fig. 4 The RM (left) and the archetype (centre) are the blueprint for an EHR document (right).

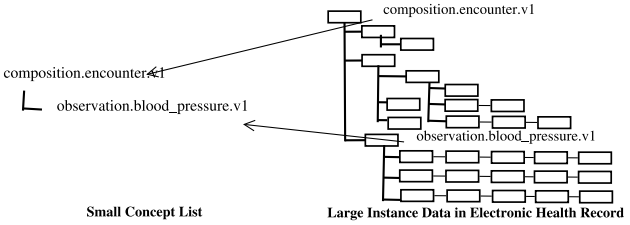

Fig. 5 QBC Map (small) compared to EHR data (large).

archetypes containing multiple concepts such as the blood pressure, the temperature and the heart rate. These have a 'containment' relationship among them because of the hierarchical nature of the EHR. Different archetypes are at different hierarchical levels, belonging to different categories.

(3) Query-by-Concept (QBC) System:
An individual EHR is organized as a collection of medical concepts, represented in the form of archetypes. The QBC system presents the user with a user's view as an interface to interact with the EHR repository.
Step1. Select the medical concept from the menu.
The structure is built from the available archetype repository. The view is presented by the system to ease the querying. The user selects the desired concept (user's object) (**Fig. 6** (a)).
Step2. QBC presents a tabular form of concepts and related attributes using the EHR repository and stored concepts (archetypes) and their links. The user indicates through micro steps, his/her choice of options, with respect to the query.
The EHR structure is built from the UML description of the EHR [39].

QBC overcomes the challenges of a complex schema and ADL structure as follows.

i) A clinical document contains the information about the concepts and the archetype-based data. The user's choice is mapped to a list of required archetypes based on the EHR document 'containment' relationships. The QBC system generates internal mappings of the concepts and clinical documents. These mappings produce a list of concepts which are linked together in the clinical document. For example, the concept list for an EHR document in Fig. 4 is shown in **Fig. 5**. This concept list is relatively small as compared to the large instance data in an EHR. Thus, QBC generates the smaller number of concepts which are involved in querying large slabs of data (Fig. 6 (b)).

ii) Also, each archetype (clinical concept), has a complex structure of its own contents. The data available in ADL [27] is used to create an individual QBE like interfaces (Fig. 6 (c)). In this view, the user can query using a selected level of granularity and specify the conditions. Consider the view for the concept 'blood_pressure' in Fig. 6 (c).

Thus, the user is presented with a high-level view independent of an underlying EHR database. In QBE, the conventional user selects the right tables and fills the predicates. In this QBC approach, the user works with the high-level view of the user's object (concepts). These objects have internal attribute mapping functions to the actual data stored in an EHR database.

4.2 QBC Example

The proposed approach is implemented on the basic SQL style data operations for queries (relational algebraic operations/set-theoretic operations [13]). The proposed QBC steps generate a well formed formula (with no ambiguity). It can further generate the AQL expression. Please see Appendix A.1 for QBC expressions for queries 1–14 for all data operations. We explain the QBC through the following query example.

Query Scenario. Find all the blood pressure values where the systolic value is greater than or equal to 140 or the diastolic value is greater than or equal to 90 within a specified EHR. (Sample Query in Ref. [4])

QBC interface (Fig. 6):

Step 1. The concept 'blood_pressure' is known to the health worker. It is selected in this step for a specified patient (Fig. 6 (a)).

Step 2. The required archetypes connected with 'blood_pressure', that is, 'encounter' and 'blood_pressure' are prompted to the user. QBC internal mappings generate this and present it to the user (Fig. 6 (b)).

For this query, the user indicates his choice of option as 'specify condition' (Restrict) and 'display' (Project), i.e.,
[Restrict] and [Project] single patient data

The user interface based on the archetype description of 'blood_pressure' is presented by the system as shown in Fig. 6 (c). The user can click the graphical widgets (checkboxes) to display the data items required in the result.

The system supports user through graphical widgets for specifying 'restrict' operation (systolic $\geq$ 140 OR diastolic $\geq$ 90) as

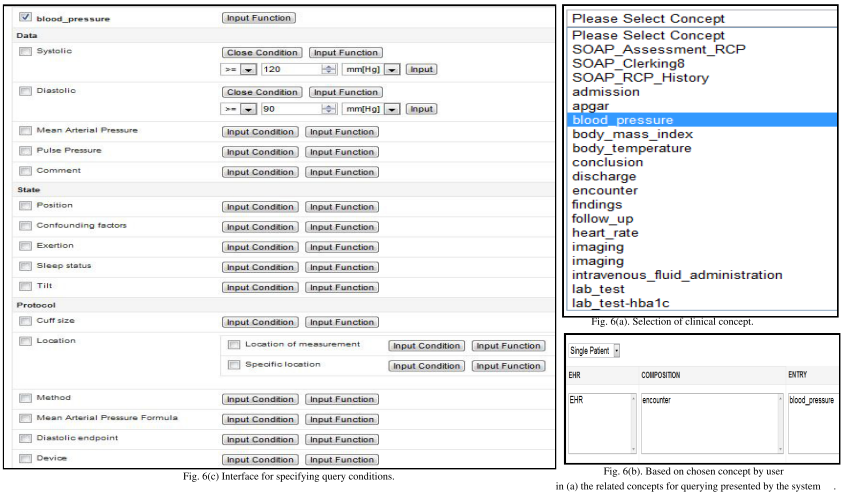

Fig. 6(c) Interface for specifying query conditions.

Fig. 6(a). Selection of clinical concept.

Fig. 6(b). Based on chosen concept by user in (a) the related concepts for querying presented by the system .

Fig. 6 QBC interface for Query Scenario presented in Section 4.2.

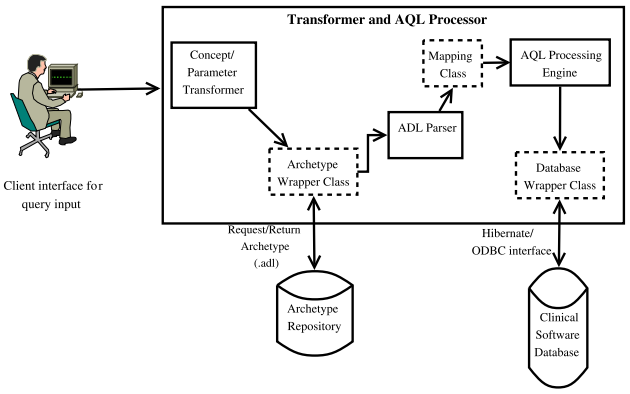

Fig. 7 QBC implementation scheme.

shown in Fig. 6 (c).

The example explains the top-down approach, where the user is presented with the conceptual view of objects (concepts) despite of the physical view (database objects), hiding the complex schema and ADL details.

4.3 Implementation Details

The proposed system has been implemented as shown in **Fig. 7**. It shows the view of an end-to-end request processing. A user's request is mapped on to the stored concepts through concept/parameter transformer. The archetype wrapper class acts as a communicating class. The required concept is requested and retrieved from the archetype repository. It is parsed and validated by the ADL Parser component. Upon a successful validation; the request is forwarded to the AQL Processing engine for the query execution through the mapping communication class. The application layer of the openEHR database is presented through hibernate (object relational mapping) interfaces. This interface communication is handled by the AQL Processing Engine at the time of an actual AQL execution. Therefore, the transformer and AQL processor acts as a mediator between the skilled/semi-skilled client and the openEHR database. This approach eliminated the low level AQL complexities from the end user view. A prototype system with query support for the high-level interface for the semi-skilled users has been utilized for tests and usability studies.

4.3.1 QBC Prototype System

The QBC prototype has been developed as a client-server application. The implementation uses Scala 2.9 [22] and Lift 2.4 [28]. The server part runs on an Apache Tomcat application server [3]. To get the clinical knowledge (information) from an archetype, the 'ADL parser', implemented in the openEHR Java Reference Implementation Project is being used [17].

Experiments have been conducted for a sample of queries for various (algebraic) operations such as select, project, join, rename, intersect and negation. QBE is a relationally complete language [7]. Similarly, the study infers that the QBC is a relationally complete language. Thus, the skilled/semi-skilled user's perception has been used for accessing and querying archetype-based EHR data. The feasibility of QBC as a complete approach has been enhanced by work in Refs. [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45].

4.3.2 Technical Challenges and Solutions

The challenges during the prototype implementation deal with:

i) A multitude of representations.
ii) Different categories of archetypes having different structures.
iii) The openEHR paths (path to the RM class attribute and the archetype path).

The QBC system deals with plain text, coded text, paragraphs, measured quantities with values and units, date, time, date-time, and partial date/time, encapsulated data (multimedia, parsable content), basic types (such as boolean, state variable), container types (list, set) and uniform resource identifiers (URI). It handles the major difficulty encountered in 'reference' object and 'media-type' object representations. The prototype system builds a different presentation model according to each category of archetype. The software has been coded for automatic generation of the openEHR path.

5. Advantages of Proposed Visual Query Interface

The QBC overcomes the following key challenges in querying

Table 1 Comparison of Query Languages and their functionalities.

Query Languages	SQL	AQL	Proposed Approach (QBC)
Project	✓	✓	✓
Restrict	✓	✓	✓
Rename	✓	✓	✓
Existential	✓	✓	✓
Nested	✓	✓	✓
Negation	✓	✓	✓
Join	✓	✓	✓
Relational operators/Boolean operators	✓	✓	✓
Renaming	✓	✓	✓
TOP operator	✓	✓	✓
Disjunction	✓	✓	✓
Union	✓	✓	✓
Sorting	✓	✓	✓
Difference	✓	✓	✓
Filtering	✓	✓	X
Parameterization Support	✓	✓	✓
Grouping	✓	Not yet	Not yet
Cartesian product	✓	Not yet	Not yet
Universal Qualification	✓	Not yet	Not yet
Path Expression	X	✓	✓
Update	✓	X	X
Querying Schema Order	✓	X	X
Querying Instance Order	✓	X	X

EHRs:

– Complex and domain specific semantics – Archetypes can support intelligent querying [33]. These contain domain specific knowledge. Thus, these ensure semantic interoperability.

– Frequent references to external information sources such as dictionaries and ontologies – The structure of an archetype contain a 'term binding' section for mapping the internal terms to the external terminologies (such as SNOMED [24]), and

– Special treatment of time and location attributes – The proposed GUL has been experimented for the openEHR standard, where the internal structure of archetypes and the standardized information model based on the clinical investigator recording process [33] take care of these attributes.

– Large slabs of data – Relatively few archetypes are used to construct quite large slabs of data. For example,consider 'ECG results', where one archetype corresponds to 10 leads' worth of time-series data, potentially hundreds of samples.

6. Performance Profile

QBC has been proposed on the basis of two criteria – functionality and user-friendliness. Functional capabilities refer to what one can do, whereas usability refers to the effort required.

6.1 Performance Considerations

A sample set of queries has been used for preparing the performance profile of various query languages. These have been considered against the conventional functionality requirements, which include project, select, rename, existential, nested, negation and join operations. All developer level query languages, such as SQL and AQL can provide these basic functionalities as depicted in **Table 1**. However, the visual languages tend to provide limited functionalities in comparison. AQL is under development, and few conventional query features, such as, grouping, cartesian product and universal quantification have not been implemented in AQL grammar. The QBC provides the basic functionality which is sufficient as per the requirement of healthcare workers. However, it does not provide all functionalities as compared with AQL (or SQL) (shown in Table 1).

6.2 Usability

The following steps were adopted for the estimation of the usability.

(i) Sample set of queries – For the proposed visual QL, the queries covering all basic algebraic operations have been considered for studying the usability. Any language as powerful as the relational algebra is called relationally complete [13]. Thus, if the proposed QBC is relationally complete, it can facilitate the user in formulating queries for all the common query needs.

(ii) Formulated steps – Identify objects/concepts, instances and then operator/condition.

(iii) Tested with participants – It may be noted that experiments often depend on students as participants (see Refs. [1], [30], [32]).

6.2.1 Participants

In all, 15 individuals participated in this study. The participants were randomly selected from the University of Aizu's student population of over 1,500. Eight undergraduate students and 7 postgraduate students participated in the study. On average, the students were about 24 years old. However, although all the students had some computing experience, none of them had any database QL experience. Thus, the participants would be representative of a common young generation of users who are computer literate but normally have little or no training in database query skills.

6.2.2 Apparatus

The aim of the experiments is to assess the usability as well as the intuitiveness of the QBC interface. It was decided to conduct

the experiments on a one-on-one basis. Each participant was observed by an experimenter while using the interface to formulate queries for the test questions. By observing the participants during the experiments, it became possible to assess the usability of the interface from close range. These experiments also helped to show if there were any difficulties faced by the participants (or to find any ambiguity that may arise). At the end of the experiment, the experimenter discussed the observations with the participant and asked for the participant's comments on the query interface.

For all experiments, the same Dell Studio 1558 Notebook computer was used by each participant. The participants used the QBC prototype system for archetype-based EHRs (Fig. 7). This usability estimation study is aimed at examining the effectiveness, the usability and the users general acceptance of the approach.

Each participant was given a question sheet containing the same set of 10 queries (query numbered 2, 3, 4, 5, 7, 8, 9, 11, 12, 14) (see Appendix A.1) based on the clinical query examples used in this article. An effort was made to ensure that the participants fully understood the queries.

6.2.3 Tasks

For the experiment, each participant used the QBC to formulate queries in response to a common set of questions based on the clinical query examples used in this article. All participants answered the questions in the same order.

After each successful query completion, the participant was asked for his or her confidence level (level of ease) in completing the task, by indicating the level of agreement to the first two questions from the ASQ [31].

6.2.4 Procedure

Prior to the experiment, participants were given a 15-min PowerPoint presentation by the experimenter to give a brief overview of the QBC interface with two query examples (1, 6). The QBC was further demonstrated for building queries based on three sample questions (the queries numbered 6, 10, 13).

After a short break, the participants had a 5-min practice session. Its aim was to acquaint the participants with the mechanics of the steps of the QBC and the interface so that during the test, they do not have to spend time to figure out how to use the GUL. The measured time for the test was then the real query formulation time. Only a minimal amount of time was allocated for the demonstration and no training on database QL was given to participants. The effort was made to test the hypothesis that the interface is intuitive and easy to use by novice users.

When a participant indicated his or her readiness, the experimenter asked the participant to formulate the query for the first test question and started the timer. The participant prepared the query steps. The experimenter observed the participant's actions on the screen and took note of any difficulty or problem. Once the query had been formulated correctly, the experimenter recorded the time-to-complete the task and the associated number of query attempts taken. In case of a failure to construct a query correctly, the Reset option was used to start over again.

Participants were not allowed to refer to any training notes or use paper and pencil to help formulate the steps. The participants were expected to reason, in terms of concepts and algebra operations, and sequences of algebra operations, in their minds to formulate query steps. This was done to resemble the way a user would ultimately use the interface in a real hospital scenario. The 'concept description' support (facilitated in the interface) removes the complexity of remembering the complex structure of a concept (archetype). Thus, providing the user the flexibility of querying fine granularity of data.

As mentioned under the Tasks section, after completing each query formulation task, participants indicated their level of ease/confidence to the first two questions from the ASQ.

6.2.5 Experimental Design

In line with the objectives, the following measures were recorded: (a) the number of attempts taken to formulate a query accurately, (b) the associated time taken, (c) the query accuracy score, and (d) the participant confidence in task completion.

The choice of these performance measures is based on the need to study ease-of-use and the effectiveness of the QBC approach, as well as users' acceptance of the approach in general. The first three performance measures would assess the ease-of-use and effectiveness factors, whereas the fourth would give an indication of the users' acceptance of the approach, that is, its efficiency and effectiveness from the users' perspective. In addition, the aim was to assess the effectiveness of the QBC interface for formulating correct queries. To get correct answers to their questions, the users had the need to formulate their queries correctly according to logical and specific steps (or any one of a set of alternative steps). Thus, in this study, the number of attempts and the time taken to build an accurate query are important performance measures rather than the accuracy of the query in essence. Furthermore, determining the number of attempts taken to build an accurate query was straightforward and involved no subjective judgment. It is also the case with calculating the query accuracy score (discussed later). The query performance of end-users is commonly assessed by three variables: the query accuracy, the time taken to formulate the queries, and the participants' confidence in their queries (see Refs. [1], [30], [32]). The confidence level is usually self-reported by the participant for each query.

In the case of QBC, the user satisfaction with the QBC interface is measured using the ASQ score. We used the first two items of the three-item ASQ proposed by Lewis [2], using the average response to these two items as the overall ASQ score for this study. The two items measure the user satisfaction in terms of the "ease of completion" and the "time to complete" each query formulation task. ASQ scores range from 1 to 7, with lower scores indicating a greater satisfaction. The level of users' skill was a constant in this experiment and was set at the novice level, for reasons stated earlier. The query complexity has three levels: simple, medium, and complex. A simple query is defined as a single patient query involving a single concept (query numbered 1 to 5), medium as a single/multiple patient (population) query involving one or two concepts (query numbered 6 to 11), and complex as a query involving two or more concepts for multiple patients or multiple patient query involving data from a single patient query (query numbered 12 to 14). A medium and complex query formulation would involve progressively building a query using microsteps. As the number of concepts increases, the query becomes more difficult. It was aimed to test whether there is any signif-

Table 2 Mean number of attempts taken to formulate accurate queries.

Query Complexity	Mean no. of Attempts Taken to Formulate Accurate Queries
Simple	1.20
Medium	1.07
Complex	1.15

Table 3 Means, standard deviations, and the 95% confidence limits for the time taken (in Sec) to formulate accurate queries.

Query Complexity	Mean	SD	95% Lower Limit	95% Upper Limit
Simple	21.8	11.08	18.50	25.17
Medium	34.3	9.57	31.35	36.52
Complex	50.2	15.45	49.35	54.82

icant difference between the accuracy scores (defined and determined as shown next) for simple, medium, and complex queries. This effort would also show whether users can cope with queries of increasing complexity using sequences of algebra operations without any difficulty. This would determine whether the QBC interface is indeed simple and intuitive for users. In this study, the accuracy score for each query is defined and determined as follows:

The query accuracy score is defined as the percentage of accurate queries formulated on the first attempt for a particular test question = (No. of participants who formulated the query accurately in the first attempt/Total no. of participants) $*$ 100%.

This accuracy score calculates, for each query, the percentage of participants who were able to formulate the query accurately on the first attempt. This score would assess the overall effectiveness of QBC for formulating that query. By computing the accuracy scores for simple, medium, and complex queries, we would be able to determine whether QBC can be used to formulate all types of query with equal ease.

6.3 Results

6.3.1 Number of Attempts

The means for the number of attempts taken by participants to formulate accurate queries of different levels of complexity are shown in **Table 2**. All the means for the number of attempts are marginally over 1. This indicates that, on average, participants were able to formulate accurate queries on their first attempts for each type of query. Overall, the maximum number of attempts recorded was 2. Thus, it can be concluded that QBC can be efficiently used to formulate simple, medium, and complex queries with equal ease. The second attempts were mainly due to the fact that participants were initially not familiar with all the algebra operations and the interface, and thus mistakes were (probably) higher during this learning phase. Some participants, in their efforts to achieve a record time in completing each query, inadvertently clicked on the wrong options in their haste. The other second attempts were mainly due to carelessness. These involved cases where the participants did not read the question carefully and had chosen the wrong concept, for example, laboratory instead of laboratory-hbA1c. However, most participants realized their mistakes after checking the tabular contents of the chosen concept presented in step 2.

Overall, participants were able to formulate queries accurately on the first attempt, and if not, by the second attempt. And it should be noted that most cases involving second attempts were not due to ambiguity in formulating queries or difficulty in using the interface.

6.3.2 Time Taken to Formulate Queries

The means and standard deviations for the times taken (in seconds) to formulate accurate queries are shown in **Table 3**. The time taken to formulate an accurate query is defined as the total time recorded by a participant to click through the correct sequence of query steps and options until successful query completion. As expected, the mean times taken to formulate accurate queries increases in tandem with the level of query complexity. However, all the means (as well as the 95% confidence interval for the population means) are within 1 min, which indicates that the interface can be efficiently used for formulating queries of differing levels of complexity.

As participants were given only a limited time for (5-min) practice session using QBC, some participants took an additional time to familiarize themselves with the GUI interface. These participants took longer times (than the other users) to formulate queries. They clicked directly on the displayed options (like selecting options on a desktop application) instead of the designated buttons (e.g., drop down keys, hot keys, shift and select keys) of the form when choosing options. This resulted in the variations (as above) in the time taken to formulate queries.

Generally, participants took the longest times to formulate complex queries. There are basically two reasons for this. First, the number of query steps is larger. Second, participants need to employ more reasoning in terms of database operations and sequences of database operations to formulate queries correctly. However, participants were able to resolve any difficulty by the facilitation of structure provided by the system for a chosen concept. The standard deviation for the times taken to formulate queries is also the highest for complex queries. The reason for this higher variation is explained. First, at one extreme, some participants were able to reason very quickly when performing queries sequences of algebra operations. At the other extreme, some other participants took more time to figure out the correct sequences of algebra operations.

6.3.3 Query Accuracy Score

The query accuracy score is defined as the percentage of accurate queries formulated on the first attempt for a test question. **Table 4** shows the means of the query accuracy score for the three types of queries. The mean for a simple query is 83%). For complex queries, the mean score dropped slightly to 85.6% as partici-

pants needed to use more reasoning and time to formulate queries correctly. Overall, the high mean query accuracy scores recorded prove that the QBC interface is simple to use even for formulating complex queries.

6.3.4 Subjective Ratings After-Scenario Questionnaire (ASQ) Scores

The mean ratings (and standard deviations) for each ASQ item and for the overall ASQ are shown in **Table 5**. Overall, the means for each ASQ item and the overall ASQ score indicate that participants were highly satisfied with the QBC interface for formulating queries of varying complexities. As expected, the participant satisfaction levels were highest for simple queries, as these queries are the easiest and fastest to complete (see Table 3). The satisfaction levels decreases marginally as the level of query complexity increases (higher ASQ score indicates lower satisfaction). Thus, it can be concluded that participants are highly satisfied with the QBC for formulating simple, medium, and complex queries, both in terms of ease of use and time for successful completion of task, that is, query formulation.

6.3.5 Gradual Learning Curve

It is little challenging for the semi-skilled users to perform multiple patient query involving data from a single patient query (e.g., query 14: Get all patients who are suffering from the same problem as a specific patient (e.g., Patient X)). A training support can be prepared for such queries, in the form of interactive examples (similar examples). It is expected that the users can acquire the skills.

6.4 Evaluation

Table 6 gives the evaluation of the proposed approach with respect to health 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 human level 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 the application dependency, the database dependency, the functionality and the selectivity. Table 6 shows that the query formulation effort required for QBC is low and the functionality provided is sufficient according to the EHR user's need.

Table 4 Mean for query accuracy score.

Query Complexity	Mean (in percentage)
Simple	83
Medium	92
Complex	85.6

Table 5 Means and standard deviations for ASQ scores.

Query Complexity	ASQ Item 1[a]		ASQ Item 2[b]		Overall ASQ	
	Mean	SD	Mean	SD	Mean	SD
Simple	1.88	1.19	1.83	1.03	1.86	1.11
Medium	2.08	1.03	2.14	1.03	2.11	0.95
Complex	2.45	1.22	2.39	1.30	2.42	1.26

[a]ASQ Item 1 measures satisfaction with "ease of completion" of the task.
[b]ASQ Item 2 measures satisfaction with "time to complete" the task.

7. Related Studies

Many efforts are made to create query language interfaces [44]. A query interface for searching EHRs temporal patterns is discussed in Ref. [42]. RetroGuide, enables non-experts to formulate query tasks using a step-based, patient-centered paradigm inspired by workflow technology [38]. It has been compared to SQL and proposes a research for further development of a novel query paradigm for EHR data. A recent research explored how to implement access control for PHRs through standard relational database queries [35]. vSPARQL helps to access relevant content by querying against view definitions in semantic web for biomedical ontologies [36]. For effective MEDLINE document retrieval, a query reformulation technique has been described [37].

MUMPS (Massachusetts General Hospital Utility Multi-Programming System) is a programming language created in the late 1960s, originally for use in the healthcare industry. It is currently used in electronic health record systems as well as by multiple banking networks and online trading/investment services [23]. Due to its programming nature, it cannot be used for making queries to EHRs. Further, a search engine can be implemented to be on the top of a database. The approach suffers from the lack of returning precise and accurate answers. Thus, all these provide an inadequate support to express a query. It has been found that AQL cannot support clinicians. AQL or a similar query language requires the use of domain knowledge (stored in the form of archetypes) [5]. Users find it difficult to write AQL syntax. An alternative is to use an AQL query builder [8]. It often limits the query expressiveness. QBE [7], Query By Template (QBT) [15] and Information Requirement Elicitation (IRE) [16] were investigated. Their study has led to the design of the QBC proposal.

The information model given by openEHR and HL7 has a very

Table 6 Evaluation of the proposed approach.

Query Languages	Expression Formulation Effort				Language: Query Capability		
	Thinking	Input	Probability of Error	Training requirement	Application dependent	Database dependent	Functionality and selectivity
Domain Specific QL (AQL)	High	Medium	Medium (Clerical, Syntactic)	High	Yes	No	High (Table 1)
High-Level QL Support (QBC)	Low	Low	Low	Low	Yes	Yes	Relationally Complete Sufficient for health users (Table 1)

complex schema for representing the health record [19], [20]. One solution proposed is schema summarization [21]. It represents the original complex schema with a smaller and conceptually simpler schema. It helps the users explore the schema, but still requires the knowledge of the schema by skilled/semi-skilled medical workers.

The challenge in the healthcare domain is the unknown schema. The solution can be schema-free XQuery [18], but the end-users are still uncomfortable in using it because of the requirement of a partial knowledge of the schema. XGI [34] is an XQuery graphical interface designed for inexperienced biomedical researchers to effectively query different XML data sources. It is unable to capture the full complexity and the variability of XQuery but still can fulfill most of their desired needs. It requires underlying schema knowledge. Similarly, Natural Language interfaces help users avoid the burden of learning logic-based language [29]. However, these are difficult to build. These also suffer from problems of the linguistic variability and ambiguities. A form based approach has limitations, such as, designing a new form whenever a new need arises. Moreover, a person with a programming background has to come each time to design a form [14].

8. Summary and Conclusions

The present study examines the need to support "general purpose" user-level QL interface for querying EHR databases. The GUL is easy to use and has been engineered for clinicians using concepts. It is based on a user-centric model. The framework can be easily extended for various domains. One such emerging domain is the biomedical research (e.g., data on genomics), where capturing the context of information has also been identified as an important requirement. Similar high level querying interfaces can be proposed by presenting the user with user's view of clinical concepts, without requiring any intricate knowledge of an object or stored procedures.

The bottom up approaches of providing high level interface such as, QBE and XQBE in case of SQL and XQuery respectively, are straightforward. However, it may take much more effort in case of AQL because AQL uses a mixture of Xpath and SQL. This study has presented the semi-skilled users with the top-down approach. The user is presented with a user's view of clinical concepts, without requiring any intricate knowledge of an object or stored structures. The proposed QL offers a higher level of usability for healthcare specialists who are well acquainted with the parameters and concepts. Thus, this approach provides medical persons a more active role in querying EHRs. Currently, all the prevalent EHR standards (HL7, openEHR, CEN 13606) are moving towards the archetype-based technology. Hence, the proposed QL will serve as a model.

Reference

[1] Chan, H.C., Wei, K.K. and Siau, K.L.: Use-database interface: The effect of abstraction levels on query performance, *MIS Quarterly*, Vol.17, No.4, pp.441–464 (1993).

[2] Lewis, J.R.: Computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use, *International Journal of Human-Computer Interaction*, Vol.7, No.1, pp.57–78 (1995).

[3] Apache Tomcat server 6, available from ⟨http://tomcat.apache.org/⟩.

[4] Archetype Query Language Description, available from ⟨http://www.openehr.org/wiki/display/spec/Archetype+Query+Language+Description⟩.

[5] Chunlan, M., Heath, F., Thomas, B. and Sam, H.: EHR Query Language (EQL)-A Query Language for Archetype-Based Health Records, *MEDINFO 2007*, pp.397–401 (2007).

[6] Braga, D., Campi, A. and Ceri, S.: XQBE (XQueryBy Example): A Visual Interface to the Standard XML Query Language, *ACM Trans. Database Syst.*, Vol.30, No.2, pp.398–443 (2005).

[7] Zloof, M.M.: *Query-By-Example*, *Proc. National Computer Conference and Exposition* (*AFIPS '75*), pp.431–438 (1975).

[8] Ocean Informatics: Query builder, available from ⟨http://www.oceaninformatics.com/Solutions/ocean-products/Clinical-Modelling/Ocean-Query-Builder.html⟩.

[9] European committee for Standardization, Technical committee on Health informatics, Standard for EHR communication, available from ⟨www.cen.eu⟩.

[10] openEHR Foundation, available from ⟨www.openehr.org⟩.

[11] The HL7 organisation, available from ⟨www.hl7.org⟩.

[12] The ISO organization, available from ⟨www.iso.org⟩.

[13] Silberschatz, A., Korth, H. and Sudershan, S.: Database System Concepts, *Chapters on 'Introduction, Data Models and Query Languages', 5th Edition*, ISBN 0-07-295886-3, McGraw-Hill Book Company (2005).

[14] Jagadish, H.V., Chapman, A., Elkiss, A., Jayapandian, M., Li, Y., Nandi, A. and Yu, C.: Making database systems usable, *Proc. 2007 ACM SIGMOD International Conference on Management of Data*, Beijing, China, pp.13–24 (2007).

[15] Sengupta, A. and Dillon, A.: Query by Templates: A Generalized Approach for Visual Query Formulation for Text Dominated Databases, *IEEE ADL*, pp.36–47 (1997).

[16] Sun, J.: Information Requirement Elicitation in M-Commerce - An Interactive Approach to Facilitate Information Search for Mobile Users, *Comm. ACM*, Vol.46, No.12, pp.45–47 (2003).

[17] openEHR Java Reference Implementation Project, available from ⟨http://www.openehr.org/projects/java.html⟩.

[18] Li, Y., Yu, C. and Jagadish, H.V.: Schema-Free XQuery, *VLDB*, pp.72–83 (2004).

[19] Beale, T., Heard, S., Kalra, D. and Llyod, D.: The openEHR Reference Model: EHR Information Model (2008), available from ⟨http://www.openehr.org/releases/1.0.2/architecture/rm/ehr_im.pdf⟩.

[20] HL7 Reference Information Model, available from ⟨http://www.hl7.org/v3ballot/html/infrastructure/rim/rim.htm⟩.

[21] Yu, C. and Jagadish, H.V.: Schema Summarization, *VLDB*, pp.319–330 (2006).

[22] Scala, available from ⟨http://www.scala-lang.org⟩.

[23] MUMPS, available from ⟨http://en.wikipedia.org/wiki/MUMPS⟩.

[24] SNOMED (Systematized Nomenclature of Medicine) Clinical Terms, available from ⟨http://www.ihtsdo.org/snomed-ct/⟩.

[25] Schuler, T., Garde, S., Heard, S. and Beale, T.: Towards automatic generation of GUIs from archetypes, *Studies in Health Technology and Informatics*, Vol.124, pp.221–226 (2006).

[26] GoK, M.: Introducing an openEHR-Based Electronic Health Record System in a Hospital, Master thesis, University of Goettingen (2008).

[27] Beale, T. and Heard, S.: The openEHR Archetype Model-Archetype Definition Language ADL 1.4, openEHR release 1.0.2, Issue date 12 Dec. (2008).

[28] Lift, available from ⟨http://www.liftweb.net/⟩.

[29] Marti, P., Profili, M., Raffaelli, P. and Toffoli, G.: Graphics, Hyperqueries, and Natural Language: An Integrated Aprroach to User-Computer Interfaces, *Proc. Int. Workshop on Advanced Visual Interfaces*, Vol.36, pp.68–84 (1992).

[30] Siau, K.L., Chan, H.C. and Wei, K.K.: Effects of query complexity and learning on novice user query performance with conceptual and logical database interfaces, *IEEE Trans. Systems, Man, and Cybernetics - Part A: Systems and Humans*, Vol.34, pp.276–281 (2004).

[31] Commarford, P.M.: An investigation of text throughput speeds associated with pocket PC input method editors, *International Journal of Human-Computer Interaction*, Vol.17, No.3, pp.293–308 (2004).

[32] Greenblatt, D. and Waxman, J.: *A study of three database query languages*, *Databases: Improving usability and representativeness*, Shneiderman, B. (Ed.), pp.76–87 (1978).

[33] Beale, T. and Heard, S.: openEHR Architecture, The openEHR foundation, release 1.0.2 (2008).

[34] Li, X., Gennari, J.H. and Brinkley, J.F.: XGI: A Graphical Interface for XQuery Creation, *AMIA 2007 Symposium Proceedings*, pp.453–457 (2007).

[35] Sujansky, W.V., Faus, S.A., Stone, E. and Brennan, P.F.: A method to implement fine-grained access control for personal health records

through standard relational database queries, *Journal of Biomedical Informatics*, Vol.43, No.5, pp.S46–S50 (2010).
[36] Shaw, M., Detwiler, L.T., Noy, N., Brinkley, J. and Suciu, D.: vSPARQL: A view definition language for the semantic web, *Journal of Biomedical Informatics*, Vol.44, No.1, pp.102–117 (2011).
[37] Yoo, S. and Choi, J.: On the query reformulation technique for effective MEDLINE document retrieval, *Journal of Biomedical Informatics*, Vol.43, No.5, pp.686–693 (2010).
[38] Huser, V., Narus, S.P. and Rocha, R.A.: Evaluation of a flowchart-based EHR query system: A case study of RetroGuide, *Journal of Biomedical Informatics*, Vol.43, No.1, pp.41–50 (2010).
[39] Class Diagrams of RM classes, available from ⟨http://www.openehr.org/uml/release-1.0.1/Printable/Printable101.html/#Diagrams⟩.
[40] Opereffa Project, available from ⟨http://opereffa.chime.ucl.ac.uk/introduction.jsf⟩.
[41] REST Based Services and Storage Interfaces for openEHR Implementations, available from ⟨http://www.imt.liu.se/ erisu/2010/EEE-Poster-multipage.pdf⟩.
[42] Plaisant, C. et al.: Searching Electronic Health Records for Temporal Patterns in Patient Histories: A Case Study with Microsoft Amalga, *Proc. AMIA Annu. Symp.*, pp.601–605 (2008).
[43] Ocean Informatics, available from ⟨www.oceaninformatics.com⟩.
[44] Catarci, T., Costabile, M.F., Levialdi, S. and Batini, C.: Visual query systems for databases: A survey, *Journal of Visual Languages and Computing*, Vol.8, No.2, pp.215–260 (1997).
[45] Thurston, L.M.: Flexible and Extensible Display of Archetyped Data: The openEHR Presentation Challenge, *Proc. HIC 2006 and HINZ 2006*, pp.28–36 (2006).
[46] Sachdeva, S., Yaginuma, D., Chu, W. and Bhalla, S.: AQBE–QBE Style Queries for Archetyped Data, *IEICE Trans. Inf. and Syst.*, Vol.E95-D, No.3, pp.1–11 (Mar. 2012) (to appear).
[47] Clinical Knowledge Manager, available from ⟨http://openehr.org/knowledge/⟩.

Appendix

A.1 QBC for Queries 1-14

Algebra Operations:

A.1.1 Project

Query 1. Get a patient's current medication list (Sample Query in Ref. [5], with the following AQL expression)

QBC interface:

Step 1. The concept 'medication_list' is known to the user. The user identifies 'medication_list' for the selected patient.

Step 2. [Project] single patient data

(archetype description of 'medication_list' presented by the system).

Query 2. Query on structure of EHR

Retrieve all compositions' name value, context start time and composer name from a specific EHR (Sample Query in Ref. [4]).

QBC interface:

Same steps as query 1, with 'composition' as concept.

Query 3. Return the value of laboratory-glucose for a specific patient.

QBC interface:

Step 1. The concept 'laboratory-glucose' is known and selected by the user for a specified patient.

Step 2. The required archetypes connected with 'laboratory-glucose', that is, 'encounter' and 'laboratory-glucose' are prompted to the user in the form of tabular data.

[Project] single patient data

(archetype description of 'laboratory-glucose' presented by the system).

(archetype description of 'encounter' presented by the system).

A.1.2 Restrict and Project

Query 4. Presented in section 'QBC example'.

Query 5. Get BMI values which are more than 30 kg/m2 for a specific patient.

QBC interface:

Step 1. The concept 'body_mass_index' is known and selected by the health worker for a specified patient.

Step 2. The required archetypes connected with 'body_mass_index', that is, 'report' and 'body_mass_index' are prompted in the form of tabular data.

[Restrict] and [Project] single patient data

(archetype description of 'body_mass_index' presented by the system).

The system supports facilities for specifying 'restrict' operation (body_mass_index > 30).

Query 6. Get all HbA1c observations that have been done in the last 12 months for a specific patient (sample query from Ref. [6])).

QBC interface:

Step 1. The concept 'lab_test-hba1c' is known and selected by the health worker for a specified patient.

Step 2. The required archetypes connected with 'lab_test-hba1c', that is, 'report' and 'lab_test-hba1c' are prompted to the user in the form of tabular data.

(archetype description of 'lab_test-hba1c' presented by the system).

(archetype description of 'report' presented by the system).

[Restrict] and [Project] single patient data

The system supports facilities for specifying 'restrict' operation (report - last 12 months).

A.1.3 Rename

Query 7. Find all blood pressure (BP) values for a specific patient, showing their systolic and diastolic blood pressure values; also change the tagname of systolic BP as 'Sys' and Diastolic BP as 'Dias'.

Same steps as query 4, with change of operation as 'rename'.

A.1.4 Existential Query

Query 8. Return all BP elements having a position in which the BP was recorded.

QBC interface:

Same steps as query 7, with the operation 'Exist' on the position attribute.

A.1.5 Negation

Query 9. Get the blood pressure values where the position is not standing.

QBC interface:

Same steps as query 7, with the operation 'Negate' on the position attribute.

Multiple Patient (Population Query)

A.1.6 Intersect

Query 10. Find all the patients who have diabetes but no record of hypertension diagnosis.

QBC interface:
Step 1. The concept 'problemdiagnosis' is selected by the health worker. The user selects multiple patient (population) query.
Step 2. The required archetypes connected with 'problemdiagnosis' that is, 'problem_list' and 'problemdiagnosis' are prompted to the user in the form of tabular data.
It is performed in the following iterative micro-steps:
(i) [Project] [Restrict] on 'problemdiagnosis'.
(archetype description of 'problemdiagnosis' presented by the system).
The system supports facilities for specifying 'restrict' operation (problemdiagnosis = diabetes).
(ii) [Project] [Restrict] on 'problemdiagnosis'.
(archetype description of 'problemdiagnosis' presented by the system).
The system supports facilities for specifying 'restrict' operation (problemdiagnosis != hypertension).
(iii) Result of (i) [Intersect] Result of (ii)
The system supports facilities for specifying 'intersect' operation.

A.1.7 Aggregate Operations

Query 11. Get the number of all the patients with diabetes.
QBC interface:
Step 1. The concept 'problemdiagnosis' is selected by the health worker. The user selects multiple patient (population) query.
Step 2. The required archetypes connected with 'problemdiagnosis', that is, 'problem_list' and 'problemdiagnosis' are prompted to the user in the form of tabular data.
[Restrict] [Aggregate] operation.
(archetype description of 'problemdiagnosis' presented by the system).
The system supports facilities for specifying COUNT (aggregate operation) and 'restrict' operation (problemdiagnosis = diabetes).

A.1.8 Complex Queries

Scenario 1 (*Population multiple patients Query*) *Restrict, Aggregate, Intersect*

Query 12. Get the number of all patients with diabetes who have HbA1c results greater than 7.0 in the last 12 months (Sample query from Ref. [6])).
QBC interface:
This query involves solving these 2 simple queries:
(i) Get all patients with diabetes.
(ii) Get all patients who have HbA1c observations > 7 in the last 12 months.
Perform 'intersect' on the results obtained from above queries.
For query (i),
Step 1. The concept 'problemdiagnosis' is known and selected by the health worker. The user selects multiple patient (population) query.
Step 2. The required archetypes connected with 'problemdiagnosis', that is, 'problem_list' and 'problemdiagnosis' are prompted to the user in the form of tabular data.
[Restrict] and [Project] multiple patient data.
(archetype description of 'problemdiagnosis' presented by the system).
The system supports facilities for specifying 'restrict' operation (problemdiagnosis = diabetes).
For query (ii),
Step 1. The concept 'lab_test-hba1c' is known and selected by the health worker. The user selects multiple patient (population) query.
Step 2. The required archetypes connected with 'lab_test-hba1c', that is, 'report' and 'lab_test-hba1c' are prompted to the user in the form of tabular data.
(archetype description of 'lab_test-hba1c' presented by the system).
(archetype description of 'report' presented by the system).
[Restrict] and [Project] multiple patient data.
The system supports facilities for specifying 'restrict' operation (lab_test-hba1c > 7, report - last 12 months).
Perform 'intersect' on results obtained from query (i) and query (ii) followed by [Aggregate]
The system supports facilities for specifying COUNT (aggregate operation).

Scenario 2 (*Population Query*) *Nested, Negation and Join*

Query 13. Retrieve all patients who have not been discharged.
QBC interface:
Step 1. The concepts 'admission' and 'discharge' are known to the health worker and the user selects them. The user selects multiple patient (population) query.
Step 2. [Nested] [Negate] and [Join] operation
(archetype description of 'admission' and 'discharge' presented by the system).
It is performed in the following iterative micro-steps:
(i) [Project] for multiple patients from 'admission'.
(ii) [Project] for multiple patients from 'discharge'.
(iii) [Join] (i) and (ii) with [Negate] on (ii) (i.e., admission (encounter_id) 'not in' discharge (encounter_id)).

Scenario 3. Multiple Patient query involving data from Single Patient query

Query 14: Get all patients who are suffering from the same problem as a specific patient (e.g., Patient X).
QBC interface:
(i) [Project] single patient query on problem diagnosis for X.
(ii) [Restrict] population query on problem diagnosis.
(problem diagnosis = result of step (i)).

Shelly Sachdeva received her B.E. (Hons) and M.Tech. (Hons) degrees in Computer Science, in India in 2001 and 2004, respectively. She has teaching experience of 8 years. She is currently pursuing Ph.D. from University of Aizu, Japan. Her main research interests are in the area of electronic health record databases, the high-level query interfaces, and data quality for health informatics.

Subhash Bhalla received his B.Tech in Computer Science in 1978. He received his Ph.D. degree in Computer Science in 1984 from Indian Institute of Technology, Delhi, India. His research interests include 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.