Database and Knowledge Base Integration in Decision Support Systems

Johansson B, Shahsavar N, Åhlfeldt H, Wigertz O
Department of Medical Informatics, Linköping University, S-581 85 Linköping, Sweden

Since decision support systems (DSS) in medicine often are linked to clinical databases it is important to find methods that facilitate the work for DSS developers to implement database queries in the knowledge base (KB). This paper presents a method for linking clinical databases to a KB with Arden Syntax modules. The method is based on a query meta database including templates for SQL queries. During knowledge module authoring the medical expert only refers to a code in the query meta database. Our method uses standard tools so it can be implemented on different platforms and linked to different clinical databases.

INTRODUCTION

To increase the use of medical DSSs, integration of DSS with clinical databases is important. In runtime, the DSS need to be linked to clinical databases since DSSs that require manual entry of data will not be widely used [1]. Since the terms for describing data and for representing knowledge often are not the same, the linking is not a simple task. The integration process should be straightforward for the DSS developer when data queries are mapped to the current database environment. Clinical databases at different hospitals usually do not have the same data model and naming conventions and makes it therefore difficult to find a standardized way to access patient data. Standard query languages, such as SQL [2], can be used to access some clinical databases, but all of them do not understand the SQL language and many have their own SQL dialect. In addition, the SQL is not powerful enough to perform the complex temporal operations which are often used by database queries in the KB.

Arden Syntax is an attempt to facilitate sharing of computerized health knowledge bases in the medical domain [3]. In Arden Syntax the knowledge base is represented as a set of discrete modules called Medical Logic Modules (MLMs). Today, many clinical software developers use the Arden Syntax standard to represent knowledge. Earlier work with DSS based on Arden syntax has used different approaches for mapping data queries to clinical databases. In the MLM-based DSS at Columbia Presbyterian Medical Center (CPMC) the mapping clause refers to a data access module (DAM) that is called in runtime to retrieve clinical data [4], and in Linköping one approach is to express SQL syntax within the curly brackets of the mapping clause in the data statement [5].

Standard query languages such as SQL do not permit all temporal calculations required in systems for practice guidelines and clinical protocols, but it has been shown that SQL can encode most of the temporal conditions (96%) used in protocol-based queries [6]. Extensions to SQL, to fully support temporal calculations, have been discussed by Das et al. [7] and Sujansky et al. [8].

The goal of this work has been to develop a standard mapping method that can handle Arden data queries for implementation in a clinical environment. Since Arden data queries only support time-stamped data, a method less complex than the methods referred above [7, 8], where interval-stamped data are handled, can be applied. In this paper we present a methodology for adaptation of Arden data queries in a standardized way. The method is used to integrate DSS based on Arden knowledge modules with existing clinical databases. Our approach combines a standardized query language, SQL, with a uniform access method, ODBC (Microsoft, Inc.), to access all major database systems where patient data are stored. Local data access information, such as names of attributes and relations, is stored outside the MLM, which reduces the institution-specific parts and makes the MLMs more sharable.

ARDEN DATA MAPPING

Since patient data often reside in different databases and sometimes on different platforms it is important to find a common way to retrieve the data. Arden syntax includes support for database queries, temporal calculations, aggregation operators, and other mathematical functions. An MLM is an ASCII file and is composed of slots grouped into three categories: maintenance, library, and knowledge, where the knowledge category consists of parts for data access and logic, written as if-then-else rules. All data mapping within an Arden syntax knowledge module is specified in the data slot. Knowledge modules, written in the Arden Syntax

0195—4210/96/$5.00 © 1996 AMIA, Inc.

language, are independent of the underlying patient database model and the database access language used to retrieve patient data. All information concerning data model and database language is specified in the data slot as a mapping in the institution-specific part of the query, within curly brackets, { }. Other parts of the MLM query are time constraints and aggregation operators. The time constraints are given to set a time interval for the requested patient data and the aggregator operator can finally be applied to the data after the institution-specific mapping and time constraints. A data query statement is written in the data slot of the knowledge category in an Arden syntax MLM according to the following syntax:

<var > := READ <aggregation>
(<mapping> WHERE <constraint>);

The variable <var> is assigned to the result of the query, <aggregation> is an aggregation operator or a transformation operator, <mapping> is an institution-specific part, within curly brackets, that contains necessary terms or syntax, and <constraint> is an operator that specifies time constraints for the query, which can be temporal calculations such as timepoints, intervals, and duration. Patient data are selected in the query by first applying the mapping clause, then the time constraints, and finally the aggregation operator.

METHODS

To perform the different parts of the query process, the data statement in the MLM is translated into a C++ class method as described in earlier work [9, 10]. In the class method the mapping and constraint parts are translated into a SQL query that is used to retrieve patient data via an ODBC driver in runtime. Finally, the aggregation operator is used on the result from the SQL query to get the requested data. An example of an MLM data query, specified in the knowledge category of the MLM, is shown below in figure 1. In the data slot the last Haemoglobin value that occurred within the last 5 days is requested from a patient database and later used in the logic part of the MLM. "QU1393", which is a standard code from the code system IFCC/IUPAC [11] that is under implementation for use in Swedish laboratory medicine, is mapped to access Haemoglobin data in the local patient database. The time constraints are used to select only the records that occurred within the past 5 days and the aggregator operator is applied to select the last result. All time constraints within the Arden syntax can be expressed as a lower and an upper time limit and a flag indicates if the valid time constraint is between or outside those time limits [12].

```
knowledge:
        data:
        Haemoglobin := READ last ({"QU1393"}
        WHERE it occurred within the past 5 days);
        .......
        logic:
        if Haemoglobin < 100 then
                conclude true;
        endif;
```

Figure 1. Part of an MLM where patient data are requested in the data slot and used in the logic slot.

The developed method, presented in this paper, leaves it to the MLM author to decide *what* data to access in the patient database, and to the database administrator (DA) to specify *how* to retrieve the data. In order to make the data query process automatic and compatible regarding different databases, we propose a method with the following steps:

- The DA builds a *query meta database* for the MLM author. This database includes a data dictionary (DD) code for all available patient data.
- In the *MLM authoring* phase the MLM author specifies required patient data and refers to the DD code in the mapping part of the query.
- When patient data are requested in runtime, mapping to the specified patient database is done automatically by the *query engine*.

Query meta database

A query meta database supports translation of both the mapping part and the time constraints into SQL syntax for the specific database and is used in the development phase for adaptation of Arden queries to the local patient database system. The query meta database is also used in the runtime phase by the DSS when an MLM is executed and patient data are requested in the data slot. Automatic generation of SQL queries in runtime can then be achieved. Templates for SQL queries have been prepared in advance by the DA by updating the query meta database. The query meta database contains the DD code, a SQL query template, and the parameters required to select patient data and time pairs from the local database. An example with a patient database where all relevant information is stored in a single database table is shown in table 1.

Table 1. A single patient database table with some Haemoglobin values for a patient.

Patient_ID	*Haemo*	*Measuring_time*
121212-1212	122	1996-03-10T12:00:00
121212-1212	125	1996-03-16T12:00:00
121212-1212	132	1996-03-18T12:00:00

In table 1 the patient database consists of pairs of Haemoglobin values and the corresponding measuring time. The task for the DA is to generate a SQL query template which results in a table with two fields, value and its associated primary time. To achieve this, the DA must be familiar with the local data model, the coding system, and the naming of attributes and relations. To optimize the database queries it is important to have knowledge about indexes used in the database. All this information is stored in the query meta database. In figure 2 the database tables in the query meta database are shown with information for one DD code. In the Code table information about the DD code, description, code system, and unit is stored together with reference to the Syntax table where a SQL query template for the DD code is stored. The Parameter table contains specific parameters for each DD code which are used in the SQL query template to construct a valid SQL query. Since the query meta database is separated into different tables, each entry in the Code table relates to many entries in the Parameter table (1:n) and more than one entry in the Syntax table can relate to the same entry in the Code table (n:1). This makes it possible for many DD codes to share the same query id.

Code table

DD code	*Description*	*Code system*	*Unit*	*Query ID*
QU1393	Haemoglobin	IFCC/IUPAC	g/L	201

Parameter table

DD code	*Number*	*Parameter*
QU1393	1	hb_database
QU1393	2	Haemoglobin
QU1393	3	Measuring_time
QU1393	4	Patient_ID
QU1393	5	QEDBF

Syntax table

Query ID	*Type*	*Connect*	*Syntax*
201	SQL	DSN=$PAR5	SELECT $PAR2, $PAR3 FROM $PAR1 WHERE $PAR4 = ? AND $PAR3 $NOT BETWEEN $T1 AND $T2 ORDER BY 2 ASC

Figure 2. Tables in the query meta database with information for one specific DD code, "QU1393".

The query meta database can be implemented in any database management system (DBMS). In the prototype system, built to test and verify the described method, we have used ODBC to access the query meta database. Using ODBC makes it possible to select any DBMS on the market that has an ODBC driver. To maintain the query meta database we have selected Access (Microsoft, Inc.), a software product for development of database applications.

MLM authoring

During development of the MLM, when the domain expert writes knowledge modules, the MLM author specifies required patient data and refers to those data in the mapping part of the MLM data statement. For the MLM example in figure 1, where Haemoglobin data are queried, the MLM author finds the data dictionary code (DD code) for Haemoglobin in the query meta database and refers to it ("QU1393") in the mapping part of the query. In the MLM authoring process, the author interacts with a terminology server [13] for controlled terminology support, with the query meta database for correct DD codes, and with the MLM knowledge base where MLMs are stored.

Query engine

In runtime, a query engine within the DSS controller maps the Arden query into an executable form. Information stored in the query meta database, together with the time constraints for the query, is used to build a SQL query to retrieve data for a specific patient. The DD code is used to find the parameters and the corresponding query model. Parameters in the SQL query template are substituted for in runtime and a correct SQL query is sent to the patient database via an ODBC driver. A correct driver is selected by the connection string provided in the query meta database. If patient data are moved to another database system, only the connection string in the query meta database is changed. Time constraint variables in the query are substituted for in runtime with the current time limits. Figure 3 displays the process in the query engine where an Arden query is mapped to a patient database and data are retrieved. The query passes through several steps in the process. In the steps described below the step numbers refer to the corresponding numbers in figure 3, and the MLM query example is from figure 1.

1. According to the DSS method described by Gao et al. [10], the data slot part of the MLM is translated into a C++ class method, *read_last()*, in the MLM compilation / translation phase. During execution, the class method is called with necessary parameters for the MLM. For the example shown in figure 1 the class method call will be as follows: *Haemoglobin.read_last ("121212-1212","QU1393", T1, T2, IS);* where T1 and T2 refer to the lower and upper limits of the time constraints and IS indicates that the valid time constraint *is* within the limits.

2. The Arden query preprocessor performs the query in runtime by sending the mapping part (DD code) of the query to the query meta database to retrieve a SQL query template and DD code specific parameters. The time constraint parameters (T1, T2, IS) are sent to the SQL

query builder and the aggregation operator (last) is stored and later used by the Arden query postprocessor.

3. The collected SQL query template and specific parameters are used, together with the time constraint parameters, by the SQL query builder to generate a SQL query.

4. The SQL query is sent to the patient database via an ODBC driver. This query will result in a table with all Haemoglobin data for the specified time constraints. Figure 4 shows the generated SQL syntax for the MLM query example.

5. The SQL query result is sent to the Arden query postprocessor where the aggregation operator, *last*, in the Arden query is used to select the last of the results.

6. Requested patient data are then returned and used in the logic part of the executing MLM.

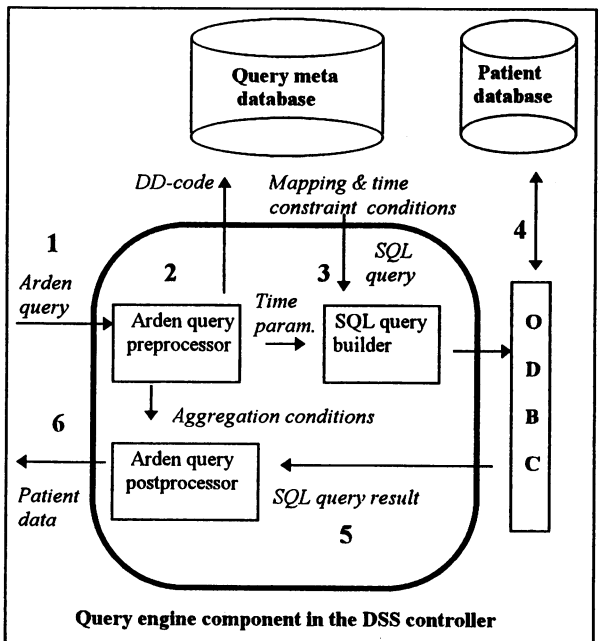

Figure 3. Interaction between the query meta database, the patient database and the query engine in the DSS controller.

```
SELECT Haemoglobin, Measuring_time FROM hb_database
WHERE Patient_ID = '121212-1212'
AND Measuring_Time BETWEEN '1996-03-15T12:00:00'
AND '1996-03-20T12:00:00'
ORDER BY 2 ASC
```

Figure 4. Generated SQL query from the query model shown in figure 2, assuming evoke time is 1996-03-20T12:00:00 and patient ID is 121212-1212.

Managing different database models

Patient databases are often based on data models with different relations between tables. In the example, the mapping method was used for a patient database where data reside in a single database table. If patient data are stored in several tables the syntax will be more complex.

In the described example, the time values in the databases are stored according to the ISO [14] format 'YYYY-MM-DDTHH:MM:SS' used in the Arden syntax. Many patient databases do not have that format for stored time values, so the time values need to be formatted for use in the SQL 'SELECT' statement. For the SQL 'WHERE' statement it is important to format the time constraints into the original format of the time field in the patient database. Otherwise the SQL query will not be able to use index for the time field, if available. Using index for time constraints will in most cases give higher performance, especially if a patient has many records in the database. In a client-server environment the workload on the network will be reduced if time constraints are used in the SQL query, even if index is not available. The query meta database makes it possible to store information about the format for time representation in the patient database.

DISCUSSION

We have shown that it is possible to adapt data queries within MLMs to different clinical database models using the method presented here. With the method, patient data are automatically accessed from different databases in runtime, in a standardized way. The MLM author refers to *what* patient data is needed in the MLM, no specification of *how* the data is retrieved is needed during the MLM authoring phase. How the data are accessed is described in the query meta database by the database administrator.

All Arden mapping and time constraint operators are expressed in the SQL language, and the aggregation operators are performed by class methods since it is not possible to translate all Arden aggregation operators into SQL syntax. This strategy makes it possible to optimize the SQL queries by using the available indexes in the database.

In our method we have selected ODBC to access the patient databases. ODBC is used to build applications independent of database platforms, which makes it possible to access different databases, ranging from text files to relational databases, in a uniform way. Even some object oriented databases, such as ObjectStore (Object

Design, Inc.), have provided the ability to use ODBC to access data. The SQL language is used in ODBC, but the same dialect is used for all databases, even for the ones that do not normally support SQL. Changing the patient database to a new location or to another DBMS will not affect the MLM, as only the query meta database will be updated by the database administrator. Access to the query meta database is also managed by ODBC. This gives a DSS that is portable between different operating and database systems.

There are several advantages to this method. The MLM author needs no knowledge about the database models or how to access the clinical databases. It is not necessary to define a SQL syntax for every attribute in the patient database since similar attributes can share the same syntax. Dynamic changes of the query meta database are possible without modifying the DSS controller. Since all parts of the DSS are built with standard tools that are available on most platforms, the DSS can be ported to different platforms and integrated with different clinical databases.

The proposed method needs to be further evaluated, especially from the MLM authoring perspective [13]. Further development will include integration of MLM editing functions with general purpose terminology servers [15] and the proposed query meta database server. The expectation is that such an environment will facilitate both the knowledge acquisition process and the local database integration issues.

Acknowledgements

This work was supported by the Swedish National Board for Industrial and Technical Development (P2475).

References

1. Hripcsak G, Johnson SB, Clayton PD. Desperately seeking data: Knowledge base - Database links. In: Proc. 17th Annual Symposium on Computer Applications in Medical Care/SCAMC, McGraw-Hill, 1993; 639-43.
2. Date CJ, Darwen H. A guide to the SQL standard. Addison-Wesley 1993.
3. Hripcsak G, Clayton PD, Pryor TA, Haug P, Wigertz OB, Van der Lei J. The Arden Syntax for Medical Logic Modules. In: Proc 14th SCAMC, IEEE Comp Soc Press, 1990; 200-4.
4. Johnson SB, Hripcsak G, Chen J, Clayton PD. Accessing the Columbia repository. Proc 18th SCAMC, Hanley & Belfus, Inc., 1994; 281-285.
5. Johansson B and Bergqvist Y. Integrating Decision Support, based on the Arden Syntax, in a Clinical Laboratory Environment. Proc. In: 17th Annual Symposium on Computer Applications in Medical Care/SCAMC, McGraw-Hill,1993;394-8.
6. Das AK, Musen MA. A comparison of the temporal expressiveness of three database query methods. In: Proc. 19th Annual Symposium on Computer Applications in Medical Care/SCAMC, Hanley & Belfus, 1995; 331-7.
7. Das AK, Musen MA. A temporal query system for protocol-directed decision support. Meth Inform Med 1994; 33: 358-70.
8. Sujansky W, Altman R. Towards a standard query model for sharing decision-support applications. In: Proc.18th Annual Symposium on Computer Applications in Medical Care/SCAMC, Hanley & Belfus, 1994; 325-31.
9. Johansson B and Wigertz O. An Object Oriented Approach to Interpret Medical Knowledge Based on the Arden Syntax. In: Proc. 16th Annual Symposium on Computer Applications in Medical Care/SCAMC, McGraw-Hill, 1992; 52-6.
10. Gao X, Johansson B, Shahsavar N, Arkad K, Åhlfeldt H and Wigertz O. Pre-compiling Medical Logic Modules into C++ in Building Medical Decision Support Systems. Computer Methods and Programs in Biomedicine 1993; 41: 107-19.
11. Rigg, Brown, Bybkeae, Olesen. In: Compendium of terminology and nomenclature of properties in clinical laboratory medicine. Blackwell Science, 1995 (in press).
12. Arkad K, Gao X-M, Åhlfeldt H. Query handling in MLM-based Decision Support Systems. Medical Informatics (accepted).
13. Ahlfeldt H, Shahsavar N, Gao X, Arkad K, Johansson B, Wigertz O. Data driven decision support based on the Arden Syntax within the HELIOS environment. Comp Methods Progr Biomed 1994; 45; Suppl S97-S106.
14. International Organization for Standardization, Data elements and Interchange Formats - Information Interchange - Representations of Dates and Times (ISO 8601:1988E). International Organization for Standardization, Switzerland, 1988.
15. Rector AL, Solomon WD, Nowlan WA, Rush TW, Zanstra PE, Claassen WMA. A Terminology Server for Medical Language and Medical Information Systems. Meth Inform Med 1995; 34: 147-57.