

An Object-Oriented Tool for Clinical Queries

Roy Schoenberg¹, Ido Schoenberg², Charles Safran¹

¹Center for Clinical Computing, Beth Israel Deaconess Medical Center, Harvard Medical School
²iMD-Soft Inc.

Abstract: *Clinical data repositories are viewed as digital goldmines for both clinical researchers and managed care utilization reviewers. Querying the data is, nevertheless, challenging due to the heterogeneity of the sources and the lack of intuitive interfaces for question makers. We present a prototype of a query tool that uses the object metaphor to make sense to users and database administrators alike.*

As health organizations around the country build their information systems and expand the scope of digital acquisition, abundant data is collected and saved in clinical repositories. Traditionally, such repositories were built to mimic paper records allowing swift access to a single file at a time. Querying the databases in cross-sectional (population) manner was seldom conceived as a requirement. Managed care via its financial tracking and utilization reviews has rectified this oversight at the same time as researchers began to realize the power in querying such databases¹. Unfortunately, getting meaningful answers typically requires understanding in both the structure of the underlying database as well as in query languages². Paradoxically, as clinical systems grow more sophisticated and comprehensive, it is likely that vendors will be required to provide simpler ways to query them³. In this paper, we present a prototype of a tool targeted at this very goal. The tool was developed in collaboration with iMD-SoftTM, a clinical information system vendor.

To make the tool applicable to a wide range of systems we have made a distinction between the underlying data layer and the user interface. We have categorized patient data into 6 objects, namely: patients (demographical data), signals (continuous data like heart rate readings), orders (medications, interventions etc.), problems (problem list entries), diagnoses (ICD-9 compatible entries), and notes (progress notes). Each of the objects can be populated with a list of identifiers that point to actual records (or SQL “views”) in the underlying database. All identifiers in a specific object point to the same type of data structure. For example – each identifier in a “patients” object points to a record containing fields like first and last name, whereas identifiers in an “orders” object will point to drug-name, schedule and

route. Multiple instances of each of the six objects can co-exist, each having the same structure but different content depending on the identifiers that populate them.

Each of the six objects was then tailored with a set of “qualifiers”. Such qualifiers are the procedures that allow the user to filter the population of an object using specific criteria. Because the data structure in each object is homogenous, adding tailored qualifiers is straightforward. For example, we created a qualifier for the “patients” object that requires the first name to follow a certain pattern (e.g. start with an “S”, or be equal to “John”). The user can specify criteria for any number of qualifiers for the object. Executing the set returns a new object of the same type with a subset of the original population that complies with the criteria in the qualifiers. In this way we can create an object holding all male patients over 35 years of age, or all orders for over 40mg of IV Lasix on a given day.

The Six objects can now communicate. Using a “convert” function, each object transform its population to another object type. If we convert the patients object: “male patients over 35” to an “orders” object, we get the orders prescribed to those patients. In the opposite direction, we get the patients who were prescribed with Lasix. We can use this method to sequence queries and retrieve specific subsets like “progress notes of males over 35, who were diagnosed with both MI and diabetes, received Lasix and had a hypokalemic event”. In addition to the convert function, the system supports object merging (e.g. AND) and crossing (e.g. XOR). After the final object has been defined, a report generator is deployed to produce elaborate reports based on the population identified in the object.

1 Murphy SN; Morgan MM; Barnett GO; Chueh HC; Optimizing healthcare research data warehouse design through past COSTAR query analysis; Proc AMIA Symp 1999;:892-6

2 Stein HD; Nadkarni P; Erdos J; Miller PL; Exploring the degree of concordance of coded and textual data in answering clinical queries from a clinical data repository; J Am Med Inform Assoc 2000 Jan-Feb;7(1):42-54

3 O'Connor MJ; Tu SW; Musen MA; Applying temporal joins to clinical databases; Proc AMIA Symp 1999;:335-9