The PatientBank Health Graph

A Flexible Data Model for Health Information

Paul Fletcher-Hill
Advised by Ruzica Piskac

December 18, 2014

Abstract

PatientBank is a service to help patients take control of their health information. It is an integrated health record system, offering a personal health record to patients and a health information exchange to providers. At the core of the product is a data framework we call the PatientBank Health Graph. Built from a few simple components, the Health Graph is able to integrate a patient’s medical information from across the health care system and represent it in a meaningful and scalable way.

Background and Motivation

Over the last five years, the United States has undergone more significant changes to its health care system perhaps since Medicare and Medicaid were introduced in the 1960s. The Health Information Technology for Economic and Clinical Health (HITECH) Act in 2009 and the Patient Protection and Affordable Care Act of 2010 have paved the way for tremendous changes to the system’s information backbone and providers across the country have begun using electronic health record systems in their practices.

The fundamental problem with this transformation is that the electronic health record systems lack interoperability—a provider using Epic Systems cannot transfer a patient’s record to another provider using Cerner, for instance. The real benefit of a digital system is its ability to quickly and effectively share information between providers, but the US system currently lacks that ability. As a
response, a number of health information exchanges (HIE) have been proposed or attempted. These are information systems that construct interfaces between certain electronic health record vendors or health systems to share health data. However, these have also proved incredibly difficult to build and finance.

We started PatientBank as an alternative solution to this problem. Instead of taking a top-down approach to health information exchange (where providers try to merge their respective data), why not build a product that encourages patients to aggregate their own health information and share it with family members and providers? The result is the same: providers have access to full patient health records, but the patients are the ones driving change rather than providers.

The first version of PatientBank was developed for Professor Ruzica Piskacs CPSC 439 Software Engineering class in the spring of 2014. The project included a personal health record for patients to pull together their health records and view them online. The original team included Feridun Mert Celebi, Paul Fletcher-Hill, and Rafi Khan. Over the summer of 2014, Celebi and Fletcher-Hill continued to develop parts of the application, including two satellite applications for administrators to view patient information and providers to upload health records.

In the fall of 2014, a group of students has continued to work on PatientBank, from both a technical and business perspective. This group includes Paul Fletcher-Hill, Feridun Mert Celebi, Kevin Grassi, Graham Kaemmer, Mikayla Thompson, and Abhishek Chandra. Celebi, Thompson, and Chandra all also worked on various aspects of the PatientBank platform in the scope of CPSC 490.

While there are a number of fascinating technical problems to discuss at the intersection of health care and technology, this paper focuses on the challenges of storing health information from disparate sources. Currently, early PatientBank users simply store their medical records as PDF files on the platform. However, as the team’s ability to integrate these records improves and expands, the platform will migrate to what we are calling the PatientBank Health Graph. The Health Graph is a framework that efficiently integrates information into patients’ records while maintaining and growing an ontology for classifying the integrated health information.
System Design

The PatientBank Health Graph has two primary components: a generic record framework and patient-specific streams of data. The record framework operates as a classification system for the health information associated with each patient. The Health Graph was designed as a flexible system that could be deployed universally, interoperating with any health record framework. Instead of building a relational database with thousands of customized tables for each component of a patient’s health record, we abstracted away the details of the graph into two distinct concepts—the RecordSection and RecordSectionField classes.

Three additional classes are required to represent actual health data, and those are the RecordUpdate, RecordUpdateValue and RecordUpdateGrouping classes. Instances of these classes are always associated with a patient (in contrast with the standalone RecordSection and RecordSectionField classes) and a specific node in the graph, facilitating the process of writing to the graph. The framework provides contextual information for patients’ collections of data points.

Diving into the graph

Health records are just glorified trees. At the root, we have the record, with each additional level representing another degree of specificity. An allergic reaction to Amoxicillin would be classified as a child to the Allergies branch, for instance. As a more detailed example, consider the International Classification of Diseases (ICD) framework. The system is maintained by the World Health Organization (WHO) and was designed to help providers and medical professionals classify diseases and symptoms more consistently. There have been multiple versions of the framework, but the most recent version, ICD-9, lists seventeen root-level folders. The codes take on a three-digit format to represent their location, with additional digits added after the decimal point to add specificity. For example, the ICD-9 code for “Acute alcoholic intoxication in alcoholism, unspecified” is 303.00, where 303 represents all “Alcohol dependence syndrome” codes. The code is embedded within “Mental Disorders” and then “Neurotic Disorders, Personality Disorders, And Other Nonpsychotic Mental Disorders.”

The ICD framework is one of many coding frameworks for health information. The Logical Observation Identifiers Names and Codes (LOINC) is another framework used primarily for encoding laboratory results.

The Health Graph uses two concepts to represent the generalized record frame-
work: the RecordSection and RecordSectionField classes. Instances of the RecordSection class are equivalent to the nodes of the record framework tree, and RecordSectionField instances are specific attributes of those sections. As an example, consider a Complete Blood Count (CBC) with Differential test result. The LOINC code for this specific test is 57021-8. A CBC with Diff test result has multiple values, such as Hemoglobin density and Platelet count, each of which has its own unit and reference range. As you can see in the diagram below, the PatientBank Health Graph would represent this example with two RecordSection instances Test Results and Complete Blood Count with Differential and RecordSectionField instances for the specific attributes of each test. A Complete Blood Count with Differential actually has far more fields than are shown in the diagram, but this is a conceptual exercise.

![Diagram of record section hierarchy]

Figure 1: Complete Blood Count with Differential Example

The record framework is generalized across all patients, and it evolves as data is integrated into the graph. Whereas another system might fail by not offering enough complexity up front, the PatientBank Health Graph is simple and flexible enough to adapt to changes in how records are organized and managed. In fact, the graph is initialized without any data. As health records are added to it, it builds up the framework around this data, using metadata from classification standards like ICD-9 and LOINC.

Adding some data

While some health record systems store only current information about a patient, PatientBank aims to represent patients entire medical histories. Conceptually, the model is very similar to Facebook’s Timeline, Twitter, or other social
media streams: collections of granular data points are stored on each patient, each classified using the RecordSection and RecordSectionField graph.

Each visit to the doctor’s office generates an update to a patient’s health record. The doctor might adjust a patient’s height, weight, blood pressure, or other measures. Each of these values is just another data point in the patient’s medical history, which we represent with a RecordUpdate class. A RecordUpdate instance is simply a timestamped object containing one or multiple RecordUpdateValue instances—the actual data points. Another class, the RecordUpdateGrouping class, wraps the collection of RecordUpdate instances for each appointment or procedure.

For example, a patient might have her blood pressure measured. This data point would be represented as a RecordUpdate instance, timestamped to when it was measured, and with two RecordUpdateValue instances—one for systolic and another for diastolic pressure.

![Blood Pressure RecordSection with RecordUpdate instances](image)

Figure 2: Blood Pressure RecordSection with RecordUpdate instances

It is important to understand that the RecordUpdate class does not store any contextual information, it simply references a RecordSection instance in the health graph. In the case above, the Blood Pressure RecordSection instance
classifies the RecordUpdate object from December 17, 2014. Instances of the RecordUpdateValue class are associated with the RecordSectionField objects belonging to the specified RecordSection instance, which provide units for these values.

The resulting data model is a clean and efficient separation of data points and their context, which speeds up querying and minimizes redundancy.

Implementation

The PatientBank Health Graph has been implemented using the Ruby on Rails framework, the ActiveRecord Query Interface for building associations and data binding, and PostgreSQL as an object-relational datastore.

Acknowledgements

Thank you to Professor Ruzica Piskac for advising me and the rest of the PatientBank team this semester. The design and implementation of the platform has been a time-consuming (and sometimes frustrating) process, but I have had the pleasure of working with incredible people throughout. Thank you to the PatientBank team and anyone who has contributed ideas, code, or just raw enthusiasm over the last year.