

## **Integrating Clinical Data into the i2b2 Repository**

Aaron Abend, MBA, Managing Director; Dan Housman, BS, Managing Director; Bruce Johnson, BA, Managing Director; Recombinant Data Corp, 391 Totten Pond Road, Waltham, MA 02451

The Clinical Research Chart, developed at the National Center for Biomedical Computing known as i2b2 (Informatics for Integrating Biology and the Bedside) is gaining acceptance as an important addition to NIH's software to support translational research. The i2b2 platform allows an institution to integrate a wide variety of clinical data sources into a single repository, and it supports the integration of genomic data for clinical and translational research (1).

Developing a clinical data repository from disparate data sources is often a bottleneck for translational research programs because the effort requires collaboration among technical personnel from different parts of the organization. The individuals working with the EMR work in one department, those working with claims data work in another, and the systems collecting data from Intensive Care Units, Operating Rooms, and Emergency Rooms are often stand-alone systems each with its own system architecture, database, and administrative policies. Cooperation among the various groups is therefore both administratively and technically challenging.(2)

This paper describes our approach to integrating clinical data into the i2b2 repository, including the use of EMR data, the use of medical claims data as a surrogate and supplement for EMR data, and genomic data. The approach uses 3 zones to manage the transformation and integration process. Each zone provides isolation of processes that provide the data from those that use the data. This makes data validation and data quality control processing possible. The authors have used this approach for successful i2b2 deployments at several academic medical centers.

Data validation metrics are collected during operation and analyzed each time new data is loaded, thereby allowing assessment of the accuracy of the processed data.

### **Architecture**

The Clinical Repository is set up as a set of databases that we call "zones". In Microsoft SQL Server, each zone is a database. In Oracle, each zone is an account ("schema").

There are 3 zones required in our architecture:

1. Landing Zones provide the entry point for all new data (there is one landing zone for each data source)
2. The Working Zone is where the "heavy lifting" takes place. Data from the landing zones is integrated, de-duplicated, and cleaned in the working zone
3. A Staging Zone is the working zone for a particular application. For research, the Research Staging Zone is an i2b2 data schema. Algorithms for creating the dimensions, aggregations, and ontological mappings necessary for research are implemented in this zone.

When the Staging Zone is ready for production, it is copied or moved into Production.

### **The Landing Zone - Source Data**

Acquisition of source data is outside the scope of this paper. In our production system, all sources are acquired in one or more flat-files that permit loading into relational database tables. The processes that acquire these files gives each file a unique name. Ideally, the files from each source will have the same structure. In many systems, the formats often change from month to month, generating exceptions that require programmer intervention. This problem is not going to be addressed here.

Each source file is loaded into a “loading table” that matches the column structure of the inbound file. All of the columns in the loading table are defined as variable character (“varchar”) so that no data type failures are encountered. The lengths are defined with the smallest length necessary to guarantee that every record will fit. When more than one file is loaded, we add a column to indicate from which file each record originates from to help in validation.

### ***Data Type Normalization***

After all of the flat files are loaded into loading tables, each loading table is transformed into a table with the same structure, but valid data types. In this process, we typically make the following transformations:

1. All dates are validated and converted to actual dates in the database. Invalid dates are either 1) set to “null”, 2) set to a rule-based value, or 3) the record is marked as invalid
2. All numeric data types are validated and converted to valid numbers with the correct decimal precision. Invalid numbers are either 1) set to “null”, 2) set to a rule-based value, or 3) the record is marked as invalid.
3. Character (text) values are checked to ensure special characters have not caused truncation of important data, and text is trimmed to remove invalid characters and spaces at the end of otherwise valid data.
4. Ensure that the final table has no duplicate records. (If a number of flat files come from the same source, it is possible that there is overlap among the data in the files.)

### ***Common Pitfalls in the Landing Zone***

1. Failure to validate dates properly. Date formats from source systems vary widely, and sometimes (rarely) formats can change. Do not rely on implicit conversions – make sure all formats are explicitly converted and that data is validated (e.g., ensure no birth dates in the future). Correct handling of centuries is, of course, important and must be checked. In our system, an early file provided no century for birth dates, causing ambiguity and requiring rewrite of the extraction program.
2. Attempting to do too much data correction. For example, attempting to clean up names in the landing zone will result in the same task being performed in each of the landing zones in the repository. Whenever there is doubt, processing should be performed later in the process, in the working zone, to ensure consistent handling across all data sources.

## **The Working Zone - Integration**

### ***Processing***

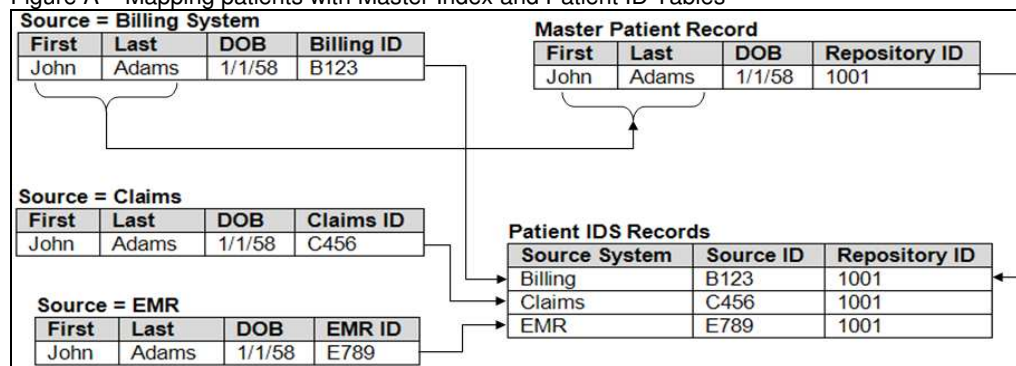
In the Working Zone, data from different sources is integrated. The first step in this process is the integration of dimensions. In clinical data repositories, the dimensions always include the patient, the provider, and the concept. In addition, there may be clinics, encounters, hospitals, and other concepts that must be integrated across data sources. An integration process is needed for each normalized concept. The steps are the same for each concept you integrate.

For an example we will use patient information. To start, it is necessary to identify the set of characteristics that will be used to uniquely identify a patient. Ideally, this will be a medical record number that all sources use, and if that is your situation, no special processing is needed. Just use that medical record number. In such cases, an Enterprise Master Patient Index system is usually in place to make sure patient identification is consistent.

In many cases, however, there is data that does not have a common record number. In these situations an alternative key is needed to uniquely identify a patient. We have found that first name, last name, and date of birth is a highly accurate identifier. Middle initial can add additional accuracy but more often introduces substantial duplication middle initials are not always tracked. In those situations, there are many ambiguous records. Since the purpose of this repository is research, the small level of accidental merging of records with the same name and birth date is regarded as acceptable.

When a new patient record is received, its key is checked against a master index of first, last, date of birth values. If the value is found, the repository's identifier for that patient record is added to an index of patient ids. This table contains the identifier for the patient in the source system, the source name, and the repository identifier. This process is repeated until the master index contains one record for each patient, and the patient IDs table contains a record that matches the master index ID (the repository's key for that patient) with each ID used in each source system for that patient (see Figure A)

Figure A – Mapping patients with Master Index and Patient ID Tables



### Concept Mapping

The mapping of concept domains such as problems, medications, lab tests, etc. presents different challenges. Problem lists frequently combine free form text with optional codes such as ICD9 and CPT. In order to avoid missing important concepts, it may be necessary to append additional values to standard dictionaries. This requires a separate concept table that stores new repository-only codes for data that lacks a standard dictionary code.

This example uses problems from an EMR system. Each record contains 3 values of potential interest: 1) a short description labeled "problem", 2) a long description labeled "comment", and 3) a field labeled "dx\_code". Analysis shows that the dx\_code is optional, that it contains both ICD9 codes in a A999.99 format and CPT values with a 5 digit format ("99999"), and that 2 records with the same dx\_code can have different "problems".

The first step is to extract dx\_codes and put them into a concept lookup table that maps the concept identifier (such as the ICD9 code) to a repository ID. This is necessary because we do not want our keys to use ICD or CPT codes. That would limit our ability to map non-coded values. Our coding table maps both codes and problem descriptions. Problem descriptions that match a coded entry are properly coded. A list of all remaining unique "problems" is added to the code list.

We are now in a position to create "fact" records for the repository. A series of passes through the data are the best approach to applying a series of rules that decrease in accuracy. In our limited example, the only source of inaccuracy is the concept coding. Patient coding is assumed to be accomplished through the first/last/dob key. The first pass through the records could use ICD codes to map matching facts. The records successfully mapped with the first pass are removed and the next rule, CPT code matching is applied. The process is repeated with rules that match on the short problem description and, if desired, on increasingly complex algorithms that map.

### **Overlapping Data**

In most cases, there is more than one source being integrated. A patient sees a provider who enters “Diabetes” on a problem list. A bill is created for the diagnosis and is stored in the billing system. Finally a claim is sent to an insurance company and approved for reimbursement. Ideally, the EMR would have all of the data, but unfortunately that is not always the case. Even institutions with fully implemented EMR systems have historical data buried in claims.

To deal with overlapping data it is necessary to determine the priority of each source. As with the patient name, a key for updating each fact must be determined. For problems, for example, the patient ID, service date, provider, and ICD9 code can be used to determine a unique record. The highest priority source should be loaded first. Then, inbound records with a matching key are deleted from that source before the remaining records are imported. This process is repeated for each source until the lowest priority source is loaded.

#### **Problem Facts from Initial EMR Data (first priority)**

Patient	Provider	Date	ICD9	Explanation
John Smith	Dr. Jones	2/3/2008	250.0	Initial diagnosis of diabetes
John Smith	Dr. Jones	6/8/2008	583.81	Initial diagnosis of nephritis

#### **Problem Facts from Claims Data (second priority)**

Patient	Provider	Date	ICD9	Explanation/Processing
John Smith	Dr. Jones	9/3/2007	034.1	Claim, Diagnosis of Strep throat – retained since there is no existing fact
John Smith	Dr. Jones	2/3/2008	250.0	Claim, initial diagnosis of diabetes, will be deleted since it overlaps with a higher priority EMR fact
John Smith	Dr. Jones	6/8/2008	250.0	Claim, diagnosis of diabetes, is retained since it is a second fact (by virtue of the newer date). A second diagnosis record loaded into the repository improves the specificity of queries by providing data that confirms the diagnosis.
John Smith	Dr. Jones	6/8/2008	583.81	Claim, initial diagnosis of nephritis, will be deleted since it overlaps with a higher priority EMR fact

#### **Resulting Repository Problem Facts**

Patient	Provider	Date	ICD9	Source
John Smith	Dr. Jones	9/3/2007	034.1	Claims
John Smith	Dr. Jones	2/3/2008	250.0	EMR
John Smith	Dr. Jones	6/8/2008	250.0	Claims
John Smith	Dr. Jones	6/8/2008	583.81	EMR

### **“Omic” Data**

Genomic, proteomic and other types of biological data that is associated with a specific patient can be loaded into i2b2 as long as it is normalized into facts. It is not necessary that all facts be of one type or another. For example, for a single patient, we might have data on an entire gene, a SNP, or a reference sequence. For each fact, the name of the data element (e.g., “PPP1R2” for a gene or “rs861098” for a RefSeq) is used as the concept and the allele or expression data is entered can be entered as a numeric value as if the record were a lab test:

Patient	Provider	Date	Gene/RefSeq	Expression Value
John Smith	Genomic	4/18/2008	PPP1R2/probe 12345	6
John Smith	Genomic	4/18/2008	rs861098/probe 34567	7

### ***Common Pitfalls in the Working Zone***

1. Creating a master key that allows too many duplicates, or one that will be ambiguous when null values are encountered (e.g., middle initial for a patient or using clinic code in the provider key – providers often work at multiple clinics).
2. Attempting to track sources in the dimension tables - any logic that attempts to retain the source id can inadvertently introduce duplicates since each source will have a matching set of dimensions
3. Failing to identify keys that require integration. For example, the ICD9 code 250.0 does not specify whether the patient has Type 1 or Type 2 diabetes, so patients with both types will get the 250.0 code. Unfortunately, specifying the type of diabetes is only possible in the ICD9 coding system with the specification of other manifestations (ophthalmologic/nephrologic complications). Entering 2 definitions of 250.0 in the problem concept dimension table is incorrect and could cause duplicates to emerge later in the processing.

## **The Staging Zone – i2b2 Schema**

### ***De-identification***

The i2b2 schema includes few of the identifiers precluded by HIPAA from inclusion in a limited data set for research. Zip code is one of those values, and populating with the first 3 digits of the zip code is permitted. To prevent the use of dates to identify individual patients we shift the dates of each patient's data by a random value (all dates for a patient are shifted the same amount, but the degree of shift varies from patient to patient). Finally, the patient's medical record number is replaced by an arbitrarily generated sequence number. This becomes the key in the i2b2 database.

The elimination of the patient medical record number introduces another problem – how to obtain actual patient records once IRB approval is obtained? To accomplish that, we retain an identified version of the database in a secure, limited access database. Once a query is developed using the i2b2 server, the XML document for that query is submitted to the hive in the identified database and the IRB approved data set is generated with the result set.

### ***Loading the data***

The essential tables in the i2b2 database schema are: 1) the Patient\_dimension table, 2) the Concept\_Dimension table, and 3) the Observation\_Fact table. Data from the clinical repository is moved into this simple schema by a set of insert statements:

1. Populate the Patient\_dimension from the Repository. Populate as many data columns as possible (e.g., language, ethnicity, religion), but note that the Patient\_Dimension table is not used when doing queries in i2b2; the Patient\_Dimension table is only used to describe records that are returned by the query, such as in the timeline plug-in. Instead of empty values, populate null values with the '@' sign.
2. Populate the concept\_dimension table by selecting the dimension table for each data type (e.g., problems, medications, labs).
3. Populate the fact table by selecting the facts from each source into the fact table. Whether you load all the dimensions then all of the facts, or the dimension and fact for each data type, is not important.

When the database is loaded, make sure that referential integrity has been maintained by joining the facts to the Concept and Patient dimensions (via the "concept\_cd" and patient\_num columns).

### **Validation**

When data has been loaded into the dimension and fact tables of the repository, the data is tested for integrity to ensure that all repository IDs resolve to values in the dimension tables. Metrics that show how many records have been transferred from the source to the data warehouse are