

Informatics for Integrating Biology and the Bedside (i2b2) Workbench software release of June 1, 2007

The Clinical Research Chart is not a single entity, database or software component, but a conceptual model for structuring clinical research data systematically into a cohesive collection that can be sustained over time for use by clinical investigators. To execute this vision in digital form, the i2b2 Hive is being developed as the software engineering architecture that will support the implementation of the Clinical Research Chart. The characteristics of the i2b2 Hive are driven by our nascent understanding of the current and future comprehensive needs of a Clinical Research Chart; namely, that it will need to serve complex and diverse storage and computation needs while simultaneously maintaining a relatively simple interface for both clinical investigators and developers and integrators of biocomputing resources.

The June 1 2007 i2b2 Workbench software release is summarized as follows, and described in detail in the rest of this document:

1. the I2B2 Hive: a standard, self-aggregating software architecture for the clinical research chart

The integration strategy for i2b2 applications has been, and continues to be, for the use of a service oriented architecture (SOA) that allows the exchange of well defined eXensible Markup Language (XML) messages between software modules. The software modules of the Hive are called "Cells", and represent atomic units of functionality that exist as software programs. The specifics of these software programs, and what makes them a "Cell" will be described below. What is new this year is the client presentation and linking of the cells through an integrating application framework named "Eclipse". This application allows a user to aggregate various pieces of functionality available from individual i2b2 Cells into a common i2b2 Workbench, from which the Hive will work together as a unit to accomplish the work of the Clinical Research Chart.

2. The I2B2 Cell: Programming APIs for clinical research chart "plug-ins"
These are the specifications and software application programming interfaces that will allow anyone to produce new functional modules for the I2B2 clinical research chart environment.

The plug-ins that we have developed for the Eclipse-based i2b2 Workbench are technically not i2b2 Cells. They are client-sided components that communicate with the cells and are visible to the users. These plug-ins expose the functionality of the cells to the users of the software, and allow actions of the i2b2 Cells to be tied together by user interaction. The cells are tied together by having the users direct messages from one plug-in to another, usually through drag and drop. These are generally the same or similar XML messages that are being used for intercommunication between the Cells of the hive. The application that utilizes this technology is available at the i2b2 web site <https://www.i2b2.org>. A demonstration, fully built application is available for download. It has the complete functionality of the full application. It communicates with demonstration cells active outside the firewall of Harvard University. The software is licensed under an open source i2b2 license which is included at the end of this document. The data that is used in the demonstration was carefully prepared and is an aggregate of obfuscated medical record data where the meaning is retained as much as possible, but no record represents the record of a single patient.

3. I2B2 Ontology: Specifications for an ontology and data representation for the clinical research chart.

Specifications for a Version 1 ontology cell, regarding messaging to and from the cell, and general workings of the ontology, are available at the <https://www.i2b2.org> web site. A demonstration cell is operating at <https://www.i2b2.org>. Vocabularies, including International Disease Classification Version 9 – CM (ICD9-CM), National Drug Classification (NDC), and Logical Observation Identifiers Names and Codes (LOINC), are loaded and available from the site to work with the data in the clinical research chart. The source code and design of the ontology cell will become available under the i2b2 open source license during the summer of 2007.

4. Core I2B2 Cells: Minimum software components to instantiate an I2B2 Hive

A demonstration, fully built Data Repository Cell and a demonstration, fully built Project Management Cell are operating at <https://www.i2b2.org>. The source code and design of the Data Repository and the Project Management cell will become available under the i2b2 open source license during the summer of 2007. Demonstration data and Tutorials for the use of the i2b2 Workbench are included and descriptions below outline the interactions with these cells.

5. Functional I2B2 Cells: Specific functionality for the clinical research chart added through optional cells that interoperate within the hive.

Specifications for a Version 1 Pulmonary Function Test (PFT) parsing cell regarding messaging to and from the cell, and general workings of the cell, are available at the <https://www.i2b2.org> web site. A demonstration cell is operating at <https://www.i2b2.org>. The application, source code and design of the PFT cell and its associated i2b2 Workbench plug-in are available under the i2b2 open source license at the <https://www.i2b2.org> web site. Complete documentation is available with the download, serving as a working example of communication using i2b2 XML messaging.

Detailed description of the Eclipse-base i2b2 Client Workbench

Eclipse is an open-source, platform-independent software framework, written primarily in Java, for delivering what the project calls "rich-client applications", as opposed to "thin client" browser-based applications (<http://www.eclipse.org>). So far this framework has typically been used to develop Integrated Development Environments (IDEs), however it can be used to host programs written in many different software languages, each through the concept of a "plug-in". Another program for biology that has been written in this framework is Bioclipse (<http://www.bioclipse.net>), a free, open source, workbench for chemo- and bioinformatics. Eclipse was originally developed by IBM as the successor to its VisualAge family of tools. It is now managed by the Eclipse Foundation, an independent not-for-profit consortium of software industry vendors. Many software tool vendors have embraced Eclipse as a future framework for their IDEs.

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the rich client platform, in contrast to some other IDEs where functionality is typically hard coded. This plug-in mechanism is a lightweight software component framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the plug-in framework allows Eclipse to work with typesetting languages like LaTeX, networking applications such as telnet and database management systems. The plugin architecture supports writing any desired extension to the environment.

The basis for Eclipse is the Rich Client Platform (RCP). The following components constitute the rich client platform:

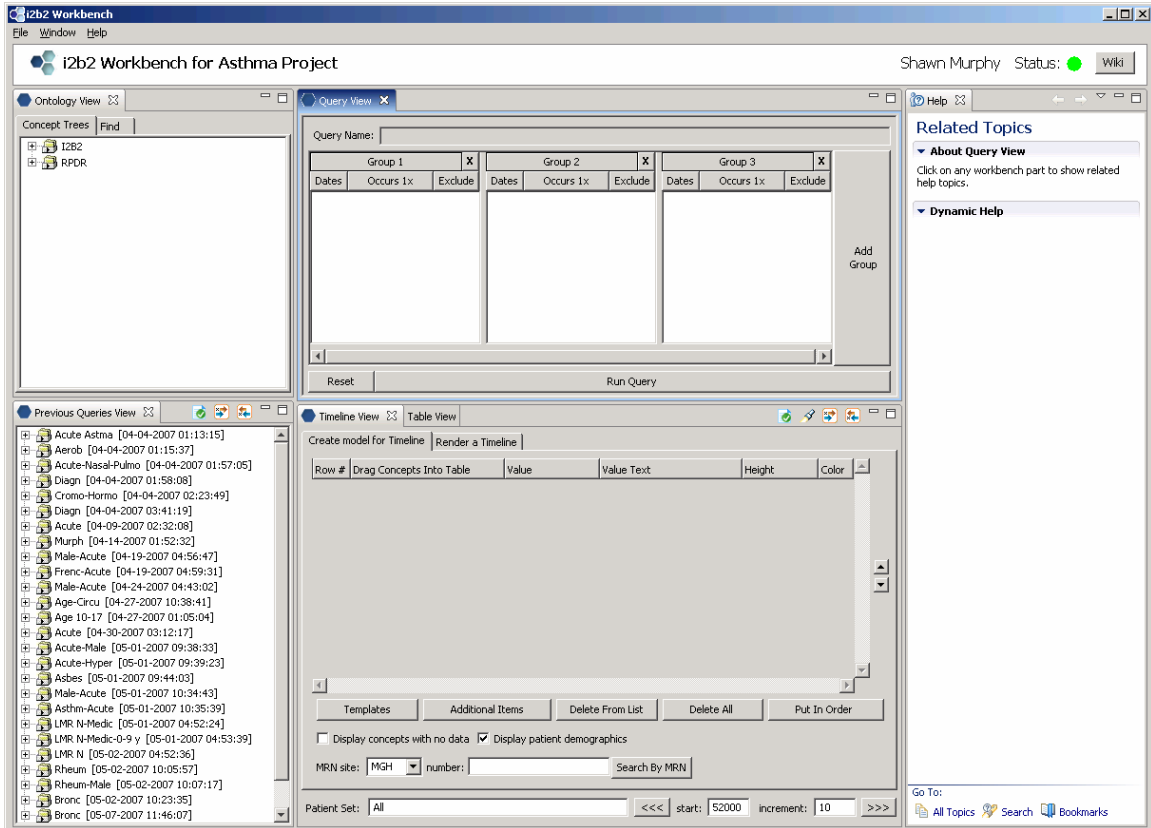
- Core platform - boot Eclipse, run plugins
- OSGi - a standard bundling framework
- the Standard Widget Toolkit (SWT) - a portable widget toolkit
- JFace - file buffers, text handling, text editors
- The Eclipse Workbench - views, editors, perspectives, wizards

The plug-in technology uses OSGi - a standard bundling framework. The OSGi Alliance (formerly known as the Open Services Gateway initiative) is an open standards organization founded in March 1999. The core part of the specifications is a framework that defines an application life cycle model and a service registry. Based on this framework, a large number of OSGi Services have been defined, including Framework Layering which is the standard used for Eclipse plug-ins.

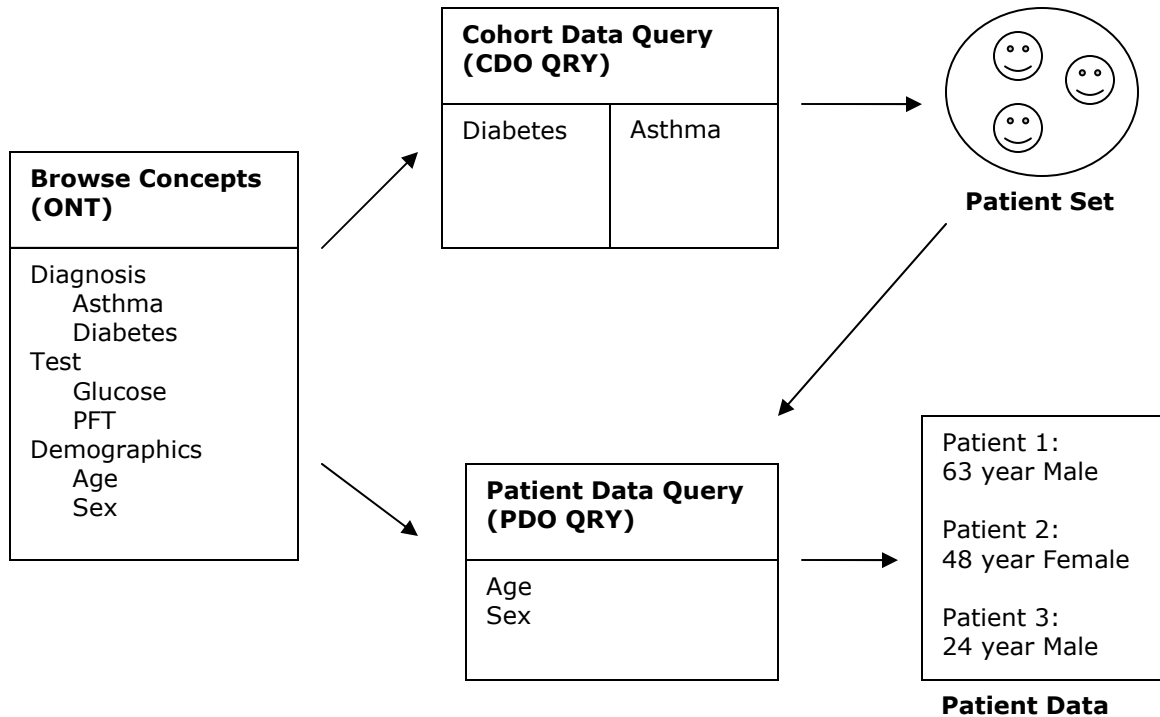
The Framework implements a complete and dynamic component model - something that is missing in standalone Java/VM environments. Applications or components (coming in the form of bundles for deployment) can be remotely installed, started, stopped, updated and uninstalled without requiring a reboot - management of Java packages/classes is specified in great detail. Life cycle management is done via APIs which allow for remote downloading of management policies. The service registry allows bundles to detect new services, or the going away of services, and adapt accordingly

The plug-ins that we have developed for the Eclipse-based i2b2 Workbench are client-sided components that communicate with the i2b2 Cells and help aggregate their functionality in the Hive. Working with the i2b2 Workbench is illustrated in this report as a series of interactions. Descriptions of each plug-in and it's interaction with the cells of the hive will follow, but the series of interactions between plug-ins will show how the Eclipse product ties the i2b2 cells of the Hive together and allows the user to think of them as a single cohesive entity.

After the logon, the user sees the following Workbench. There are actually several different "perspectives" that allow for different arrangements of plug-ins, but the default arrangement is shown below. Each tabbed item is a plug in, and there are six visible in this perspective. The top white bar is also a plug-in, but the user can not close it. It controls the user authentication, user roles, security keys, and the locations of other cells in the hive. It is populated during the login process.



The interaction with this perspective will then flow approximately as diagrammed below:

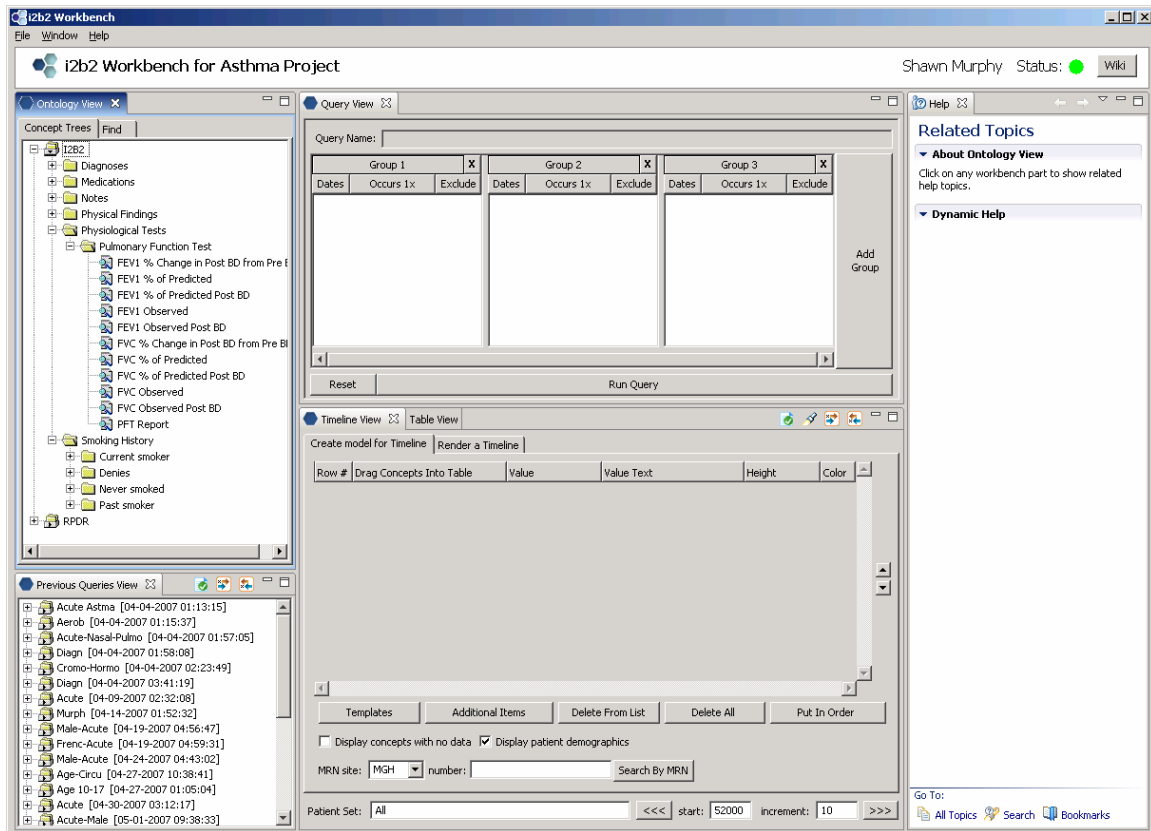


The **Browse Concepts** interaction allows the user to search and find concepts of interest. This interaction is the starting point for many other interactions. For example, a user must browse and choose concepts to define the criteria in a Patient Set Query and also to select the data to be returned in a Patient Data Query.

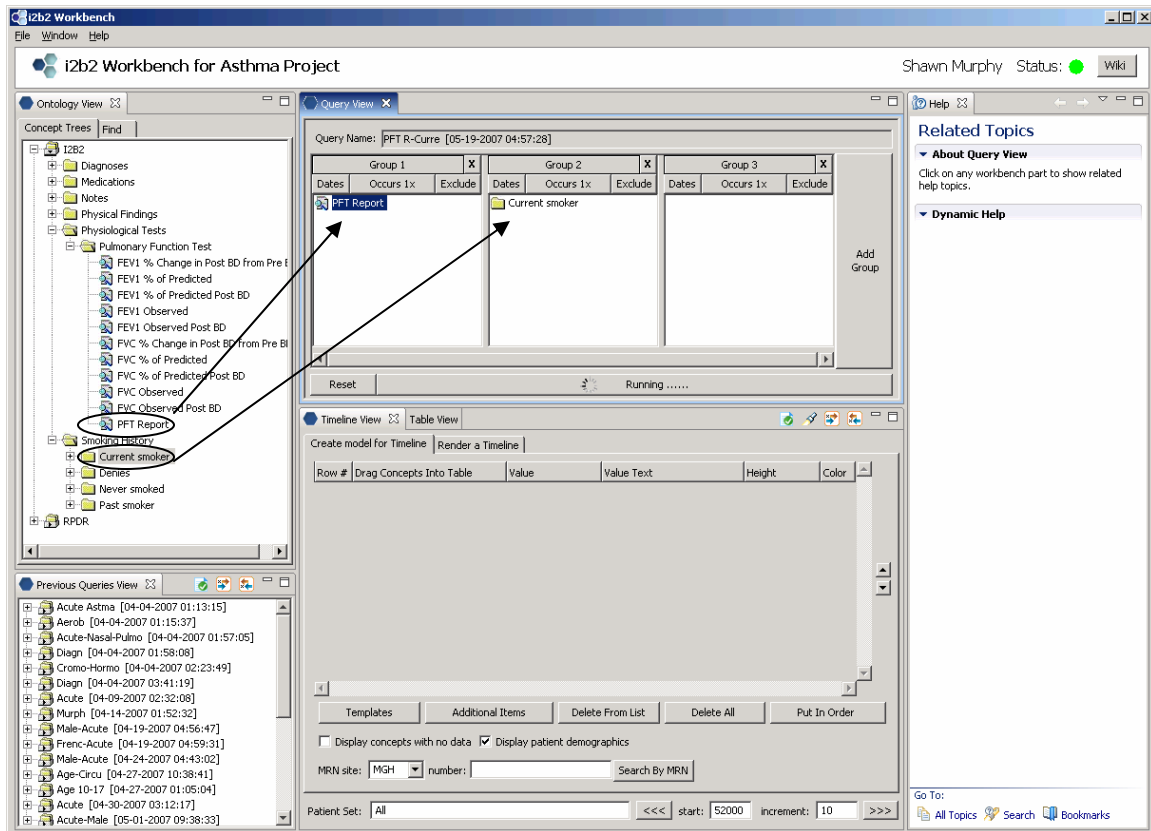
A **Patient Set Query** interaction allows the user to create criteria that will define a set of patients. The user chooses concepts from the Browse Concepts interaction and a set of patients will be created that fit the criteria. The patient set can then be used in other interactions such as in a Patient Data Query.

The **Patient Data Query** interaction allows the user to select the type of data to be returned on a set of patients. The user chooses concepts from the Browse Concepts interaction and chooses a Patient Set, which may have previously been created with a Patient Set Query. Patient Data is returned for any patient in the Patient Set that has any of the chosen concepts.

First one will use the ontology plug-in to browse concepts in the ontology cell:



Next one drags the “Current smoker” and “PFT Report” to the Query View to create a query. Information retrieved from the Ontology Cell is dragged to the Query view plug-in in an XML message that communicates with the Data Repository Cell when the “Run Query” button is pushed, and will initiate a query on the Data Repository cell that will find those patient who are both current smokers AND have had a PFT report.

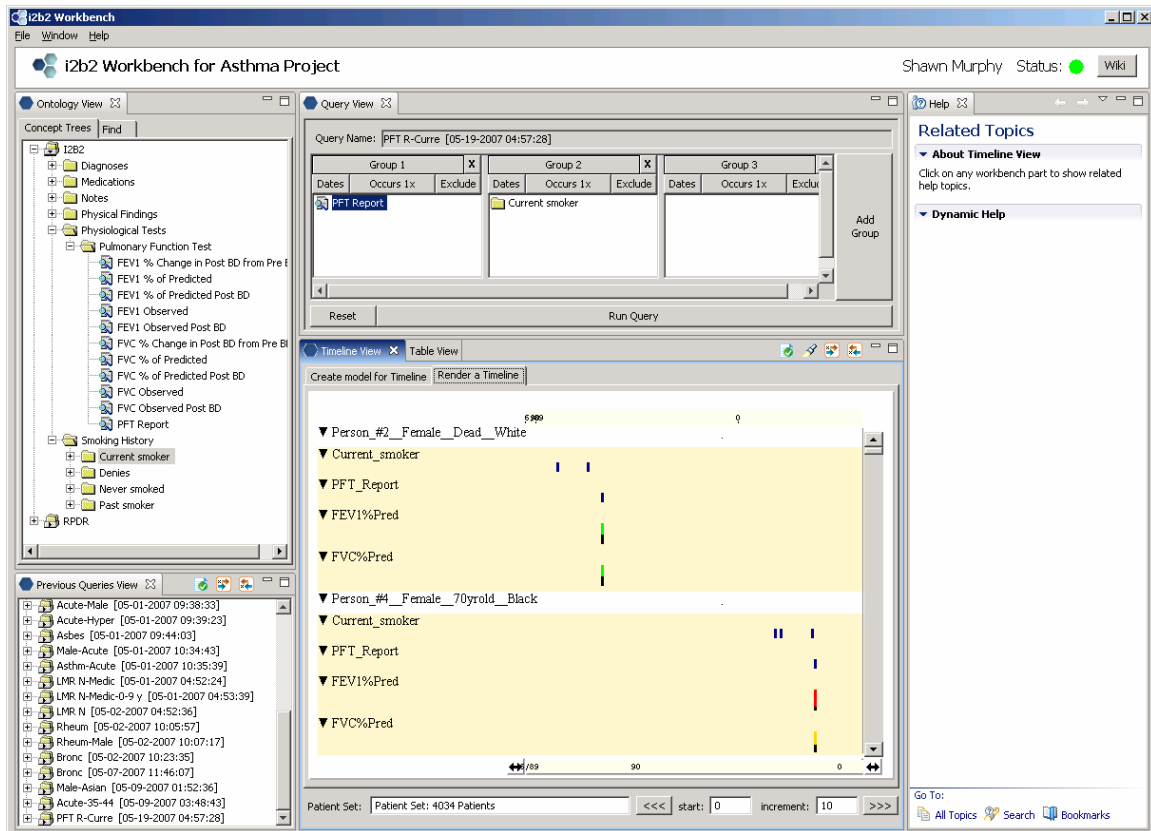


After the query is run, the results are available in the “Previous Queries View”. The Ontology View is again the source of information that travels during the drag-and-drop as an XML message to the “Timeline View”, as illustrated below. The patient set of 4034 patient will be used to determine what patients will show in the timeline. In the case below, we are going to show the forced expiratory volume in the first second (FEV1, which can be very poor compared to normal controls for those who have asthma and is a good marker for severity) and the forced expiratory vital capacity (FVC, also often poor in asthmatics).

The screenshot displays the i2b2 Workbench interface for an Asthma project. On the left, the 'Concept Trees' panel shows a hierarchy of concepts under 'I2B2', including 'Pulmonary Function Test' and 'Smoking History'. The 'Query View' panel shows a query named 'PFT R-Curre' with three groups of concepts: Group 1 (PFT Report), Group 2 (Current smoker), and Group 3 (empty). The 'Timeline View' panel shows a table of patient observations with columns for Row#, Value, Value Text, Height, and Color. The table contains 12 rows of data, including 'Current smoker', 'PFT Report', and 'FEV1 % of Predicted' observations. The 'Previous Queries View' panel shows a list of previous queries, including 'Acute-Male', 'Acute-Hyper', 'Asbes', 'Male-Acute', 'Asthm-Acute', 'LMR N-Medic', 'LMR N-Medic-0-9 y', 'LMR N', 'Rheum', 'Rheum-Male', 'Bronc', 'Male-Asian', 'Acute-35-44', and 'PFT R-Curre'.

Row#	Value	Value Text	Height	Color
1	Current smoker	N/A	N/A	Medium
2	PFT Report	N/A	N/A	Medium
3	FEV1 % of Predicted	NVAL_NUM	> 80	Medium
3	FEV1 % of Predicted	NVAL_NUM	between 55 and 80	Low
3	FEV1 % of Predicted	NVAL_NUM	< 55	Very Low
4	FVC % of Predicted	NVAL_NUM	> 80	Medium
4	FVC % of Predicted	NVAL_NUM	between 55 and 80	Low
4	FVC % of Predicted	NVAL_NUM	< 55	Very Low

The timeline then shows each patient (de-identified by a pseudo-number only) and the time course of their pulmonary function test (PFT) observations, as well as when it was recorded in time that they were a smoker.



The PFT report can be dragged in an XML message to the plug-in that sends the report to the PFT parsing cell. Indeed, it was only through an automated workflow that uses the PFT cell that the numeric PFT results are available at all, as the only raw data available from the PFT lab originally are the text reports. The way to manually perform this action actually involves first double clicking the bar which represents the report. The report is then displayed (see later slide), and the report is grabbed from the viewer and dragged to the PFT plug-in.

The screenshot shows the i2b2 Workbench for Asthma Project. The main window is titled "i2b2 Workbench for Asthma Project" and includes a user profile for Shawn Murphy. The interface is organized into several panes:

- Ontology View:** A hierarchical tree on the left showing concepts such as "Diagnoses", "Medications", "Notes", "Physical Findings", "Physiological Tests", "Smoking History", and "RPDR".
- Query View:** A central pane showing a query named "PFT R-Curre" with three groups of concepts. Group 1 includes "PFT Report", Group 2 includes "Current smoker", and Group 3 is empty.
- Timeline View:** A central pane showing a vertical timeline for two patients: "Person #2_Female_Dead_White" and "Person #4_Female_70yroid_Black". The timeline displays various events represented by colored bars, including "Current_smoker", "PFT_Report", "FEV1%Pred", and "FVC%Pred". A black arrow points from the "PFT_Report" event to the "PFT View" report.
- PFT View:** A right-hand pane displaying a detailed text report for a patient named Jane Doe. The report includes patient information (PT: DOE, JANE; PT#: 12345678; AGE: TECH) and spirometry results (FVC, FEV1, FEV1/FVC, FEV25-75%, FEVmax, TET).

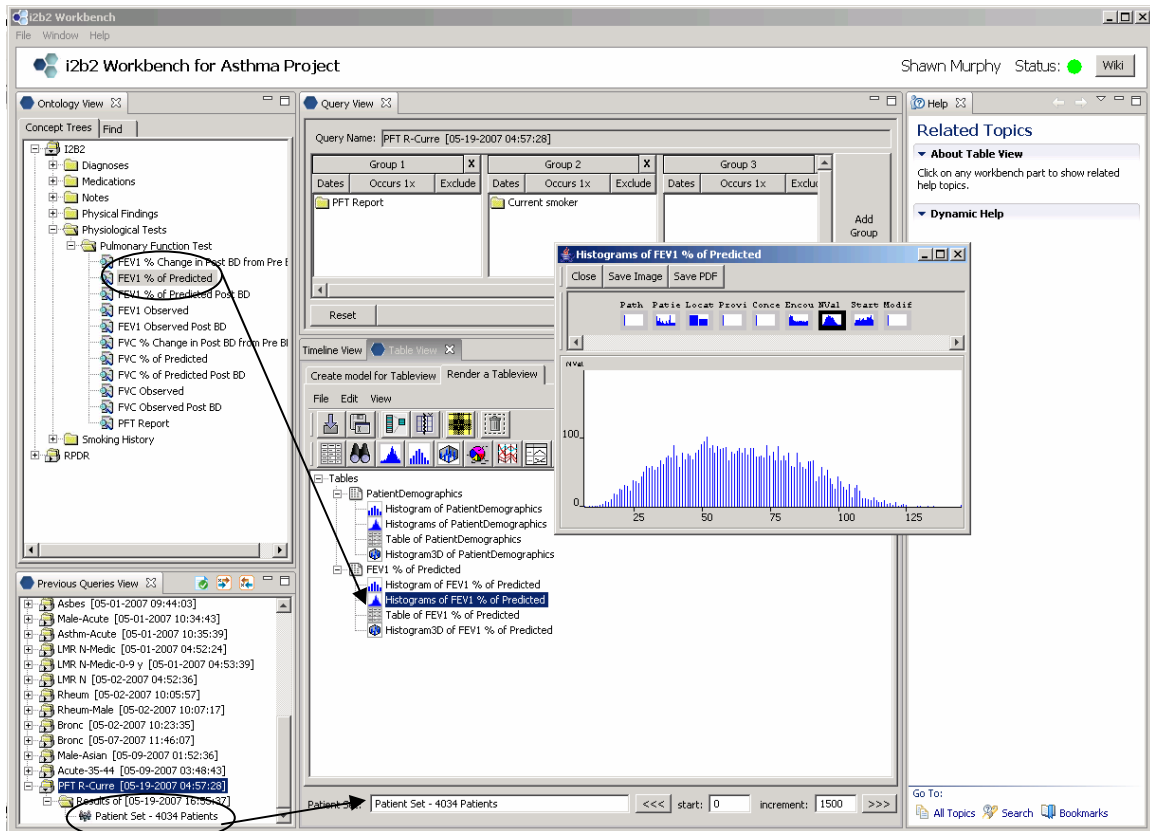
Illustrated below is the result of sending the text report to the PFT parsing cell manually by pushing the "Get Pulmonary Data" button.

The screenshot displays the i2b2 Workbench for Asthma Project. The main window shows a query named "PFT R-Curre" with three data groups. The "Timeline View" shows data points for two patients: "Person #2_Female_Dead_White" and "Person #4_Female_70yroid_Black". The "PFT View" panel on the right displays the following data:

Name	Value/Units
Height	63.0 inch
Weight	105.0 pound
FEV1 Observed	1.51 liter
FEV1 % of Pr...	76.0 percent
FVC Observed	2.12 liter
FVC % of Pre...	82.0 percent

Help is available for each plug-in as one activated the plug-in and works with it, as illustrated below in the panel to the right:

Other plug-ins can be added fairly easily to the i2b2 workbench. Illustrated below is the “TableView” application available at <http://ccgb.umn.edu/software/java/apps/TableView/>. It is a software package written by Jim Johnson, available under an LGPL open source software license at the Center for Biomedical Research Informatics, University of Minnesota. It is able to perform many software functions, but illustrated is a histogram of a random set of the FEV1’s of a random set of 1000 patients from the previous patient set.

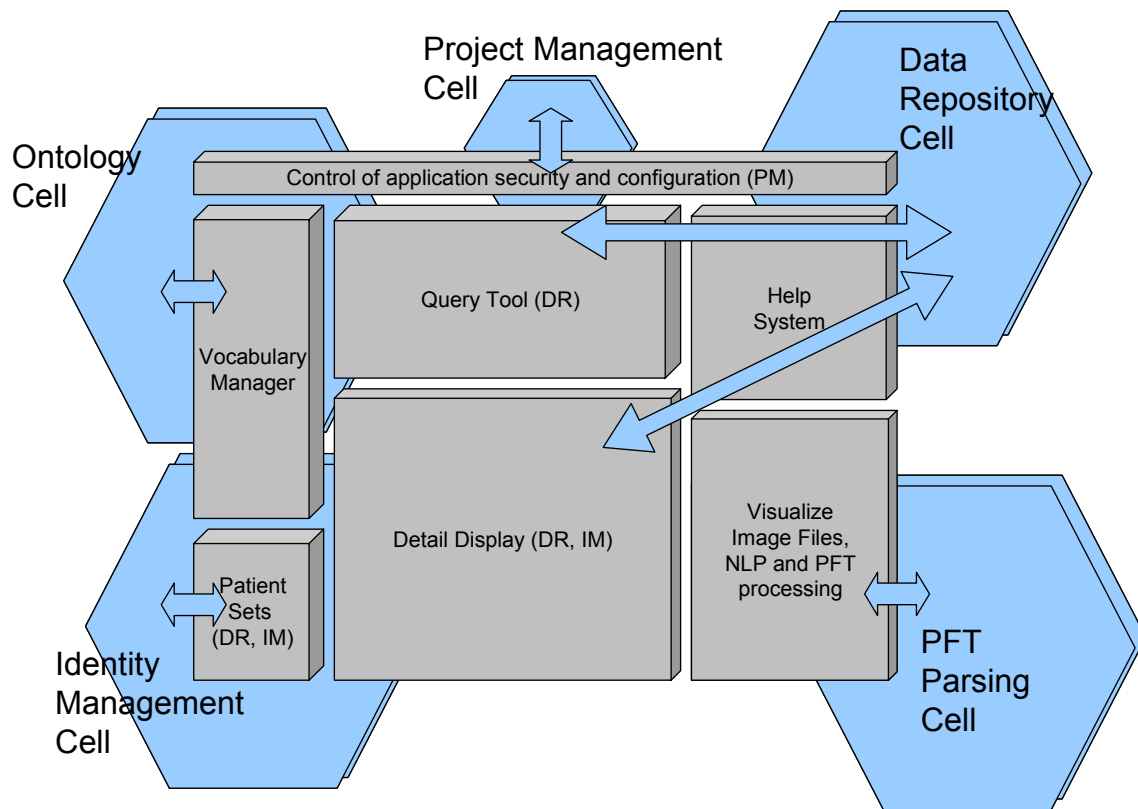


Recall that the i2b2 Hive and Workbench is written to accommodate three strata of users:

- *Clinical investigators who want to use the software in as “shrink-wrapped” a way as possible,*
- *Bioinformatics scientists who want the ability to customize the flow of data and interactions, and*
- *Biocomputational software developers who want to develop new software capabilities that can be integrated easily into the computing environment*

The user interface in the i2b2 Workbench allows an interaction that appears natural and straightforward to the first strata of users above, while accommodating the two other strata of users as well. We accommodate the Biocomputational software developers such that tools that they develop, like the TableView application, can easily be incorporated into the i2b2 architecture. The flow of interactions is also accommodated by this architecture. What is actually supported by the workbench is aggregating the cells of the hive at an explicit level, illustrated in the diagram that follows. The cells of the hive also

communicate without user interaction, with synchronization of data, authorization procedures, and so forth which will be described below.



The framework of software modules is called the i2b2 Hive, and is centered on two concepts. The first concept is the existence of services provided by applications that are “wrapped” into functional units called “cells”, such that their functionalities are exposed as messages that travel to and from the various cells of the hive. The second concept is that of persistent data storage optimized to serve as a repository for medical data. The data that is stored in the i2b2 Hive is organized so that issues of data ownership, data privacy, and disparate forms of medical data are accounted for in the way the data is represented. The i2b2 Cell is the basic building block of an i2b2 environment, and encapsulates business logic as well as access to data objects within the cell behind standard Web service interfaces. The communication within the i2b2 Hive occurs through these web services provided by the i2b2 Cells. A Web service describes a standardized way of integrating Web-based applications using the XML, SOAP, REST, WSDL and UDDI open standards [1] over an Internet protocol backbone.

XML is used to tag the data, SOAP or REST is used to transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available. Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide a visual user interface. Web services instead share business logic, data and processes through a programmatic interface across a network. Web services allow different applications from different sources to communicate with each other, and because all communication is in XML, Web services are not tied to any one operating system or programming language. For example, Java can talk with Perl, and Windows applications can talk with UNIX applications.

The engineering of the software in the i2b2 Hive follows a general principle of distributed responsibility that is embodied in the concept of the Hive. The Hive is composed and emerges from a collection of functional Cells. Each Cell exists as a functional unit independently, but also has a well defined set of interfaces to interact with other Cells within a Hive. The entire functionality of the Clinical Research Chart is achieved only with the combination and interaction of many Cells, but these design principles are important for two reasons. First, the Cells can be developed in a decoupled fashion. This allows functionality to be developed concurrently, with minimal interaction of the various groups involved in their construction. Ultimately, this is a feature that will enable development to scale to a nationwide level. Second, the services provided by Cells from various projects must follow a standard for making a public interface visible to other developers, and this can be leveraged to create new Cells with more integrative functionality. This characteristic will enable functionality and data management to scale up without increasing demands for coordinated development teams.

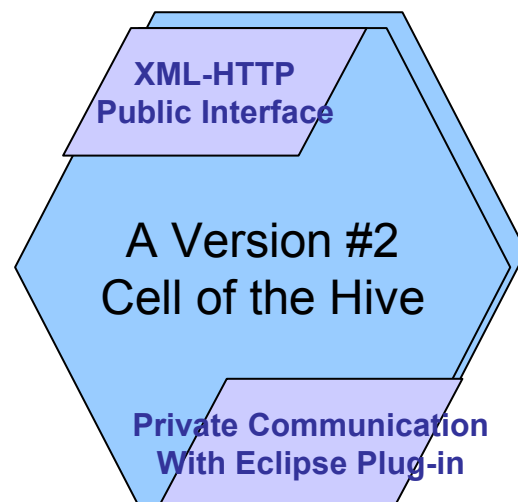
The building of the Hive is occurring through the following staged process:

- 1) Establishment of the core Cells of the Hive that must be present to achieve the absolute minimum functionality of the Hive.
- 2) Establish the Cells of the Hive with the DBP's that will be added in to support use-cases of functional Cells
- 3) Design the interfaces that each core Cell must provide
- 4) Establish the requirements for several phases (versions) of each Cell, such that each version of the Hive will be a functional, cohesive entity
- 5) Engineer each Cell
- 6) Provide DBP's with a working version of the Hive for testing
- 7) Evaluate the use of the Hive

There are five core Cells that currently compose the backbone of the Hive as shown by the dark blue cells in the figure of the Hive above. Without these Cells, the functionality of the Hive as an autonomous entity would be limited, although conceivably a different product could serve in a local environment to substitute for any one of them. If one chose to construct one of the core Cells from the local environment, it would need to provide interfaces that comply with the specification for that specific Cell.

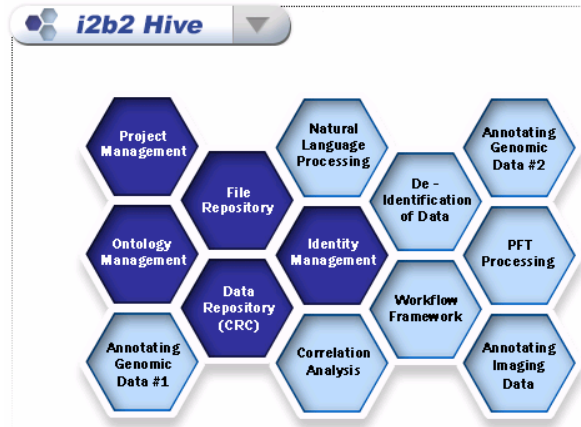
A Cell that functions in Version 2 of the Hive specification must support a public XML-HTTP interface that allows network-based communication with other Cells, and can optionally support a managed user interface hosted in the Eclipse i2b2 Workbench described above that allows visible interaction with users (although all the examples above use the defined public XML-HTTP interfaces for their functionality). These interfaces to the Cell are the primary manner in which data can flow into or out of a Cell. All of the functionality of the Cell should be defined through these interfaces. In this way, the Cell

corresponds to a programmer's concept of an "Object", and the XML-HTTP interfaces correspond to the Cells public methods. Communication from the cell to the Eclipse plug-in must support user authentication through communicating with the project management plug-in available on the workbench or the Project Management cell directly. Otherwise, the visual interaction can be any program or client to a cell, and the Eclipse workbench



fully supports routine displays of HTML-HTTP pages and interactions. The ability to interact directly with any plug-in on the eclipse workbench adds some of the functionality of which is commonly provided to a developer through an integrated development platform. These views are important when one wants to check the internal operation of a Cell, status information, or to provide quality assurance of the operations. However, this form of interaction will not be able to be reused by other cells, unlike the XML-HTTP interfaces correspond to the Cells public methods.

The XML schema that defines i2b2 Hive compatibility for messaging has been defined and a public version has been available on the i2b2 web site since December 2006. The schema consists of a header that manages security, routing,



session, and other management information, and a payload section that has a variety of schemas as different clinical data objects are represented. The payload schema that is in use reflects a generic model of observations and events that allows integration of various phenotypic and genotypic data types.


What follows is a description of individual cells that have been developed. For those whose complete Messaging specification is available on the web, the URL is posted. Three are fully available, and 2 others will be coming this summer. The messaging header and payload scheme are available and were mentioned above.

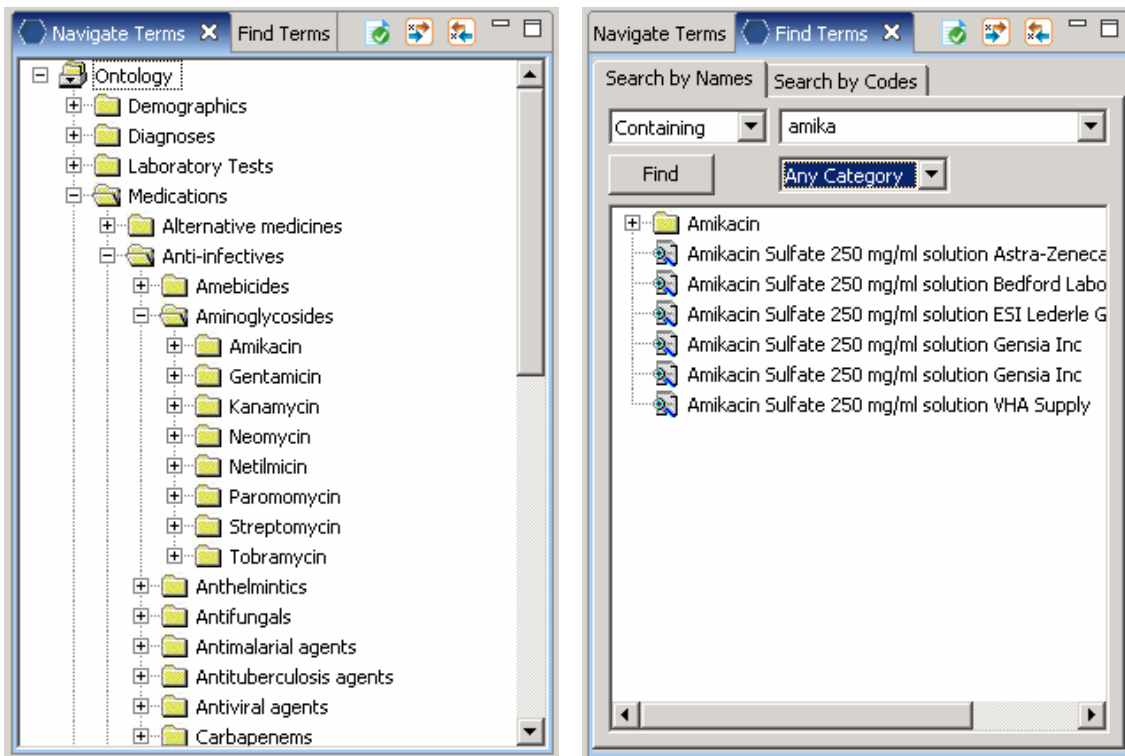
Ontology Management Cell

Full Messaging 0.9 specification available at <https://www.i2b2.org/resrcs/pdf/OntologyMgmtMessage.pdf>

Maintaining relationships between terms in multiple vocabularies is a vital activity for any organization attempting to support the integration of data coming from multiple sources. In a clinical research setting, in fact, this activity becomes even more significant. Combined laboratory and clinical data from diverse systems, with semantically related codes, terms and concepts, can help transform genomic knowledge into the practice of healthcare. Unless these vocabulary relationships are defined, however, the potential for considerable insight is lost. Maintaining these relationships, then, becomes a priority.

Vocabulary mapping is a process of specifying and maintaining relationships between terms in multiple vocabularies. One vocabulary, designated as master becomes the classification scheme for the other subsidiary vocabularies. In order to use a system like this, there must be some way of maintaining freedom and control over the master vocabulary, while still maintaining the integrity of the mappings to each source. One tool, Protégé (Musen, 1998), is particularly well suited to maintaining concepts and relationships within a vocabulary. Its support for the creation and maintenance of ontologies, including taxonomies,

classifications and their associated vocabulary make it an ideal tool for managing a distinct vocabulary. The master vocabulary, in turn, provides the base hierarchy upon which to map other vocabulary sources. A new mapping tool developed as an i2b2 Cell will integrate with Protégé and facilitate mappings between the base hierarchy and source terms. However, we have delayed its development to give the National Center for Biomedical Ontologies  time to develop a plan to incorporate some of our needs with regard to the development of Protégé tools. The presentation of the OM Cell in the i2b2 Workbench is in a tree navigating plug-in and a find plug-in that allow the ontologies of the i2b2 Hive to be effectively navigated, as described in the examples above.

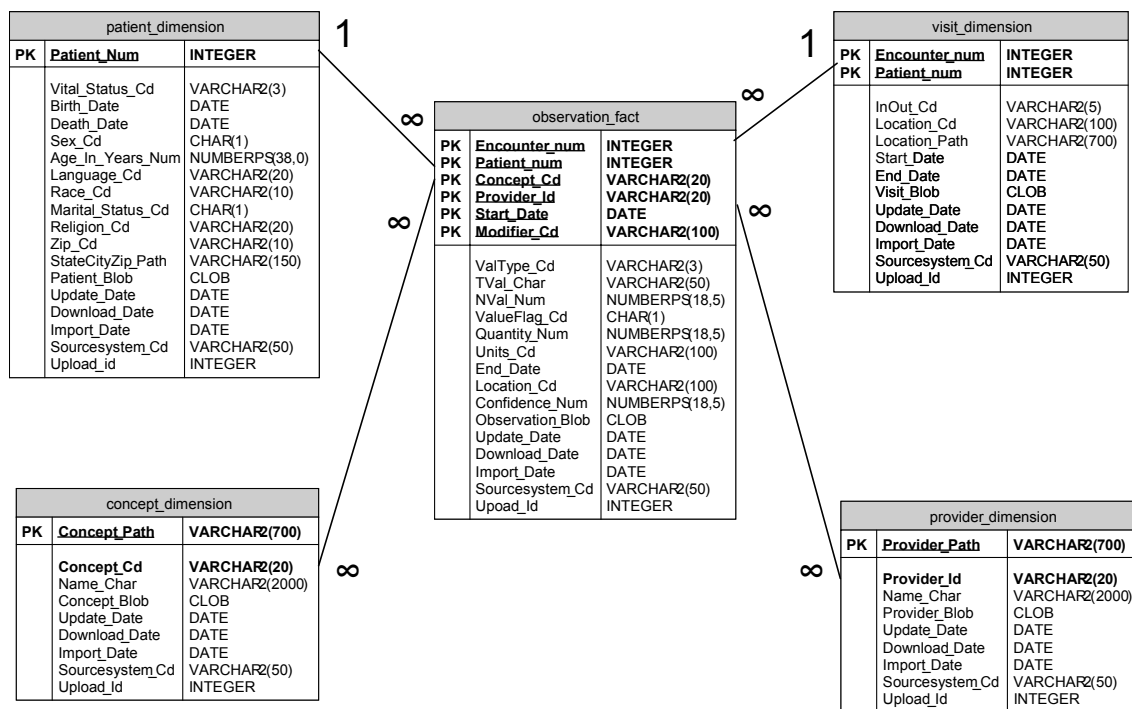


Data Repository Cell

<https://www.i2b2.org/resrcs/pdf/DataRepositoryMessage.pdf>

The data repository piece is designed with several requirements. It must be able to hold healthcare information from many different venues, and allow it to be queried rapidly even if there are hundreds of millions of rows. It must be easily combined with other projects repositories to form large unified repositories. Finally, it must allow objects to be stored that are present in the genomic data.

The data repository Cell is a database based on the Star Schema structure, a design proposed initially by Ralph Kimball in the 1980s. It is named this because of the appearance of the final database schema diagram that looks like a star (see figure). Once a database grows to over 10 million items, the advantages of a star schema can start to take hold. The first consideration is the speed and integrity of the queries. When one exceeds 0.5 billion rows in a database, it becomes important to have the data expressed in very large indexes. Very large indexes are only possible with very large tables. If one has several hundred or thousand tables in a database (easily attained in large transaction systems), one will have at least one index on each table resulting in several hundred or thousand small indexes. Joins between 100-1000 indexes for each query will result in slow performance (hours), while joins between 3-4 indexes, even representing 100's of millions of rows, will be fast (seconds). Furthermore, the integrity of queries in a transactional database is also compromised because queries can often be answered through several paths in a circular manner.

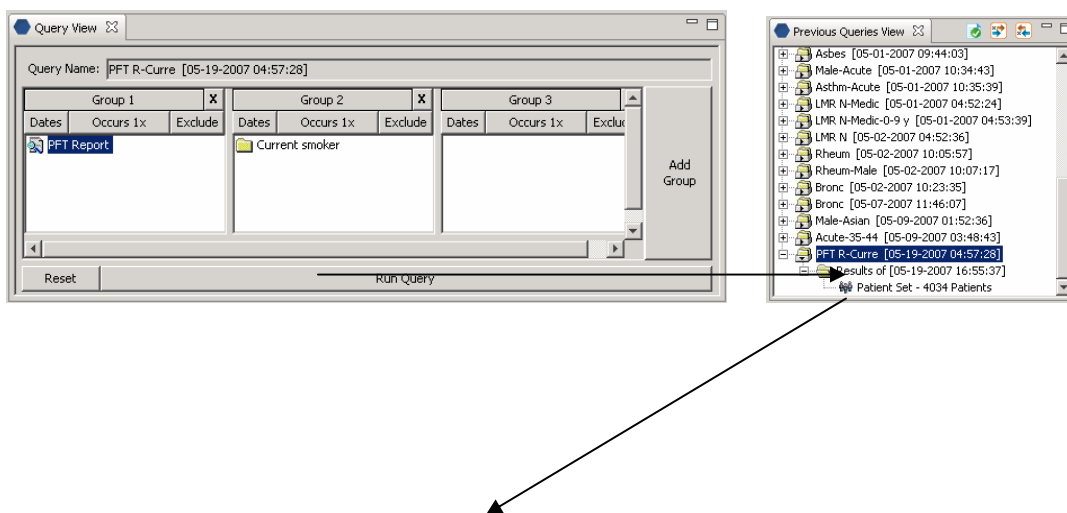


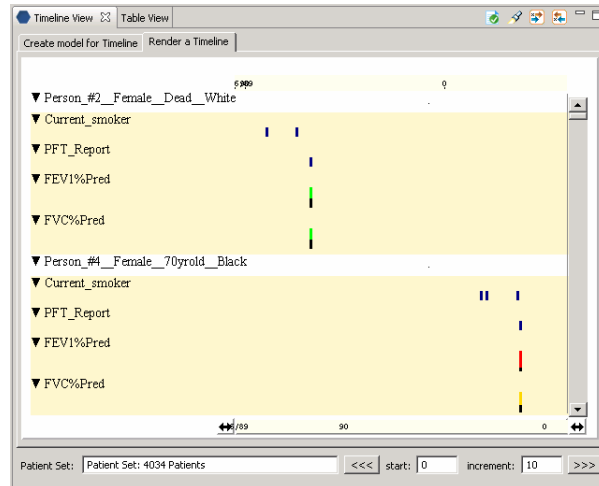
The second consideration is the need for a large analytic database to constantly absorb new data. The database schema does not change as new data sources are added. New data will result in additional rows added to the fact, patient, and visit tables. New concepts and observers will result in new rows added to the concept and provider tables. But new columns and tables do not need to be

added for each new data source. This is very useful in large projects where there are many tools depending upon a specific database schema. A strategy where the database grows by adding rows for new data rather than adding new tables and columns allows tools developed to work with one kind of data to also work with a new source of data.

The third advantage of the star schema is the ability to manage the metadata of a large analytic database. Metadata is used to perform queries, and if it is incorrect a query will be profoundly affected. For example, if one wanted to find all the patients with diabetes, but left out one of the codes used to represent diabetes in a database, none of those orphaned patients would be counted. Almost all the metadata in this healthcare data warehouse is present in two tables, the concept table and the providers table. Additionally, a small amount is needed to describe the codes in the patient and visit tables. The detection of orphaned concepts is easily achieved in the star schema by joining the fact table to the concept and provider tables and reporting those fact table concepts and providers left out by the join.

I2b2 Workbench plug-ins of the data repository cell allow interactions with the database, and creation of patient and event sets. The plug-ins include one for performing queries of the database, one for interacting and obtaining patient and event sets from previous queries, and displays of detailed data obtained from these patient and event sets. The query interface has been described in detail previously (Murphy 2000, 2003), and is essentially a way of building queries using concepts represented in the ontology tables.





Observations in the central table of the Data Repository Cell are organized one row per observation, which works well for representing data in a general fashion, but can be difficult for a casual user to understand by viewing the table. But the observations are all established with a time stamp, so one of the most intuitive displays of this data is along a time line, The timeline plug-in aims to achieve the following principle objectives:

- 1) Allow relatively naïve clinical researchers the opportunity to survey the data to understand what data exists within the CRC.
- 2) Allow users to have a way to check data integrity for obvious anomalies.
- 3) Allow users to compare to data coming from various sources.
- 4) Allow data on single patients to be concisely displayed.
- 5) Allow the exploration of various levels of detail in the term hierarchies.
- 6) Show time-oriented averages (trends) within the data using sentinel events to synchronize the starting points.
- 7) Allow users to perform time-oriented queries.

After the Timeline has been generated, a graphical representation of the data is presented. There are scrollbars on the right to see all the time lines, and time itself can be expanded in more detail along the bottom. The top contains the date that represents an even distribution of months and years. If the patient, listed on the left side, contains an entry for a specific concept, than a color mark will be presented with the concept start date.

Project Management Cell

<https://www.i2b2.org/resrcs/pdf/ProjectMgmtMessage.pdf> (pending)

The Project Manager cell contains the functionality needed to set up an i2b2 project with users and roles. It keeps the references to the other cells in the hive, and takes care of various functions with respect to aggregating the cells into one project. In this regard, it maintains the security infrastructure in regards to authentication and authorization, for both this cell and the other cells in a project hive. It provides the central url for a project, thus providing a unified web interface based on a portlet specification, where portlets owned by the various cells of the hive will be aggregated into a single browser-based container.

The project management cell is set up within the GridSphere portal framework (<http://www.gridisphere.org/>). The GridSphere portal framework provides an open-source portlet based Web portal. GridSphere enables developers to quickly develop and package third-party portlet web applications that can be run and administered within the GridSphere portlet container. Here you will find the GridSphere portal framework available for download and documentation related to the installation and development of portlets using GridSphere.

The GridSphere Portal Framework offers the following features:

- Portlet API implementation fully [100% JSR 168 compliant](#)
- Support for the easy development and integration of new portlet applications
- Higher-level model for building complex portlets using visual beans and the GridSphere User Interface (UI) tag library
- Flexible XML based portal presentation description can be easily modified to create customized portal layouts
- Built-in support for Role Based Access Control (RBAC)
- Sophisticated portlet service model that can encapsulate reusable portlet logic into services that may be shared between many portlets
- Persistence of data provided using [Hibernate](#) JDO/OQL for database support
- Support for portlet-izing Struts applications using the [Portals Struts Bridge](#)
- GridSphere core portlets:
 - Login, Logout, Locale settings
 - Profile personalization and Layout customization
 - Administration portlets for creation of users, groups, portlet management and portal layout customization

The open source project is the same as that used by the Biomedical Informatics Research Network, and we are essentially reusing their solutions both for our project management and the File repository which uses the Storage Resource Broker of the San Diego Supercomputer Center (SRB, <http://www.sdsc.edu/srb/>). Use of these two infrastructure pieces will enable a Grid infrastructure for i2b2 in the future.

In the i2b2 Workbench, the links to the Project Management Cell are nearly invisible, but occur through the “Banner” plug-in visually located along the top. Error logging and single sign on among plug-ins also occurs through this plug-in shown below:



Pulmonary Function Test (parsing) Cell

<https://www.i2b2.org/resrcs/pdf/PFTProcessingMessage.pdf>

The Pulmonary Function Test (PFT) Processing Cell parses a pulmonary function report and extracts embedded test values. The report provided must be in a specific format. This cell works as part of the hive for this specific localized purpose. Communication with the PFT Cell, like all cells in the i2b2 hive, occurs through web services. The XML messages must conform to the i2b2 messaging standard which allows cell-specific XML within the <message_body> tag. The rest of this document describes PFT services and XML formats which are specific to the PFT Cell and it illustrates how these XML messages are used to accomplish a set of interactions that correspond to typical PFT use cases.

The PFT processing cell has been provided to the community as an example i2b2 cell. Source code that describes how to create a cell and an i2b2 Workbench plug-in to communicate using i2b2 XML messaging is available at <https://www.i2b2.com>, and has been used by both the NLP Cell and the de-id Cell as model code on how to build a cell. Complete documentation is available with the download.

The PFT processing cell also describes how to link a Python-language base into an i2b2 Cell. This cell was used to parse the PFT reports in the workflow described below and enter them into the clinical research chart.

PFT View X

Report Results Request XML Response XML

Date: 01/23/05

PT: DOE, JANE
 PT#: 12345678 AGE: 67 SEX: F HT: 63.0 in
 PHYSICIAN: SMITH TECH: ABC
 DIAGNOSIS: DYSPNEA
 SMK HX: NEVER

Spirometry		Predicted	Pre-Drug*	Actual	%Pred	Ac
FVC	(L)	2.58		2.12	82	
FEV1	(L)	1.97		1.51	76	
FEV1/FVC	(%)	77		71	92	
FEF25-75%	(L/S)	1.79		1.09	61	
FEFmax	(L/S)	5.16		2.70	52	
TET	(SEC)			9.66		

TREND REPORT

DATE	TIME	FVC (L) (PRE)	FEV1 (L) (PRE)	FEV1/FVC (%) (PRE)
01/23/05	08:25:40	2.12	1.51	71.36

Get Pulmonary Data Clear Text

BWH PFT Report Load Sample

PFT View X

Report Results Request XML Respons

Name	Value/Units
Height	63.0 inch
Weight	105.0 pound
FEV1 Observed	1.51 liter
FEV1 % of Pr...	76.0 percent
FVC Observed	2.12 liter
FVC % of Pre...	82.0 percent

I2b2 Software License ("Software License")
Version 1.0

This Software License covers downloads from the i2b2 project ("i2b2") maintained by The Brigham and Women's Hospital, Inc. ("BWH").

Your downloading, copying, modifying, displaying, distributing or use of any software and/or data from i2b2 (collectively, the "Software") constitutes acceptance of all of the terms and conditions of this Software License. If you do not agree to such terms and conditions, you have no right to download, copy, modify, display, distribute or use the Software.

1. As used in this Software License, "you" means the individual downloading and/or using, reproducing, modifying, displaying and/or distributing the Software and the institution or entity which employs or is otherwise affiliated with such individual in connection therewith. The BWH hereby grants you, with right to sublicense, with respect to BWH's rights in the software, and data, if any, which is the subject of this Software License (collectively, the "Software"), a royalty-free, non-exclusive license to use, reproduce, make derivative works of, display and distribute the Software, provided that:

(a) you accept and adhere to all of the terms and conditions of this Software License;

(b) in connection with any copy of or sublicense of all or any portion of the Software, all of the terms and conditions in this Software License shall appear in and shall apply to such copy and such sublicense, including without limitation all source and executable forms and on any user documentation, prefaced with the following words: "All or portions of this licensed product (such portions are the "Software") have been obtained under license from The Brigham and Women's Hospital, Inc. and are subject to the following terms and conditions:"

(c) you preserve and maintain all applicable attributions, copyright notices and licenses included in or applicable to the Software;

(d) modified versions of the Software must be clearly identified and marked as such, and must not be misrepresented as being the original Software; and

(e) you consider making, but are under no obligation to make, the source code of any of your modifications to the Software freely available to others on an open source basis.

2. The license granted under this Software License includes without limitation the right to (i) incorporate the Software into proprietary programs (subject to any restrictions applicable to such programs), (ii) add your own copyright statement to your modifications of the Software, and (iii) provide additional or different license terms and conditions in your sublicenses of modifications of the Software; provided that in each case your use, reproduction or distribution of such modifications otherwise complies with the conditions stated in this Software License.

3. This Software License does not grant any rights with respect to third party software, except those rights that BWH has been authorized by a third party to grant to you, and accordingly you are solely responsible for (i) obtaining any permissions from third parties that you need to use, reproduce, make derivative works of, display and distribute the Software, and (ii) informing your sublicensees, including without limitation your end-users, of their obligations to secure any such required permissions.

4. The Software has been designed for research purposes only and has not been reviewed or approved by the Food and Drug Administration or by any other agency. YOU ACKNOWLEDGE AND AGREE THAT CLINICAL APPLICATIONS ARE NEITHER RECOMMENDED NOR ADVISED. Any commercialization of the Software is at the sole risk of the party or parties engaged in such commercialization. You further agree to use, reproduce, make derivative works of, display and distribute the Software in compliance with all applicable governmental laws, regulations and orders, including without limitation those relating to export and import control.

5. The Software is provided "AS IS" and neither BWH nor any contributor to the software (each a "Contributor") shall have any obligation to provide maintenance, support, updates, enhancements or modifications thereto. BWH AND ALL CONTRIBUTORS SPECIFICALLY DISCLAIM ALL EXPRESS AND IMPLIED WARRANTIES OF ANY KIND INCLUDING, BUT NOT LIMITED TO, ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL BWH OR ANY CONTRIBUTOR BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, EXEMPLARY OR CONSEQUENTIAL DAMAGES

HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY ARISING IN ANY WAY RELATED TO THE SOFTWARE, EVEN IF BWH OR ANY CONTRIBUTOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. TO THE MAXIMUM EXTENT NOT PROHIBITED BY LAW OR REGULATION, YOU FURTHER ASSUME ALL LIABILITY FOR YOUR USE, REPRODUCTION, MAKING OF DERIVATIVE WORKS, DISPLAY, LICENSE OR DISTRIBUTION OF THE SOFTWARE AND AGREE TO INDEMNIFY AND HOLD HARMLESS BWH AND ALL CONTRIBUTORS FROM AND AGAINST ANY AND ALL CLAIMS, SUITS, ACTIONS, DEMANDS AND JUDGMENTS ARISING THEREFROM.

6. None of the names, logos or trademarks of BWH or any of BWH's affiliates or any of the Contributors, or any funding agency, may be used to endorse or promote products produced in whole or in part by operation of the Software or derived from or based on the Software without specific prior written permission from the applicable party.

7. Any use, reproduction or distribution of the Software which is not in accordance with this Software License shall automatically revoke all rights granted to you under this Software License and render Paragraphs 1 and 2 of this Software License null and void.

8. This Software License does not grant any rights in or to any intellectual property owned by BWH or any Contributor except those rights expressly granted hereunder.