**A Database to Support Development**

**And Evaluation of Intelligent Patient Monitoring**

by

Christine Lieu

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

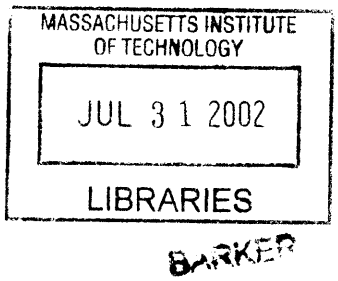
Bachelor of Science in Electrical [Computer] Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 24, 2002








Author_____
Department of Electrical Engineering and Computer Science
May 24, 2002

Certified
by_____
Roger G. Mark, M.D., Ph.D.
Distinguished Professor in Health Sciences and Technology, HST
Professor of Electrical and Bioengineering,
Department of Electrical Engineering and Computer Science, MIT
Thesis Supervisor

Accepted
by_____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

# A Database to Support Development

# And Evaluation of Intelligent Intensive Care Monitoring

By Christine Lieu

Submitted to the Department of Electrical Engineering and Computer Science
May 24, 2002

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
And Master of Engineering in Electrical Engineering and Computer Science

## Abstract

Advances in medical observation equipment have allowed hospital caregivers to collect a wealth of information about a patient's condition. Automated data collection systems make it easy to store multiple clinical notes, lab results, and physiological waveforms electronically. Organizing this data and making it useful to researchers is a complex problem that requires a solution that is easily accessible, intuitive to use, and versatile. The major aim of this project is to develop a framework for the automated acquisition of patient data from clinical information systems and patient monitors, and the subsequent storage, indexing, and presentation of patient records using web and relational database technologies.

Thesis Supervisor: Roger G. Mark, M.D., Ph.D.
Distinguished Professor in Health Sciences and Technology, HST
Professor of Electrical and Bioengineering,
Department of Electrical Engineering and Computer Science, MIT

# 1 Problem Statement

Researchers in intelligent patient monitoring need large amounts of data to test and validate hypotheses about patient care. The data test set must be sufficiently large to ensure that results are not the artifacts of a specific group of patients, and that hypotheses can be applied to the general patient population. One scientist notes that data collection is one of the most complex and resource-intensive stages in clinical research. He writes, "One of the difficulties facing researchers in clinical informatics has been 'getting the data in.' In particular, the costs of acquiring detailed and structured data from the clinical care process have been daunting."[1]

Advances in patient care monitoring have allowed physicians to track an Intensive Care Unit (ICU) patient's physiological state more closely and with greater accuracy. Modern computerized clinical information systems can take information from multiple data sources and store it in a central location. To be useful to researchers, the data collected from patient monitors and clinical information systems must be indexed and presented in a user-friendly interface. It should be searchable so that researchers studying a specific problem or pattern can locate relevant records.

This paper describes a utility, called MIMIC, which uses a relational database to organize real patient data and makes it widely available to the general research community. The MIMIC (Multi-parameter Intelligent Monitoring for Intensive Care) database takes advantage of the data collection systems installed at a partner hospital to collect large numbers of real patient records. This utility also organizes the data and makes it searchable on multiple dimensions and provides an intuitive user interface to view and browse data.

# 2 Background

Our partner hospital uses the CareVue Clinical Information System, developed by Philips Medical Systems[2], to store clinical data in its ICU units. Each patient's room and nursing station is equipped with a CareVue terminal where nurses may enter notes, medications information, fluid balances, and more. CareVue also stores and collects data from other sources, such as bedside monitors and results from lab tests. The data collected by CareVue is also stored in the hospital's Information Support Mart (ISM), which is built around Oracle database technology.

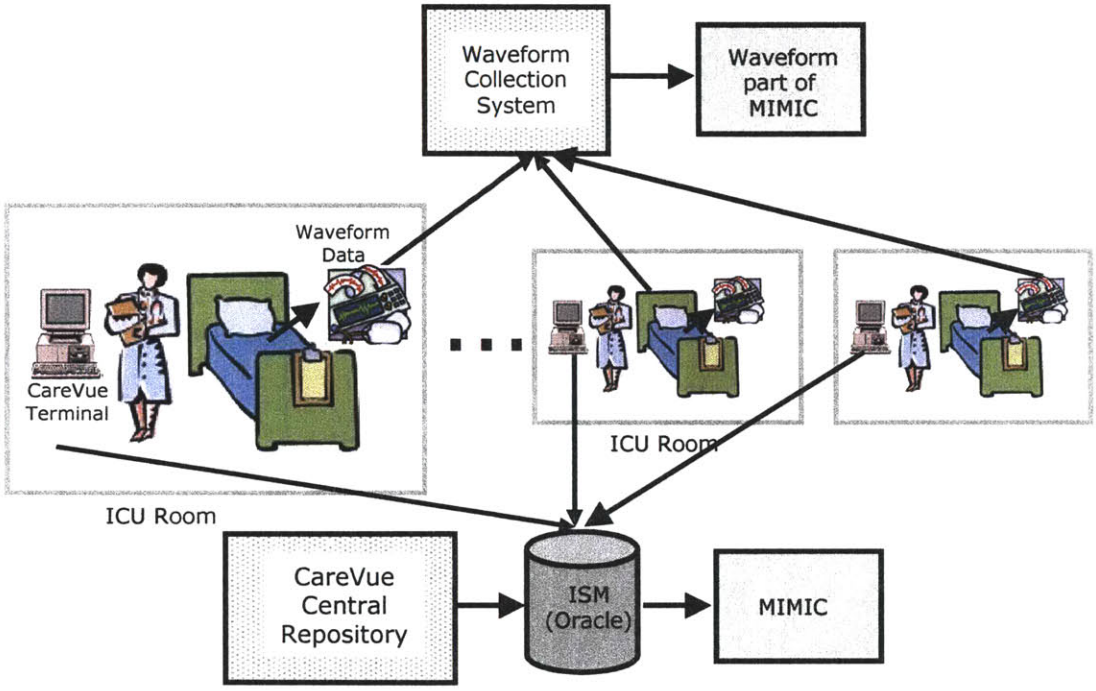



*Figure 1 Data Collection at the Partner Hospital*

The data in the ISM includes nursing notes, medication dosages and durations, fluid balances, and vital signs. Real-time waveform data, such as blood pressure, pulse oximetry, and ECG signals, are recorded in bedside monitors and collected in a separate system that is not connected to CareVue.

We have access to the data stored in the ISM and the data collected in the waveform

collection system. The CareVue system ISM system is presently connected to 42 ICU beds, which in a year can collect more than 15,000 patient days' of records.

# 3 Current Research

There are currently several alternate solutions to the problem of collecting and organizing real patient data for research. Many are targeted toward solving specific research problems, while others have less complete data or are not yet published.

## *3.1 Previous Solutions*

MIMIC1[3] is a small-scale set of records that provides some support for research in intelligent patient monitoring for ICU patients. MIMIC1 was developed by George Moody and Roger Mark and is stored in Microsoft Access and contains data from the same partner hospital's ICU. It contains only about 100 records and was entered manually from paper records, leaving room for inconsistencies and errors. MIMIC1 contains both waveforms and clinical data. It provides search capabilities and a neat user interface but is severely limited by the lack of automated data collection. MIMIC1 is the precursor to the solution described in this paper.

## *3.2 Other Databases*

Clinical Informatics is an exciting area that has made many advances in recent years. This section describes other available databases of clinical data.

One interesting solution being explored by the Medical Informatics group at Columbia University is to create an online support system based on a set of common clinical questions.

> "The aim of this study was to examine a method for creating an online support system for developers of a clinical information system (CIS) from existing documentation and question-answer exchanges (Q-A's). A question answer exchange consisted of a user's question and an expert's corresponding answer. The study was motivated by the need to improve online support systems for system developers using locally developed programs within complex information systems. "[4]

This is an innovative approach to a similar problem but lacks the flexibility of being able to answer questions not included in a pre-determined set of frequently asked questions.

The IMPROVE (Improving Control of Patient Status in Critical Care)[5] database was constructed from records from intensive care patients in Kuopio, Finland. IMPROVE stores comprehensive records, including ECG waveforms, hemodynamics, respiratory signals, lab tests, and annotations for 59 patients. For this project, a "physician observed the patient state and monitored signals online and annotated any changes in patient state or possible external causes for artifacts."[5] This provides useful information that would be missing if annotations were made afterwards, but requires extra effort by clinicians. IMPACT stores detailed records for ICU patients, but needs more records to be useful in clinical research.

There are many clinical databases that have been developed to support research in a specific problem. The HIV Information Infrastructure Program is developing a Central Research Database (CRD) to record clinical data on HIV patients. The CRD will "provide HIV researchers with access to real-time clinical data on a large number of patients, creating unprecedented research opportunities." [6] The CRD provides are similar to but are targeted toward HIV and AIDS patients.

There are several other databases similar to the CRD and the FAQ. The Penguin project[7] at Stanford is a database intended to support biomedical applications for experimental data. The University of Virginia is developing an Echocardiogram data collection system[8] to build up a database of records to support research) Duke University's Clinical Informatics group is working on TMR (The Medical Record)[9] that

provides a solution to the problem of organizing and indexing patient records, but it is in early stages of development. The MGH/MF Waveform Database[5] is a collection of ECG and hemodynamic waveforms, annotations, and relevant clinical data that was collected from the Massachusetts General Hospital. The MGH project is only a waveform database, so it does not process trends or contain other supporting data.

# 4 Solution

Our solution distinguishes itself from other currently available databases and search engines by providing real high-quality patient data that is searchable through a general-purpose search engine. It is different from other databases that were created for research in only a specific problem, or lack search capabilities or data collection facilities.

MIMIC is a utility that uses relational database technology to organize and index large amounts of multidimensional patient data. The **MIMIC RDBMS** (Relational Database Management System) is used to administer this database and create table definitions. Data can be downloaded from the hospital's ISM and entered into the MIMIC database. Once the MIMIC database is populated, the **MIMIC Application** dynamically generates HTML pages that allow any researcher with a web browser to access the records stored in MIMIC. The MIMIC application also provides search capabilities on multiple dimensions and presents records in an intuitive interface.

The use of real patient data from the partner hospital and the development of this project were approved by the Institutional Review Board (IRB), a review board for clinical research studies. Since it is not feasible or convenient to request patient consent for all data collected in the CareVue system, approval to archive the data was contingent on removal of all patient identifiers from the database. The de-

identification measures are described in detail in Section 12.2.

# 5 Design Requirements

To useful in a research context, MIMIC needs to scale to large numbers of records. It should take advantage of data collection resources at the partner hospital as a source of data. The following sections describe other design ideals and requirements specified for this project.

## *5.1 Provide Easy Access*

MIMIC should be easily accessible to researchers. Scientists seeking real patient data should not have to download the entire database or install specialized software to view records. Instead, MIMIC is intended to be immediately available to any researcher, free of charge, and have the flexibility to be useful for a wide range of research problems.

## *5.2 Protect Patient Confidentiality*

The partner hospital allows MIMIC access to its ISM under the condition that all data in MIMIC must be sanitized to protect patient identities. "The Hippocratic oath incorporated the principle of medical confidentiality into doctors' professional ethics."[8] MIMIC has the responsibility of de-identifying the data so that patients are not identifiable by the records available in the MIMIC database. Patient records often contain identifiers such as names, phone number, Social Security Number (SSN), Medical Record Number (MRN). These must be removed from patient records before they are added to MIMIC.

## *5.3 Data Accuracy*

MIMIC must assure that data is downloaded and displayed accurately. "If information is corrupted, clinicians may take incorrect decisions which harm or even kill patients. If information is unreliable, in the sense that it could have been corrupted, then its

value as a basis for clinical decisions is diminished."[10] In order to be useful, the data in MIMIC must be accurate. Because the data in MIMIC is collected from the primary source, a higher level of accuracy is achieved than databases that require transcription from paper records.

# 6 Overall Architecture

The MIMIC utility consists of two major parts: the RDBMS and the application. The **MIMIC server** refers to the machine on which these parts reside. MIMIC utilizes the hospital's ISM is the source of data. There are also other minor components to MIMIC and modules that support MIMIC, that select, de-identify, parse, and upload new data. These modules are described in more detail in the following sections.

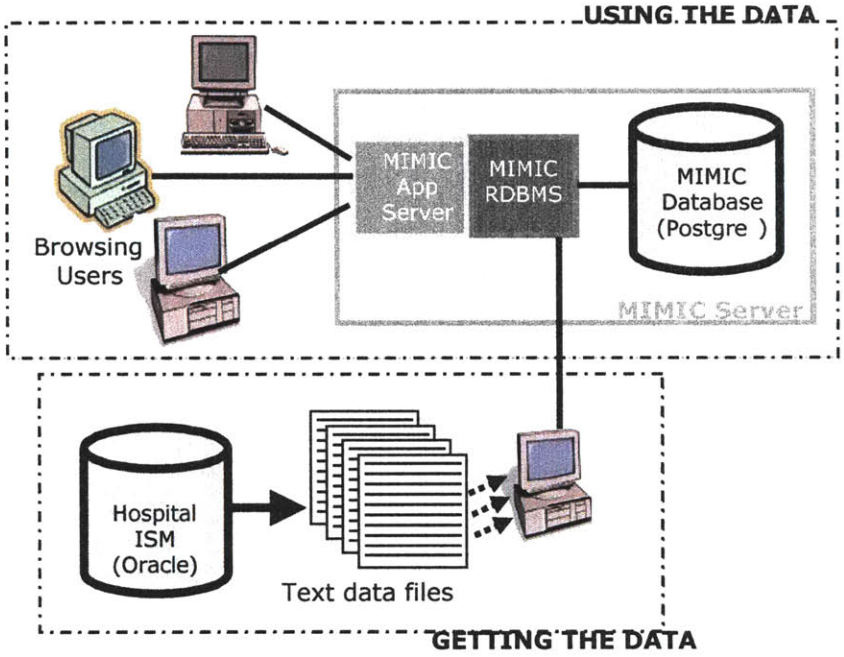



*Figure 2: Architecture of MIMIC*

# 7 Tools

A Postgres relational database was chosen for the MIMIC database because it is open-source and free of charge. AOLServer is used for the MIMIC Application because it has been successfully used with Postgres to create large-scale database-based web sites[11]. The Tcl scripting language is used to dynamically generate HTML pages and interface the web server and database.

# 8 Data Collection

As described in Section 2, the partner hospital has 42 Intensive Care Unit (ICU) beds in that are presently connected to the CareVue system. Nursing notes, medications, fluid input and output, updates to patient charts, lab results, and more are stored in the CareVue System. Waveforms such as ECGs, blood pressure, and pulse oximetry are stored in a separate waveform collection system. MIMIC downloads clinical data from the hospital ISM and stores it in a new database, the MIMIC database. Waveform data is downloaded from the waveform collection system at the hospital and stored in a separate database, which is the waveform counterpart to MIMIC.

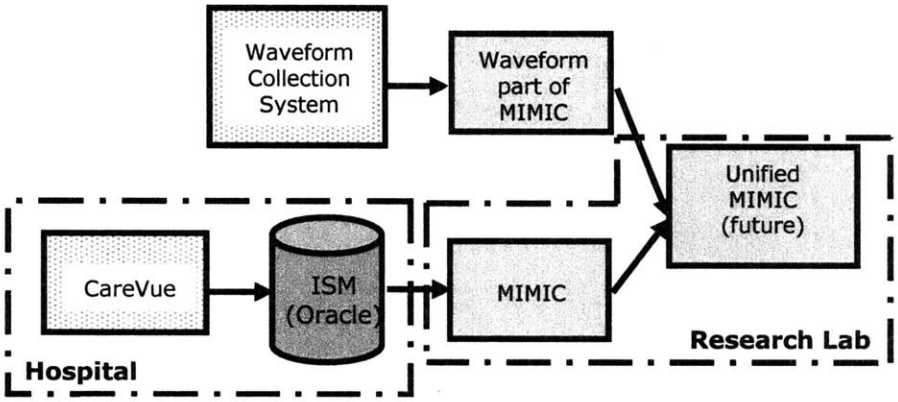



*Figure 3 MIMIC and Data Collection*

In the future, these two projects will be merged to create a unified utility to view and search through clinical and waveform data.

# 9 Types of Users

Two classes of users will interact with MIMIC. **Regular users** include normal researchers who are looking for interesting data to support their research. **Administrators** will generally have some technical knowledge, be familiar with the Administrator's manual for MIMIC (see Appendix E) and have administrative privileges on the MIMIC Server. Administrators have access to the MIMIC RDBMS and are able to upload new data.

# 10 MIMIC RDBMS

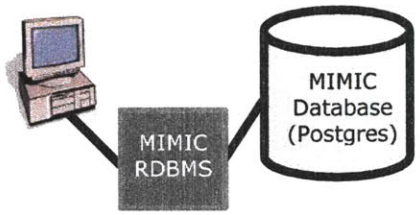



*Figure 4 The MIMIC RDBMS*

The MIMIC RDMS allows administrators to create, update, and administer the Postgres relational database that stores records for MIMIC. It provides a user interface to access the database without requiring the user to know SQL or specific data models. The MIMIC RDMS itself consists of a few metadata tables that store the table definitions and a set of Tcl pages that modify and display table definitions. The complete data models for these metadata tables appear in Appendix A

## *10.1 Managing Tables*

The MIMIC RDMS allows users to define tables and columns within tables and specify how they are used and displayed. Users can specify the database data type for the entry, i.e. `integer or varchar(100)`. The field for `extra_sql` can be used to specify whether this column references another table. An `order sort key` is used to order the elements in a table. Administrators can also specify whether the column should be included in the text search, and whether the column should be hidden.

Once the tables are defined, users can use the MIMIC RDMS to generate SQL `CREATE TABLE` and `DROP TABLE` statements. These scripts can be cut and pasted into Postgres to perform these operations. This process is described in step-by-step detail in the Administrator's Manual in Appendix E.

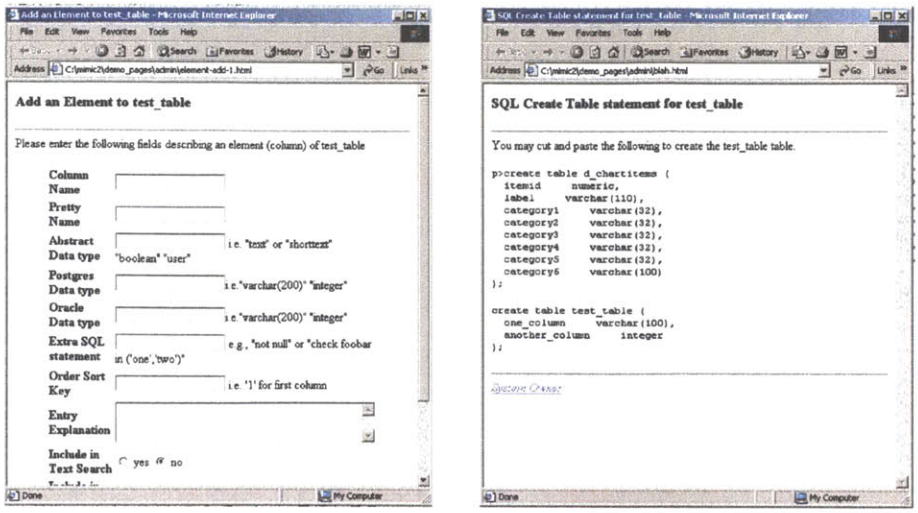



*Figure 5 Creating Table Definitions*

### 10.1.1 MIMIC Table Definitions

There are currently 34 table definitions entered into the MIMIC RDBMS. These are based on the CareVue ISM data models. There are slight differences between the data types defined in ISM and those used for MIMIC due to differences in Postgres and Oracle. The values entered for these tables can be found in Appendix B.

The ISM classifies tables into two types: **dimension tables** and **fact tables**. Dimension tables "provide details around the who, what, where, and how of the data."[12] For example, the `d_patients` table contains the patient's sex, dob, and a database-generated pid (patient ID). Other tables that chart patient data will contain a value for the pid of the patient.

Fact tables are used to "contain the raw data charted in <the> CareVue system. These properties contain the actual values being charted. The fact table also contains other properties which point to dimension tables."[8] In general, dimension tables are used to support fact tables and fact tables contain patient data. MIMIC stores both types of tables.

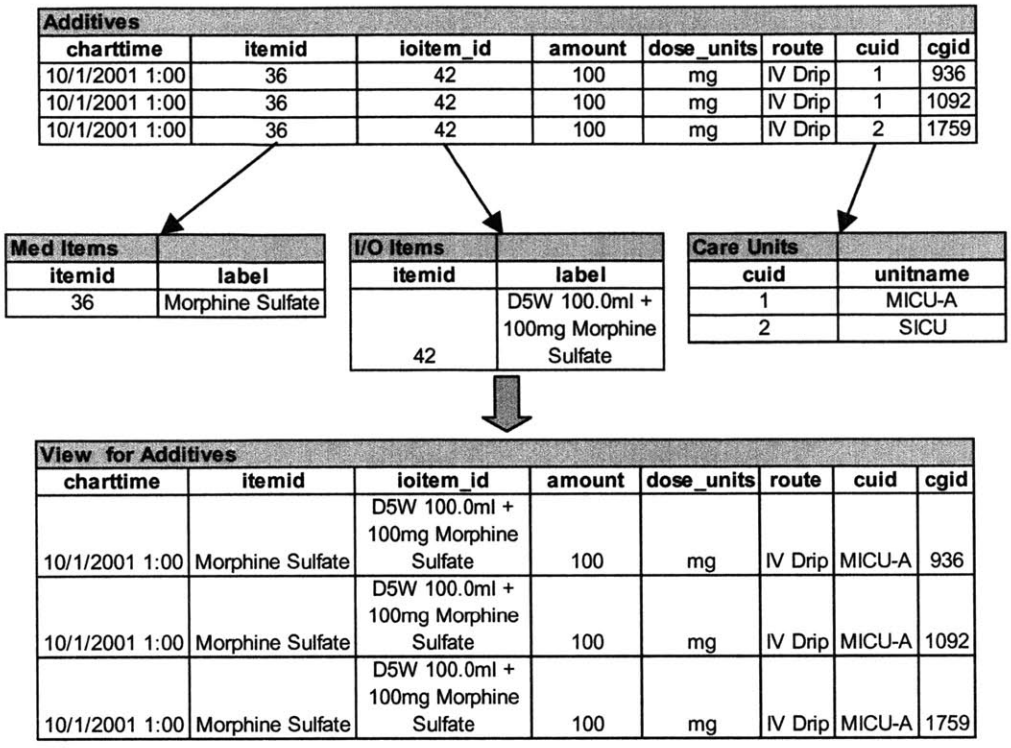
### *10.2 Managing Indexes*

The MIMIC RDMS also includes a module to create and modify table indexes. SQL indexes are used to improve the performance of common queries. Indexes can be created on a column or a collection of columns in a table based on the structure and usage of data. Users are able to use the MIMIC RDMS to define indexes and generate SQL `CREATE INDEX` and `DROP INDEX` statements. A list of indexes created for MIMIC tables can be found in Appendix C.

### *10.3 Creating Views*

A SQL view is used to provide an additional layer of abstraction between the way data is presented and its underlying data structures. MIMIC uses views to present data in a form that is intuitive to the user. For instance, tables that use an Item ID to reference a value in a dimension table will contain numerical values. However, the view created for this table (named view_for_table_name in MIMIC) will replace Item Ids with their corresponding text. Views are automatically generated based on table definitions and created and dropped at the same time as tables.

Postgres does not support materialized views, but their capability is imitated through creating of tables that act as views. When new data is entered into mimic, these table views are dropped and re-created to assure that new data is included.



**Additives**

| charttime | itemid | ioitem_id | amount | dose_units | route | cuid | cgid |
|---|---|---|---|---|---|---|---|
| 10/1/2001 1:00 | 36 | 42 | 100 | mg | IV Drip | 1 | 936 |
| 10/1/2001 1:00 | 36 | 42 | 100 | mg | IV Drip | 1 | 1092 |
| 10/1/2001 1:00 | 36 | 42 | 100 | mg | IV Drip | 2 | 1759 |

**Med Items**

| itemid | label |
|---|---|
| 36 | Morphine Sulfate |

**I/O Items**

| itemid | label |
|---|---|
| 42 | D5W 100.0ml + 100mg Morphine Sulfate |

**Care Units**

| cuid | unitname |
|---|---|
| 1 | MICU-A |
| 2 | SICU |

**View for Additives**

| charttime | itemid | ioitem_id | amount | dose_units | route | cuid | cgid |
|---|---|---|---|---|---|---|---|
| 10/1/2001 1:00 | Morphine Sulfate | D5W 100.0ml + 100mg Morphine Sulfate | 100 | mg | IV Drip | MICU-A | 936 |
| 10/1/2001 1:00 | Morphine Sulfate | D5W 100.0ml + 100mg Morphine Sulfate | 100 | mg | IV Drip | MICU-A | 1092 |
| 10/1/2001 1:00 | Morphine Sulfate | D5W 100.0ml + 100mg Morphine Sulfate | 100 | mg | IV Drip | MICU-A | 1759 |

*Figure 6 Sample View Created for the Additives Table*

## *10.4 More Metadata*

In addition to the metadata defined in the table creation process, the MIMIC RDMS also uses other metadata for managing and displaying tables.

### 10.4.1 Display Keys

A **Display Key** for a table is the column that should be used as a label for this table. In some views where only one column for a table is displayed, this label is used. For example, in the `d_meditems` table, the `label` is the display key. Other tables will reference the `d_meditems` table. When it does, the label for `d_meditems` is used to display that value.

### 10.4.2 Time Keys

**Time Keys** are used to select and order data. These are generally columns with the `timestamp` data type. Each fact table is ordered by a time key so that results can be displayed in chronological order.

### 10.4.3 Date Keys

**Date Keys** are also used to select and order data. These are columns with integers that denote which day the item was stored. These date keys reference the `d_days` table, which has entries for every day from January 1, 1970 to December 30, 2030. Date keys are helpful in finding records for a specific day.

### 10.4.4 Menu Keys

**Menu Keys** are used in displaying data in a menu fashion. Administrators can specify a column for each table and whether this menu key should be visible. This controls the menus that are displayed to normal users. For instance, the Total Balance Events table has Item ID as its menu key. This means that the values of this column are available as menu options. One value for an Item ID is *'24 Total Out.'* A user could click on this item to view only entries in the column with that menu value. For this example, the result would be a table of only *'24 Total Out'* entries. Figure 13 shows and example of a menu for a table.

Current values for the keys described above can be found in Appendix D. These can be modified and managed using the key management module in `/server_path/admin/mimic/keys`.

# 11 Data Extraction and Migration

This section describes the process of obtaining new data from the hospital system for upload to MIMIC. MIMIC uses a simple ETL (extract, transform, and load) architecture to add new data to the database.

## *11.1 Connecting via CareWeb*

We obtain access to the hospital's data collection system by connecting remotely via a virtual private network (VPN). In this way we are able to access the hospital's ISM without having to be on-site. MIMIC contains records only for those patients who

have been discharged from the ICU, to assure that each record is a complete picture of a patient's stay.

### *11.2 Choosing Records for Download*

Data is chosen to be included in MIMIC based on available waveform data. Another project, the waveform counterpart to MIMIC, computes wavelet coefficients for analysis and trending. MIMIC is intended to complement that project by organizing clinical records for the patients with corresponding waveforms. Data is chosen to be included in MIMIC based on the waveform data that is collected and processed.

### *11.3 Extracting Data*

To extract data from the hospital system, a user needs administrative privileges on MIMIC, a web browser (such as Netscape), access to the MIMIC Server, access to the hospital ISM (Oracle), and a list of patients. The list of patients should be in the form:

```
case_id|last_name|first|name|MRN
```

The MRN is optional. An example appears below.

```
3551|LADEN|JOSEPH|
3549|STEELE|REMINGTON|12428752
3546|FIELDS|ANDREW|
3541|COLE|KENNETH|
3542|PIERFOR|ANTHONY|
3545|YOUSEFFIAN|THOMAS|35408266
3564|WALDEN|LAMAR|
3565|GARCIA|ANDY
```

This file is then uploaded in the data update page to generate a script to extract patient IDs for these patients. Patient IDs are integers used in the ISM as private keys for each patient. Patient IDs are not associated with MRN or any other identifying information for the patient and are only used internally.

With this list of Patient IDs, another script is run on the hospitals Oracle system to extract data for these patients.

### *11.4 Reading from the Legacy Format*

Postgres and Oracle are largely similar, but have a few differences with respect to data types and how values are formatted. MIMIC uses Oracle's formatting functions to select data in a format that Postgres can recognize, based on metadata from table definitions. For instance, Oracle has one data type for storing dates, `date`. The default format for selecting the `date` data type returns a string as 'YYYY-MM-DD.' MIMIC tables are largely concerned with dates that are associated with times that an event was charted and use the `timestamp` type to store these dates. To extract the full timestamp of a `date` column in Oracle, we can formulate the `select` statement to select the column as a full timestamp. A similar approach can be applied to other data types to obtain data in the desired format.

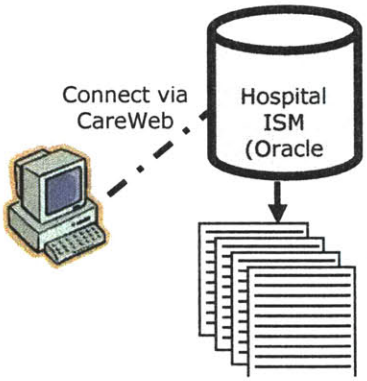



*Figure 7 Downloading Data from the Hospital ISM*

Data is downloaded into a plain text file, which takes more time than simply dumping tables from Oracle, but allows more flexibility in choosing only desired records. This also prevents having to convert from an Oracle format to one that Postgres can recognize.

```
1102|2001-07-18 22:00:00|2001-07-19 18:42:00|27|1|1242|||0|1
1102|2001-08-30 00:00:00|2001-09-09 23:00:00|27|1|15780|||0|1
1102|2001-09-24 14:00:00|2001-10-05 21:56:00|27|1|16316|||0|1
1102|2001-10-10 23:00:00|2001-11-01 11:10:00|27|1|30970|||0|1
1102|2001-07-18 22:00:00|2001-07-19 18:42:00|31|1|1242|||0|1
1102|2001-08-30 00:00:00|2001-09-09 23:00:00|31|1|15780|||0|1
1102|2001-09-24 14:00:00|2001-10-05 21:56:00|31|1|16316|||0|1
1102|2001-10-10 23:00:00|2001-11-01 11:10:00|31|1|30970|||0|1
```

*Figure 8 Sample Text Data File*

The output is saved as .csv (comma spaced values) files that can later be parsed and inserted into the MIMIC database. A sample of one of these files appears in Figure 8. This process also allows us to perform some preliminary de-identification of the data,

which is described more in Section 12.2.

### *11.5 Migration*

Since the files containing the data for the new records are large (75 MB for 100 patients), we use ftp to transfer them to the local server before parsing and inserting into the MIIMC database. The text format of the files avoids compatibility issues with versions of Oracle or Postgres by avoiding proprietary or database-specified formats. The data extraction script is generated each time new data is requested so that updates to data models (table definitions) do not disrupt the data migration process.

# 12 Adding New Data

Once the data is selected and downloaded from the hospital ISM and written to text files, these files must be uploaded to the MIMIC database.

### *12.1 Uploading New Data*

When these text files are transferred to the local system, they are ready to be parsed and entered into the MIMIC database. The administrator specifies the location of the data files using the data update page located at `/server_path/admin/mimic/data/`.

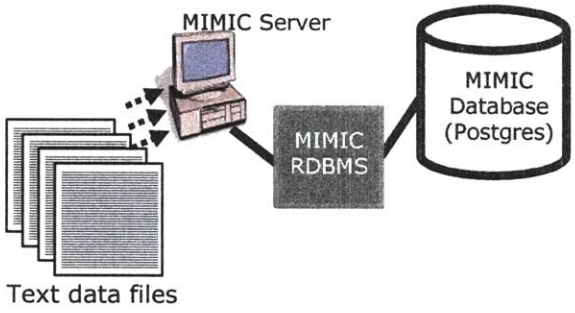



*Figure 9 Uploading New Data to MIMIC*

The MIMIC server then searches for a .csv file for each table defined in its data model. If a table_name.csv file exists, the indexes for that table are dropped in the database to speed the insertion of new data. More about index management can be found in the Administrator's Manual in Appendix C. MIMIC reads the text files and creates new files that are formatted according to the Postgres specification by getting rid of extraneous white space,

formatting null values, and putting each row on a separate line. The newly formatted file overwrites the original file. Once each file is formatted, MIMIC uses a 'COPY' statement to add that data to the corresponding table. The indexes for this table are re-created and the update for this table is complete. This process is repeated for each table defined in the MIMIC RDBMS.

### *12.2 De-Identification*

Part of the de-identification process is built into the data migration process. Much of the confidential patient information is simply not selected to be in the data files, so fields such as patient name and caregiver name are not included in the download. They are part of the hospital's ISM, but not defined for MIMIC.

Another way to protect patient identities is by using a unique key called a patient ID to identify each patient.

> "Unique Patient Identifier eliminates the need for the repetitive use and disclosure of an individual's personal identification information (i.e. name, age, sex, race, marital status, place of residence, etc.) for routine internal and external communications (e.g. orders, results, medication, consultation, etc.) and protects the privacy of the individual. It helps preserve the patient anonymity while facilitating communication and information sharing." [13]

The pid (patient ID) is assigned by the ISM. New patient IDs are assigned for each new admission, even for the same patient. MIMIC uses a Case ID to identify patients. Each patient is assigned a Case ID, and the Case ID is reused for repeat admissions.

The second line of defense for patient de-identification uses a function to replace patient names before writing the data to a file. This function runs on the hospital system when new data is being requested and works by looking up the patient name and replacing any instances of that name with 'patient.' The result is that data files do not contain any patient names. This function does not make any changes to the records on the hospital system, but only modifies the results of a query.

The resulting data files may still contain some personal information that must be removed before the new entries can be published. Part of the data upload process

```
SELECT fullname INTO temp_lastname from d_patients WHERE pid =
find_pid;
   new_string := replace(lower(replace_string),
                         lower(temp_lastname), patient);
   temp_num:=instr(temp_lastname, );
   temp_firstname:=substr(temp_lastname, 0, temp_num);
   temp_lastname:=substr(temp_lastname, temp_num);
   new_string := replace(new_string, temp_firstname, patient );
   new_string := replace(new_string, temp_lastname, patient );
```

*Figure 10 Function to Replace Names*

includes a script that deletes entries that contain personal information such as Spokesperson Phone Number and Insurance Number. We have identified 13 such fields for deletion: "Attending MD", "Spokesperson", "Spokesperson Home Phone #", "Spokesperson Work Phone #", "Religion", "Martial Status", "Home Phone No.", "Work Phone No.", "Page Phone No.", "Pt. Name", "PIN", "Cell Phone No.", and "Account #'.

Some of the free text fields may also contain phone numbers, Social Security numbers, or medical record numbers. Before new data is uploaded, a script looks for patterns (for example, a series of numbers of the form xxx-xxx-xxxx) that could be an identifying number and replaces it.

### *12.3 Logging*

Once all new data has been uploaded and entered into the MIMIC database, the update is stored in a data log. The log records the date, location of .csv source file, and the results of the update. This can be useful to administrators in evaluating when updates were made and outcome of the update.

# 13 The MIMIC Application

Now that the data is loaded into the MIMIC Database, it is ready to be presented to users. Each patient can have multiple days' worth of data, stored in multiple tables, each containing thousands of entries. The pages of the MIMIC Application were designed to provide a unified and intuitive user interface that presents a great deal of data in a way that is easy to navigate and comprehend.

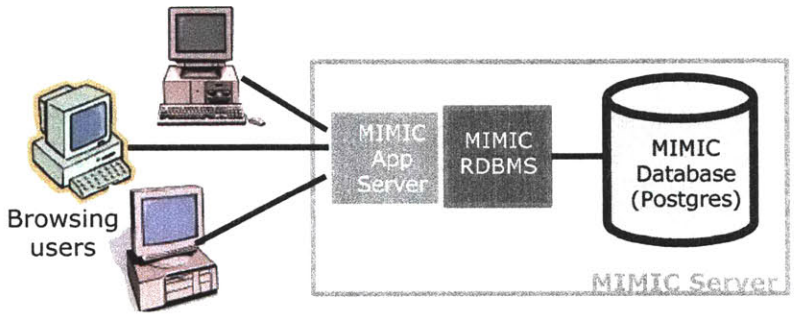



*Figure 11 The MIMIC Application*

The MIMIC Application consists of .tcl pages that interact with the MIMIC database to dynamically generate html pages to present to the end user. The MIMIC Application consists of two main sections: data pages and search pages. The data pages present patient data in an intuitive and easily browsable manner. The search pages provide facilities to formulate queries on multiple dimensions.

## *13.1 General Presentation*

In creating a user interface to MIMIC, we tried to keep the design as simple and utilitarian as possible. The result is a user interface that is clean, compact, and allows users to browse through records using several different approaches.

### 13.1.1 Hiding internals

One way MIMIC makes data more easily viewed is by hiding the internals of how data is stored and accessed. MIMIC includes metadata about whether columns or categories should be hidden or presented to users. Fields such as `case_id` or `date_key` are numeric values that are meaningless to end-users and have no clinical

significance. These are not presented in normal user views. MIMIC also uses SQL views to decode references to other tables and provide a level of abstraction between the raw data and a format that is easily understood by the end user.

### 13.1.2 Utilizing data types (metadata)

Another way that MIMIC improves upon the appearance of raw data is through utilization of data types stored in metadata. MIMIC uses this metadata to format data for display. Time stamps are passed through a conversion function to format them as text instead of ANSI format. Columns that are integers are truncated to leave out insignificant figures. MIMIC uses time keys to present data in chronological order. Other keys are used to view data based on category, table, or item type. These different types of views are described in more detail in the next section.

## *13.2 Viewing the Data*

MIMIC views organize data into three basic views. A patient record can be **viewed by table**, which presents the first 25 entries of each table in one page. The record can be **viewed by day**, which presents the first 25 entries of each table for that day in one page. The **category view** presents the first 50 entries of each table in that category in one page. It is not reasonable to present more than a few entries for each table on the same page, so MIIMC presents a limited number from each table and adds a menu to browse the rest of the entries. In addition to these views, MIMIC provides the capabilities to concentrate on specific types of data

## *13.3 Viewing Medications*

MIMIC currently uses five different tables to store medications information: `a_medurations`, `d_meditems`, `solutions`, `additives`, and `medevents`. All of these reference the `d_meditems` table for medication names. The MIMIC Application includes a medications section to view all medications information for a given patient, or concentrate on a specific medication for a given patient. This provides a

useful summary that can be used in determining how a medication was part of a patient's course of care or how it affected his prognosis.

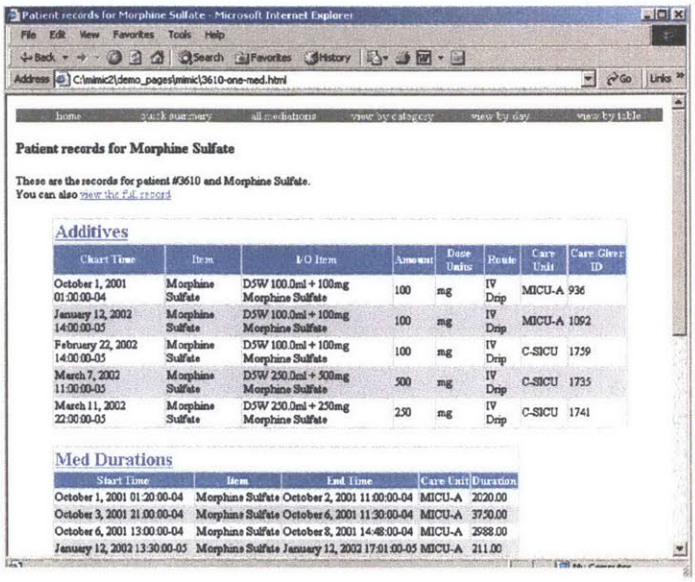



*Figure 13 Viewing Medications*

## *13.4 Menus for Browsing*

Most of the tables containing patient information reference dimension tables for the items they store. Because they reference dimension tables, these items are standardized across different entries. A column that references the `d_chartitems` table will contain multiple entries for the same `Item ID`. MIMIC utilizes the normal form of this data to create menus to view and browse all items with the same `Item ID`. In practice, this means this means there are menus to concentrate on a specific type of entry to a table. For instance, the Total Balance Events table has menus for *'24 Total Out'* and users can click on a value to view all entries for that specific item for a given patient.

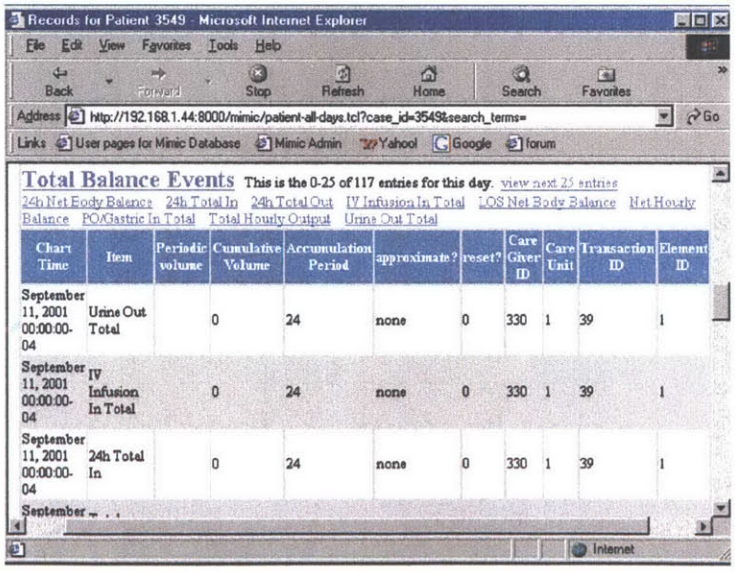



*Figure 12 Menu for Total Balance Events*

# 14 Search

The interesting capabilities in MIMIC lie in the search engine. MIMIC uses an incremental search to merge the results of criteria on multiple dimensions. This search allows users to view incremental results at each step of the search. Incremental search can further help the researcher pinpoint interesting results from the collection of records stored in MIMIC.

When the user gets to the initial search page, he can choose to add a search term based on a text search, medications search, or patient demographics. These types of searches are described and defined in more detail in the following sections. Once the user enters the search criteria, the search terms are sent to be processed. The user can then see the number of results in MIMIC that satisfy these criteria, and opt to perform the search or enter additional search terms. When no records satisfy the search criteria, the criteria are removed from the search and a notification message appears to the user.

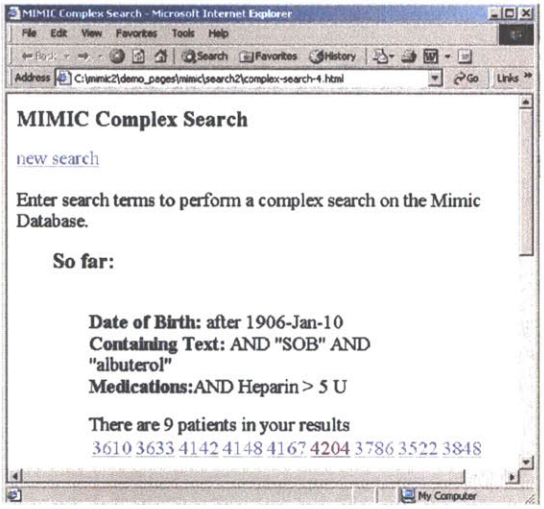



*Figure 14 Search Results*

More on these types of searches and a few details about their implementation are described in the following sections.

## *14.1 Patient Demographics*

The first and simplest search is based on patient demographics, namely age and sex. Age is specified by denoting a date of birth (DOB) greater than or less than a given

date. Obviously, the choices for sex are male or female. All patients in MIMIC have data for DOB and sex. MIMIC performs searching based on patient demographics by simply scanning the `d_patients` table, which contains this information.

### *14.2 Text Search*

One of the options for managing columns in MIMIC is whether the column should be in included in a text search. In general, this field should be turned on for text fields that are relevant to the patient record. Only these fields are considered for a text search.

The text search facility allows users to enter in text search terms and searches free text for these fields. Users may enter simple logic, such as '*patients AND doctors*. MIMIC parses the input to the text search and searches for these search terms. It automatically detects key words for logic 'AND' and 'OR' and generates SQL for these cases.

The design of the text component of the MIMIC Search engine is similar to the specifications for an early version of the bboard search on http://www.photo.net. It uses a Tcl ranking function to sort results and presents the most relevant result first. "It does a simple ranking based on a list of keywords – it is not phrase-based. The more keywords that are matched, the higher score you get."[14] Ranking is based on items matched for each patient record. For instance, a user may search for Atrial Fibrillation. A patient whose record contains 5 instances of Atrial and 3 instances of Fibrillation would receive a score of 8 in the search results.

MIMIC uses a Tcl-based ranking function and SQL '`like`' comparison statements to perform the search. SQL '`like`' statements match each element of the input string. The results are stored and ranked in a Tcl script. In text-only search, results are

returned in order of number of relevance. For more complex searches, results are returned in an arbitrary order based on multiple search terms.

### *14.3 Medications Search*

Medications are stored in the `d_meditems` table, which helps standardize spelling and notation associated with medications. Patient records reference `d_meditems.itemid` to record that a patient received a medication. Figure 6 shows an example of viewing medication records. Since we know which tables reference the `d_meditems` table and therefore contain medications information, we can limit the medications search to these tables. Since searching for medications is based on an integer key, it is faster than a text search because numerical comparison is faster than text comparison.

## 15 Design Issues

In the course of designing the components of MIMIC, there were many decisions and tradeoffs considered. In general, design decisions were made according to efficiency and simplicity.

### *15.1 Performance*

Early versions of the MIMIC Application were extremely slow. Page views took about 6 seconds to load, due to the time to process queries to the database. Most web users are unwilling to wait more than 2 seconds for a page to load. A great deal of time was spent on creating indexes to speed up queries and reduce the amount of time to retrieve data.

Another optimization measure used was to move some of the processing for queries to the data upload process. When new data is added to the MIMIC database, new table views are created and updated at the same time. These views perform the costly JOIN operations needed to generate the materialized user views described in

Section 10.3. This increased data upload time up to 100%, but once the data was processed, page views took 2 seconds. The alternative would be to create simple SQL views, instead of tables that act as materialized views, and perform the JOIN operations whenever a page was requested. The data upload process was much faster, but pages took 6-8 seconds to load. The end decision was to move the JOIN latency to the upload process to assure that page loads were fast. This makes MIMIC more convenient for end users who are browsing records, and adds only start-up costs to adding new data.

A few other approaches to optimization were also considered, including query caching and warehousing strategies. In the end, a combination of creating indexes, utilizing integer comparisons when possible, and an ad-hoc version of materialized views were used to optimize the MIMIC Application.

### *15.2 Scalability*

The MIMIC database will have hundreds of new records every few months. The partner hospital currently has 42 beds connected to the CareVue system. Over the course of a year, over 10,000 patient days' worth of data is available to MIMIC. MIMIC must scale efficiently to handle these records. Relational databases have been proven to handle scaling to thousands of records efficiently. MIMIC takes advantage of this by using Postgres for its database.

The MIMIC Application is designed to scale to large numbers of records. The search engine returns incremental results, but stores only the different search terms. Storing incremental results for the search (instead of search terms) and re-calculating only new terms as they are added becomes more complicated as the number of records increases. As MIMIC grows to thousands of records, it becomes

more efficient to store search terms and perform the entire search at each step.

The user interface for MIMIC was also designed with large numbers of patient-days in mind. The average stay for an ICU patient can vary greatly, so the user interface was designed to display varying-length records efficiently. This is achieved through menus to browse menus and a range of display options.

### *15.3 Versatility*

The difference between MIMIC and similar databases is that MIMIC was not developed for a specific application or problem. Some similar databases that were developed for more specific research problems were described in Section 3.2. The MIMIC database contains all available records from patients in one of the partner hospital's intensive care units. Data is recorded from multiple data sources and can be browsed based on medication, type of event, or category. The search utility is intended to help researchers locate data that is relevant to their research. Unlike other databases, MIMIC does not target specific problems, but aims to be a general-purpose source of ICU patient data.

# 16 Evaluation of Current Design

### *16.1 De-Identification*

The de-identification measures taken in the data upload process assure that no patient names, phone numbers, medical record numbers, or social security numbers appear in data for MIMIC. However, MIMIC does not handle other possibly identifying information, such as doctor names or names of family members. Nursing notes occasionally contain references to a patient's "sister Mary" or "home in Bedford." Nursing notes may also contain misspellings that go undetected. MIMIC will replace "Robert" but will not find "Robret." MIMIC lacks the sophisticated facilities to remove such information. In future versions, natural language processing algorithms can be

applied to solve these problems.

### *16.2 Scalability*

MIMIC currently contains about 300 patients, with a total of over 1,000 patient days. MIMIC should easily scale to contain several hundred more patients. We are unsure of how MIMIC will handle tens of thousands of records. The MIMIC server, which is currently on a machine with a 1500Mhz Athlon AMD processor and 512MB memory, would have to be upgraded to be a server for thousands of patient records. Once MIMIC is merged with its waveform counterpart and regular data updates are made to the MIMIC Database, the database should be backed up regularly and the MIMIC Server used as a dedicated server for the MIMIC Application.

### *16.3 Usability*

There are many possible additions to MIMIC that would make it more useful to researchers. Once MIMIC has been merged with its waveform counterpart, complex trending and analysis of those waveforms can be combined with searches on clinical records. Currently, MIMIC does not support any conversion of units of measure for medications searches, due in part to lack of available data. Units conversion may be possible for future versions of MIMIC.

The display for MIMIC varies slightly across different web browsers. MIMIC was tested mostly using Internet Explorer, Netscape, and Mozilla. Future versions of MIMIC could add to the current design by using HTML elements that are displayed uniformly across different browsers.

### *16.4 The Future of MIMIC*

This project is the second version of an effort to solve the problems or collecting and organizing real patient data for clinical research. However, there can still be improvements and functionality added to MIMIC that could improve its value.

MIMIC only contains a portion of the data recorded from the ICU. The hospital also collects discharge summaries, pathology reports, ECG signals, more detailed lab reports, records for surgeries, X-rays, EEGs, outpatient care, and more. This data is not currently stored in the ISM, but is stored on other hospital systems. In the future, we could gain access to their records and add them to MIMIC. The MIMIC RDBMS would have to be extended to include new table definitions and a different extraction system to access the hospital information system. The richer content of the resulting database would widen the range of research problems that MIMIC supports.

In the future, MIMIC and its waveform counterpart will be integrated into a unified resource for patient data. New versions of MIMIC could also include more sophisticated patient de-identification procedures and add different user views. The search facilities could allow for more complex searches, including searches on waveform patterns. The current framework for MIMIC leaves the ability to extend to include these capabilities. Even without these improvements, MIMIC will be a useful utility to researchers looking for real patient data.

# 17 Conclusion

MIMIC provides a solution to some of the problems of protecting patient confidentiality, migrating data to a new server, and presenting data in a usable interface. It utilizes web and database technologies to create an application that makes real hospital records available and searchable by researchers. Regular updates from the hospital ISM will populate MIMIC with real patient data. All of these elements are combined to create a useful utility for researchers looking for real patient data.

# 18 Bibliography


[1] Kohane, Isaac "The Imperative to Collaborate" Journal of the American Informatics Association Volume 7 Number 5 Sep/Oct 2000

[2] Product Description for Philips Medical Systems CareVue Clinical Information System http://www3.medical.philips.com/en-us/product_home/product/carevuecis_detail.asp

[3] Moody, George and Mark, Roger "A Database to Support Development and Evaluation of Intelligent Intensive Care Monitoring" Computers in Cardiology 23:657-660 1996

[4] Wilcox, Adam, et al. "Developing Online Support for Clinical Information System Developers: The FAQ Approach" Computers and Biomedical research, **31** 112-121 (1998) Article No. CO 981470

[5] Korhonen, Ilkka. vanGils, Mark. Gade, Jon. "The Challenges in Creating Critical Care Databases" IEEE Engineering in Medicine and Biology May/June 2001

[6] The HIV Central Research Database http://www.ohtn.on.ca/5_central_hiip.html)

[7] Stanford Penguin Project http://smi-web.stanford.edu/projects/penguin.html

[8] "A Model Electronic Patient Record System for Clinical Echocardiography http://hsc.virginia.edu/hs-library/newsletter/1994/november/echocard.html

[9] Duke Medical Informatics Research http://dmi-www.mc.duke.edu/dukemi/research/research.html

[10] Anderson, Ross , "Security in Clinical Information Systems" Computer Laboratory University of Cambridge - January 1996)

[11] Greenspun, Philip, "Phil and Alex's Guide to Web Publishing" http://www.arsdigita.com/books/panda/

[12] CareVue Clinical Data Management Information Support Mart User's Guide. Agilent Technologies, 2000

[13] Unique Patient Identifier http://www.hipaanet.com/upin4.htm

[14] Idea for doing Site-Wide Search http://openacs.org/bboard/q-and-a-fetch-msg.tcl?msg_id=0000Sy&topic_id=11

# APPENDIX A: Metadata Tables

```
create table mimic_table_elements (
 metadata_id             integer        not null,
 table_name              varchar(21)    not null,
 column_name             varchar(30)    not null,
 pretty_name             varchar(100)   not null,
 abstract_data_type      varchar(30)    not null,
 oracle_data_type        varchar(30)    ,
 extra_sql               varchar(4000)  ,
 presentation_type       varchar(100)   not null,
 presentation_options    varchar(4000)  ,
 entry_explanation       varchar(4000)  ,
 help_text               varchar(4000)  ,
 include_in_view_p      char(1)        ,
 mandatory_p              char(1)       ,
 sort_key                 integer       ,
 form_sort_key            integer       ,
 form_number              integer       ,
 include_in_ctx_index_p  char(1)        ,
 default_value           varchar(200)   ,
 order_sort_key           integer       ,
 postgres_data_type      varchar(80)
);


create table mimic_table_categories (
 category_name    varchar(100)  not null,
 cat_pretty_name  varchar(100)  not null,
 description      varchar(400),
 include_in_view_p varchar(1),
 order_sort_key   integer
);


create table mimic_time_keys (
 table_name  varchar(100),
 time_key    varchar(100)
);

create table mimic_date_keys (
 table_name  varchar(100),
 date_key    varchar(100)
);
```

```
-- for display of tables by menu item
create table mimic_item_keys (
 table_name           varchar(21),
 item_key             varchar(30),
 include_in_view_p    varchar(1)
);

-- for display
create table mimic_display_keys (
table_name varchar(100) primary key,
display_key varchar(100)
);

-- data log for updates
create sequence mimic_indexes_sequence;
create table mimic_data_log (
 update_id          integer,
 enter_date         timestamp with time zone,
 table_name         character varying(40),
 filename           character varying(100),
 file_status        character varying(400),
 extract_key        character varying(40),
 extract_key_start  numeric(30,6),
 extract_key_stop   numeric(30,6),
 insert_status      character varying(400)
);

- for management of indexe

create table mimic_indexes (
index_name   varchar(100),
table_name   varchar(21),
column_name  varchar(30),
primary key (index_name, table_name, column_name)
);
```

# APPENDIX B: MIMIC Table Definitions

```
-- SQL create table statements for all tables
-- You may cut and paste the following to create tables.

-- tables for Dimensions
create table d_caregivers (
  cgid     numeric,
  employeeno     varchar(20),
  proftitle     varchar(6)
);

create table d_careunits (
  cuid     numeric,
  unitname     varchar(20)
);

create table d_chartitems (
  itemid     numeric,
  label     varchar(110),
  category1     varchar(32),
  category2     varchar(32),
  category3     varchar(32),
  category4     varchar(32),
  category5     varchar(32),
  category6     varchar(100)
);

create table d_days (
  dayid     numeric,
  calDay     timestamp,
  month     numeric,
  dayofmonth     numeric,
  year     numeric,
  monthText     varchar(20),
  dayofweek     numeric,
  holiday     varchar(20)
);

create table d_interventionitems (
  itemid     numeric,
  label     varchar(80),
  category1     varchar(32),
  category2     varchar(32),
  category3     varchar(32),
  category4     varchar(32),
  category5     varchar(32),
  category6     varchar(32)
);

create table d_ioitems (
  itemid     numeric,
  label     varchar(256),
  category1     varchar(32),
  category2     varchar(32),
  category3     varchar(32),
  category4     varchar(32),
  category5     varchar(32),
  category6     varchar(32)
);

create table d_meditems (
  itemid     numeric,
  label     varchar(20),
  category1     varchar(32),
  category2     varchar(32),
  category3     varchar(32),
  category4     varchar(32),
  category5     varchar(32),
  category6     varchar(32)
);

create table d_outcomeitems (
  itemid     numeric,
  label     varchar(60),
  category1     varchar(32),
  category2     varchar(32),
  category3     varchar(32),
  category4     varchar(32),
  category5     varchar(32),
  category6     varchar(32)
);

create table d_patients (
  case_id     integer,
  pid     numeric,
  sex     varchar(8),
  dob     date
);

create table d_problemitems (
  itemid     numeric,
```

```
  label     varchar(60),
  category1     varchar(32),
  category2     varchar(32),
  category3     varchar(32),
  category4     varchar(32),
  category5     varchar(32),
  category6     varchar(32)
);

create table d_primarycodes (
  label     varchar(50),
  code     varchar(32),
  pcode     numeric
);

create table d_sources (
  systemid     numeric,
  siteid     numeric,
  sourceid     numeric,
  schemaRev     numeric,
  hospitalname     varchar(60),
  address1     varchar(30),
  address2     varchar(30),
  address3     varchar(30),
  sid     numeric
);

create table d_secondarycodes (
  label     varchar(50),
  code     varchar(32),
  scode     numeric
);

-- tables for Events
create table censusevents (
  pid     numeric,
  intime     timestamp,
  outtime     timestamp,
  careunit     numeric,
  destcareunit     numeric,
  dischstatus     varchar(20),
  los     numeric,
  sid     numeric,
  indayid     numeric,
  outdayid     numeric
);
```

```
create table view_for_censusevents
as select censusevents.oid as oid_for_censusevents,
d_patients.case_id, round(censusevents.pid, 0) as pid,
censusevents.intime as intime, censusevents.outtime as outtime,
d_careunits.unitname as careunit, censusevents.careunit as
key_for_careunit, d_careunits.unitname as destcareunit,
censusevents.destcareunit as key_for_destcareunit,
censusevents.dischstatus as dischstatus,
round(censusevents.los, 0) as los, d_sources.hospitalname as
sid, censusevents.sid as key_for_sid,
round(censusevents.indayid, 0) as indayid,
round(censusevents.outdayid, 0) as outdayid from censusevents,
d_patients, d_careunits, d_sources where
d_patients.pid=censusevents.pid and
d_careunits.cuid=censusevents.careunit and
d_careunits.cuid=censusevents.destcareunit and
d_sources.sid=censusevents.sid ;

create table chartevents (
  pid     numeric,
  charttime     timestamp,
  realtime     timestamp,
  itemid     numeric,
  value1     varchar(110),
  value1num     numeric,
  value1uom     varchar(20),
  value2     varchar(110),
  value2num     numeric,
  value2uom     varchar(20),
  stopped     varchar(20),
  resultstatus     varchar(20),
  annotation     varchar(500),
  cgid     numeric,
  cuid     numeric,
  scode     numeric,
  pcode     numeric,
  chartdate     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_chartevents
as select chartevents.oid as oid_for_chartevents,
d_patients.case_id, round(chartevents.pid, 0) as pid,
chartevents.charttime as charttime, chartevents.realtime as
realtime, d_chartitems.label as itemid, chartevents.itemid as
```

```
key_for_itemid, chartevents.value1 as value1,
round(chartevents.value1num, 2) as value1num,
chartevents.value1uom as value1uom, chartevents.value2 as
value2, round(chartevents.value2num, 2) as value2num,
chartevents.value2uom as value2uom, chartevents.stopped as
stopped, chartevents.resultstatus as resultstatus,
chartevents.annotation as annotation, round(chartevents.cgid,
0) as cgid, d_careunits.unitname as cuid, chartevents.cuid as
key_for_cuid, round(chartevents.scode, 0) as scode,
round(chartevents.pcode, 0) as pcode,
round(chartevents.chartdate, 0) as chartdate,
d_sources.hospitalname as sid, chartevents.sid as key_for_sid,
round(chartevents.elemid, 0) as elemid, round(chartevents.txid,
0) as txid from chartevents, d_patients, d_chartitems,
d_careunits, d_sources where d_patients.pid=chartevents.pid and
d_chartitems.itemid=chartevents.itemid and
d_careunits.cuid=chartevents.cuid and
d_sources.sid=chartevents.sid ;


create table formevents (
  pid     numeric,
  chartTime     timestamp,
  realtime     timestamp,
  formtitle     varchar(40),
  sectiontitle     varchar(40),
  subsectiontitle     varchar(40),
  itemid     numeric,
  value_     varchar(500),
  valuenum     varchar(100),
  uom     varchar(20),
  cgid     numeric,
  cuid     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric,
  chartDay     numeric,
  formid     numeric
);

create table view_for_formevents as select formevents.oid as
oid_for_formevents, d_patients.case_id, round(formevents.pid,
0) as pid, formevents.chartTime as chartTime,
formevents.realtime as realtime, formevents.formtitle as
formtitle, formevents.sectiontitle as sectiontitle,
formevents.subsectiontitle as subsectiontitle,
d_chartitems.label as itemid, formevents.itemid as
key_for_itemid, formevents.value_ as value_,
formevents.valuenum as valuenum, formevents.uom as uom,
round(formevents.cgid, 0) as cgid, d_careunits.unitname as
cuid, formevents.cuid as key_for_cuid, round(formevents.scode,
0) as scode, round(formevents.pcode, 0) as pcode,
d_sources.hospitalname as sid, formevents.sid as key_for_sid,
round(formevents.elemid, 0) as elemid, round(formevents.txid,
0) as txid, round(formevents.chartDay, 0) as chartDay,
round(formevents.formid, 0) as formid from formevents,
d_patients, d_chartitems, d_careunits, d_sources where
d_patients.pid=formevents.pid and
d_chartitems.itemid=formevents.itemid and
d_careunits.cuid=formevents.cuid and
d_sources.sid=formevents.sid ;

create table medevents (
  pid     numeric,
  charttime     timestamp,
  itemid     numeric,
  elemid     numeric,
  chartdate     numeric,
  realtime     timestamp,
  volume     numeric,
  dose     numeric,
  doseuom     varchar(20),
  solutionid     numeric,
  solvolume     numeric,
  route     varchar(20),
  site     varchar(20),
  stopped     varchar(20),
  annotation     varchar(500),
  cgid     numeric,
  cuid     numeric,
  pcode     numeric,
  scode     numeric,
  sid     numeric,
  txid     numeric
);

create table view_for_medevents as select medevents.oid as
oid_for_medevents, d_patients.case_id, round(medevents.pid, 0)
as pid, medevents.charttime as charttime, d_meditems.label as
itemid, medevents.itemid as key_for_itemid,
round(medevents.elemid, 0) as elemid,
round(medevents.chartdate, 0) as chartdate, medevents.realtime
```

```
as realtime, round(medevents.volume, 0) as volume,
round(medevents.dose, 2) as dose, medevents.doseuom as doseuom,
d_meditems0.label as solutionid, medevents.solutionid as
key_for_solutionid, round(medevents.solvolume, 2) as solvolume,
medevents.route as route, medevents.site as site,
medevents.stopped as stopped, medevents.annotation as
annotation, round(medevents.cgid, 0) as cgid,
d_careunits.unitname as cuid, medevents.cuid as key_for_cuid,
round(medevents.pcode, 0) as pcode, round(medevents.scode, 0)
as scode, d_sources.hospitalname as sid, medevents.sid as
key_for_sid, d_secondarycodes1.scode as txid, medevents.txid as
key_for_txid from medevents, d_patients, d_meditems, d_meditems
d_meditems0, d_careunits, d_sources, d_secondarycodes
d_secondarycodes1 where d_patients.pid=medevents.pid and
d_meditems.itemid=medevents.itemid and
d_meditems0.itemid=medevents.solutionid and
d_careunits.cuid=medevents.cuid and d_sources.sid=medevents.sid
and d_secondarycodes1.scode=medevents.txid ;

create table noteevents (
  pid     numeric,
  charttime     timestamp,
  realtime     timestamp,
  category     varchar(26),
  title     varchar(52),
  notetext     varchar(4000),
  correction     varchar(2),
  cgid     numeric,
  cuid     numeric,
  chartDate     numeric,
  sid     numeric,
  noteid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_noteevents as select noteevents.oid as
oid_for_noteevents, d_patients.case_id, round(noteevents.pid,
0) as pid, noteevents.charttime as charttime,
noteevents.realtime as realtime, noteevents.category as
category, noteevents.title as title, noteevents.notetext as
notetext, noteevents.correction as correction,
round(noteevents.cgid, 0) as cgid, d_careunits.unitname as
cuid, noteevents.cuid as key_for_cuid,
round(noteevents.chartDate, 0) as chartDate,
d_sources.hospitalname as sid, noteevents.sid as key_for_sid,
round(noteevents.noteid, 0) as noteid, round(noteevents.elemid,
0) as elemid, round(noteevents.txid, 0) as txid from
noteevents, d_patients, d_careunits, d_sources where
d_patients.pid=noteevents.pid and
d_careunits.cuid=noteevents.cuid and
d_sources.sid=noteevents.sid ;

create table resultevents (
  pid     numeric,
  resultid     numeric,
  chartTime     timestamp,
  chartDate     numeric,
  cgid     numeric,
  cuid     numeric,
  resulttext     varchar(1000),
  sourcetime     timestamp,
  firstresult     numeric,
  nextresult     numeric,
  status     varchar(20),
  sid     numeric,
  txid     numeric
);

create table view_for_resultevents
as select resultevents.oid as oid_for_resultevents,
d_patients.case_id, round(resultevents.pid, 0) as pid,
round(resultevents.resultid, 0) as resultid,
resultevents.chartTime as chartTime,
round(resultevents.chartDate, 0) as chartDate,
round(resultevents.cgid, 0) as cgid, d_careunits.unitname as
cuid, resultevents.cuid as key_for_cuid,
resultevents.resulttext as resulttext, resultevents.sourcetime
as sourcetime, round(resultevents.firstresult, 0) as
firstresult, round(resultevents.nextresult, 0) as nextresult,
resultevents.status as status, d_sources.hospitalname as sid,
resultevents.sid as key_for_sid, round(resultevents.txid, 0) as
txid from resultevents, d_patients, d_careunits, d_sources
where d_patients.pid=resultevents.pid and
d_careunits.cuid=resultevents.cuid and
d_sources.sid=resultevents.sid ;

create table totalbalevents (
  pid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  itemid     integer,
  realtime     timestamp,
  pervolume     numeric,
```

```
  cumvolume     numeric,
  accumperiod     varchar(20),
  approx     varchar(10),
  reset_p     integer,
  stopped     varchar(20),
  annotation     varchar(500),
  cgid     numeric,
  cuid     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  txid     numeric,
  elemid     numeric
);

create table view_for_totalbalevents
as select totalbalevents.oid as oid_for_totalbalevents,
d_patients.case_id, round(totalbalevents.pid, 0) as pid,
totalbalevents.charttime as charttime,
round(totalbalevents.chartdate, 0) as chartdate,
d_ioitems.label as itemid, totalbalevents.itemid as
key_for_itemid, totalbalevents.realtime as realtime,
round(totalbalevents.pervolume, 0) as pervolume,
round(totalbalevents.cumvolume, 0) as cumvolume,
totalbalevents.accumperiod as accumperiod,
totalbalevents.approx as approx, round(totalbalevents.reset_p,
0) as reset_p, totalbalevents.stopped as stopped,
totalbalevents.annotation as annotation,
round(totalbalevents.cgid, 0) as cgid, d_careunits.unitname as
cuid, totalbalevents.cuid as key_for_cuid,
round(totalbalevents.scode, 0) as scode,
round(totalbalevents.pcode, 0) as pcode, d_sources.hospitalname
as sid, totalbalevents.sid as key_for_sid,
round(totalbalevents.txid, 0) as txid,
round(totalbalevents.elemid, 0) as elemid from totalbalevents,
d_patients, d_ioitems, d_careunits, d_sources where
d_patients.pid=totalbalevents.pid and
d_ioitems.itemid=totalbalevents.itemid and
d_careunits.cuid=totalbalevents.cuid and
d_sources.sid=totalbalevents.sid ;

-- tables for IOEvents. Solutions, Additives, and Deliveries

create table ioevents (
  pid     numeric,
  charttime     timestamp,
  realtime     timestamp,
  itemid     numeric,
  altid     numeric,
  volume     numeric,
  volumeuom     varchar(20),
  unitshung     numeric,
  unitshunguom     varchar(20),
  newbottle     numeric,
  dressingchanged     numeric,
  tubingchanged     numeric,
  assessment     numeric,
  stopped     varchar(20),
  estimate     varchar(20),
  annotation     varchar(500),
  cgid     numeric,
  cuid     numeric,
  scode     numeric,
  pcode     numeric,
  chartdate     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);


create table view_for_ioevents
as select ioevents.oid as oid_for_ioevents, d_patients.case_id,
round(ioevents.pid, 0) as pid, ioevents.charttime as charttime,
ioevents.realtime as realtime, d_ioitems.label as itemid,
ioevents.itemid as key_for_itemid, d_ioitems0.label as altid,
ioevents.altid as key_for_altid, round(ioevents.volume, 0) as
volume, ioevents.volumeuom as volumeuom,
round(ioevents.unitshung, 0) as unitshung,
ioevents.unitshunguom as unitshunguom,
round(ioevents.newbottle, 0) as newbottle,
round(ioevents.dressingchanged, 0) as dressingchanged,
round(ioevents.tubingchanged, 0) as tubingchanged,
round(ioevents.assessment, 0) as assessment, ioevents.stopped
as stopped, ioevents.estimate as estimate, ioevents.annotation
as annotation, round(ioevents.cgid, 0) as cgid,
d_careunits.unitname as cuid, ioevents.cuid as key_for_cuid,
round(ioevents.scode, 0) as scode, round(ioevents.pcode, 0) as
pcode, round(ioevents.chartdate, 0) as chartdate,
d_sources.hospitalname as sid, ioevents.sid as key_for_sid,
round(ioevents.elemid, 0) as elemid, round(ioevents.txid, 0) as
txid from ioevents, d_patients, d_ioitems, d_ioitems
d_ioitems0, d_careunits, d_sources where
d_patients.pid=ioevents.pid and
```

```
d_ioitems.itemid=ioevents.itemid and
d_ioitems0.itemid=ioevents.altid and
d_careunits.cuid=ioevents.cuid and d_sources.sid=ioevents.sid ;

create table additives (
  pid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  itemid     numeric,
  ioitemid     numeric,
  amount     numeric,
  doseunits     varchar(20),
  mlperunit     numeric,
  route     varchar(20),
  cuid     numeric,
  cgid     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_additives
as select additives.oid as oid_for_additives,
d_patients.case_id, round(additives.pid, 0) as pid,
additives.charttime as charttime, round(additives.chartdate, 0)
as chartdate, d_meditems.label as itemid, additives.itemid as
key_for_itemid, d_ioitems0.label as ioitemid,
additives.ioitemid as key_for_ioitemid, round(additives.amount,
0) as amount, additives.doseunits as doseunits,
round(additives.mlperunit, 0) as mlperunit, additives.route as
route, d_careunits.unitname as cuid, additives.cuid as
key_for_cuid, round(additives.cgid, 0) as cgid,
round(additives.scode, 0) as scode, round(additives.pcode, 0)
as pcode, d_sources.hospitalname as sid, additives.sid as
key_for_sid, round(additives.elemid, 0) as elemid,
round(additives.txid, 0) as txid from additives, d_patients,
d_meditems, d_ioitems d_ioitems0, d_careunits, d_sources where
d_patients.pid=additives.pid and
d_meditems.itemid=additives.itemid and
d_ioitems0.itemid=additives.ioitemid and
d_careunits.cuid=additives.cuid and d_sources.sid=additives.sid
;

create table deliveries (
  pid     numeric,
  chartdate     numeric,
  charttime     timestamp,
  ioitemid     numeric,
  site     varchar(20),
  rate     numeric,
  cgid     numeric,
  cuid     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_deliveries as select deliveries.oid as
oid_for_deliveries, d_patients.case_id, round(deliveries.pid,
0) as pid, round(deliveries.chartdate, 0) as chartdate,
deliveries.charttime as charttime, d_ioitems.label as ioitemid,
deliveries.ioitemid as key_for_ioitemid, deliveries.site as
site, deliveries.rate as rate, round(deliveries.cgid, 0) as
cgid, d_careunits.unitname as cuid, deliveries.cuid as
key_for_cuid, d_sources.hospitalname as sid, deliveries.sid as
key_for_sid, round(deliveries.elemid, 0) as elemid,
round(deliveries.txid, 0) as txid from deliveries, d_patients,
d_ioitems, d_careunits, d_sources where
d_patients.pid=deliveries.pid and
d_ioitems.itemid=deliveries.ioitemid and
d_careunits.cuid=deliveries.cuid and
d_sources.sid=deliveries.sid ;

create table solutions (
  pid     numeric,
  charttime     timestamp,
  itemid     numeric,
  ioitemid     numeric,
  volume     numeric,
  doseunits     varchar(20),
  route     varchar(20),
  cgid     numeric,
  cuid     numeric,
  scode     numeric,
  pcode     numeric,
  chartdate     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);
```

```
create table view_for_solutions
 as select solutions.oid as oid_for_solutions,
d_patients.case_id, round(solutions.pid, 0) as pid,
solutions.charttime as charttime, d_meditems.label as itemid,
solutions.itemid as key_for_itemid, d_ioitems0.label as
ioitemid, solutions.ioitemid as key_for_ioitemid,
round(solutions.volume, 0) as volume, solutions.doseunits as
doseunits, solutions.route as route, round(solutions.cgid, 0)
as cgid, d_careunits.unitname as cuid, solutions.cuid as
key_for_cuid, round(solutions.scode, 0) as scode,
round(solutions.pcode, 0) as pcode, round(solutions.chartdate,
0) as chartdate, d_sources.hospitalname as sid, solutions.sid
as key_for_sid, round(solutions.elemid, 0) as elemid,
round(solutions.txid, 0) as txid from solutions, d_patients,
d_meditems, d_ioitems d_ioitems0, d_careunits, d_sources where
d_patients.pid=solutions.pid and
d_meditems.itemid=solutions.itemid and
d_ioitems0.itemid=solutions.ioitemid and
d_careunits.cuid=solutions.cuid and d_sources.sid=solutions.sid
;

-- tables for Care Plan/Pathway

create table problems (
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cgid     numeric,
  addcgid     numeric,
  cuid     numeric,
  startdate     timestamp,
  stopdate     timestamp,
  dateadded     numeric,
  problemnum     numeric,
  status     varchar(60),
  etiology     varchar(600),
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_problems
as select problems.oid as oid_for_problems, d_patients.case_id,
round(problems.pid, 0) as pid, d_problemitems.label as itemid,
problems.itemid as key_for_itemid, problems.charttime as
charttime, round(problems.chartdate, 0) as chartdate,
round(problems.cgid, 0) as cgid, round(problems.addcgid, 0) as
addcgid, d_careunits.unitname as cuid, problems.cuid as
key_for_cuid, problems.startdate as startdate,
problems.stopdate as stopdate, d_days.calDay as dateadded,
problems.dateadded as key_for_dateadded,
round(problems.problemnum, 2) as problemnum, problems.status as
status, problems.etiology as etiology, round(problems.scode, 0)
as scode, round(problems.pcode, 0) as pcode,
d_sources.hospitalname as sid, problems.sid as key_for_sid,
round(problems.elemid, 0) as elemid, round(problems.txid, 0) as
txid from problems, d_patients, d_problemitems, d_careunits,
d_days, d_sources where d_patients.pid=problems.pid and
d_problemitems.itemid=problems.itemid and
d_careunits.cuid=problems.cuid and
d_days.dayid=problems.dateadded and d_sources.sid=problems.sid
;

create table outcomes (
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cgid     numeric,
  cuid     numeric,
  comments     varchar(600),
  targetdate     numeric,
  dateadded     numeric,
  addcgid     numeric,
  evaltime     timestamp,
  evalcgid     numeric,
  shift     varchar(20),
  variancetype     varchar(20),
  variancecause     varchar(40),
  status     varchar(20),
  problem     numeric,
  probtime     timestamp,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_outcomes
```

```
as select outcomes.oid as oid_for_outcomes, d_patients.case_id,
round(outcomes.pid, 0) as pid, d_outcomeitems.label as itemid,
outcomes.itemid as key_for_itemid, outcomes.charttime as
charttime, round(outcomes.chartdate, 0) as chartdate,
round(outcomes.cgid, 0) as cgid, d_careunits.unitname as cuid,
outcomes.cuid as key_for_cuid, outcomes.comments as comments,
d_days.calDay as targetdate, outcomes.targetdate as
key_for_targetdate, d_days.calDay as dateadded,
outcomes.dateadded as key_for_dateadded, d_caregivers0.cgid as
addcgid, outcomes.addcgid as key_for_addcgid, outcomes.evaltime
as evaltime, d_caregivers1.cgid as evalcgid, outcomes.evalcgid
as key_for_evalcgid, outcomes.shift as shift,
outcomes.variancetype as variancetype, outcomes.variancecause
as variancecause, outcomes.status as status,
d_problemitems2.label as problem, outcomes.problem as
key_for_problem, outcomes.probtime as probtime,
round(outcomes.scode, 0) as scode, round(outcomes.pcode, 0) as
pcode, d_sources.hospitalname as sid, outcomes.sid as
key_for_sid, round(outcomes.elemid, 0) as elemid,
round(outcomes.txid, 0) as txid from outcomes, d_patients,
d_outcomeitems, d_careunits, d_days, d_caregivers
d_caregivers0, d_caregivers d_caregivers1, d_problemitems
d_problemitems2, d_sources where d_patients.pid=outcomes.pid
and d_outcomeitems.itemid=outcomes.itemid and
d_careunits.cuid=outcomes.cuid and
d_days.dayid=outcomes.targetdate and
d_days.dayid=outcomes.dateadded and
d_caregivers0.cgid=outcomes.addcgid and
d_caregivers1.cgid=outcomes.evalcgid and
d_problemitems2.itemid=outcomes.problem and
d_sources.sid=outcomes.sid ;

create table interventions (
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cgid     numeric,
  cuid     numeric,
  ordercgid     numeric,
  ordertime     timestamp,
  dateadded     timestamp,
  addcgid     numeric,
  targetdate     numeric,
  instructions     varchar(250),
  orderstatus     varchar(20),
  problem     numeric,
  probtime     timestamp,
  guidelinename     varchar(80),
  guideline     varchar(2000),
  chartstatus     varchar(20),
  shift     varchar(60),
  variancetype     varchar(20),
  variancecause     varchar(40),
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_interventions
as select interventions.oid as oid_for_interventions,
d_patients.case_id, round(interventions.pid, 0) as pid,
d_interventionitems.label as itemid, interventions.itemid as
key_for_itemid, interventions.charttime as charttime,
round(interventions.chartdate, 0) as chartdate,
round(interventions.cgid, 0) as cgid, d_careunits.unitname as
cuid, interventions.cuid as key_for_cuid,
round(interventions.ordercgid, 0) as ordercgid,
interventions.ordertime as ordertime, interventions.dateadded
as dateadded, round(interventions.addcgid, 0) as addcgid,
d_days.calDay as targetdate, interventions.targetdate as
key_for_targetdate, interventions.instructions as instructions,
interventions.orderstatus as orderstatus, d_problemitems0.label
as problem, interventions.problem as key_for_problem,
interventions.probtime as probtime, interventions.guidelinename
as guidelinename, interventions.guideline as guideline,
interventions.chartstatus as chartstatus, interventions.shift
as shift, interventions.variancetype as variancetype,
interventions.variancecause as variancecause,
round(interventions.scode, 0) as scode,
round(interventions.pcode, 0) as pcode, d_sources.hospitalname
as sid, interventions.sid as key_for_sid,
round(interventions.elemid, 0) as elemid,
round(interventions.txid, 0) as txid from interventions,
d_patients, d_interventionitems, d_careunits, d_days,
d_problemitems d_problemitems0, d_sources where
d_patients.pid=interventions.pid and
d_interventionitems.itemid=interventions.itemid and
d_careunits.cuid=interventions.cuid and
d_days.dayid=interventions.targetdate and
d_problemitems0.itemid=interventions.problem and
d_sources.sid=interventions.sid ;
```

```
-- tables for orders
create table driporders (
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cuid     numeric,
  verifiedtime     timestamp,
  verifiedby     numeric,
  addtime     timestamp,
  addby     numeric,
  addverifytime     timestamp,
  addverifyby     numeric,
  duration     numeric,
  durationtype     varchar(20),
  orderedby     varchar(30),
  starttime     timestamp,
  stoptime     timestamp,
  schedcomments     varchar(60),
  discontinuecomments     varchar(60),
  mdinstr     varchar(500),
  rninstr     varchar(500),
  phinstr     varchar(500),
  mdcosign     varchar(30),
  rnreview     varchar(30),
  phreview     varchar(30),
  freqlabel     varchar(16),
  action     varchar(20),
  state     varchar(20),
  stopstate     varchar(20),
  education     varchar(20),
  base     numeric,
  basevol     numeric,
  rate     numeric,
  dosemin     numeric,
  doseminuom     numeric,
  dosemax     numeric,
  dosemaxuom     numeric,
  doseunits     varchar(20),
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);
```

```
create table view_for_driporders
as select driporders.oid as oid_for_driporders,
d_patients.case_id, round(driporders.pid, 0) as pid,
d_meditems.label as itemid, driporders.itemid as
key_for_itemid, driporders.charttime as charttime,
round(driporders.chartdate, 0) as chartdate,
d_careunits.unitname as cuid, driporders.cuid as key_for_cuid,
driporders.verifiedtime as verifiedtime, d_caregivers.cgid as
verifiedby, driporders.verifiedby as key_for_verifiedby,
driporders.addtime as addtime, d_caregivers0.cgid as addby,
driporders.addby as key_for_addby, driporders.addverifytime as
addverifytime, d_caregivers1.cgid as addverifyby,
driporders.addverifyby as key_for_addverifyby,
round(driporders.duration, 0) as duration,
driporders.durationtype as durationtype, driporders.orderedby
as orderedby, driporders.starttime as starttime,
driporders.stoptime as stoptime, driporders.schedcomments as
schedcomments, driporders.discontinuecomments as
discontinuecomments, driporders.mdinstr as mdinstr,
driporders.rninstr as rninstr, driporders.phinstr as phinstr,
driporders.mdcosign as mdcosign, driporders.rnreview as
rnreview, driporders.phreview as phreview, driporders.freqlabel
as freqlabel, driporders.action as action, driporders.state as
state, driporders.stopstate as stopstate, driporders.education
as education, d_meditems2.label as base, driporders.base as
key_for_base, round(driporders.basevol, 2) as basevol,
round(driporders.rate, 2) as rate, round(driporders.dosemin, 2)
as dosemin, round(driporders.doseminuom, 2) as doseminuom,
round(driporders.dosemax, 2) as dosemax,
round(driporders.dosemaxuom, 2) as dosemaxuom,
driporders.doseunits as doseunits, round(driporders.scode, 0)
as scode, round(driporders.pcode, 0) as pcode,
d_sources.hospitalname as sid, driporders.sid as key_for_sid,
round(driporders.elemid, 0) as elemid, round(driporders.txid,
0) as txid from driporders, d_patients, d_meditems,
d_careunits, d_caregivers, d_caregivers d_caregivers0,
d_caregivers d_caregivers1, d_meditems d_meditems2, d_sources
where d_patients.pid=driporders.pid and
d_meditems.itemid=driporders.itemid and
d_careunits.cuid=driporders.cuid and
d_caregivers.cgid=driporders.verifiedby and
d_caregivers0.cgid=driporders.addby and
d_caregivers1.cgid=driporders.addverifyby and
d_meditems2.itemid=driporders.base and
d_sources.sid=driporders.sid ;

create table freeformorders (
```

```
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cuid     numeric,
  verifiedtime     timestamp,
  verifiedby     numeric,
  addtime     timestamp,
  addby     numeric,
  addverifytime     timestamp,
  addverifyby     numeric,
  duration     numeric,
  durationtype     varchar(20),
  orderedby     varchar(30),
  starttime     timestamp,
  stoptime     timestamp,
  schedcomments     varchar(60),
  discontinuecomments     varchar(60),
  mdinstr     varchar(500),
  rninstr     varchar(500),
  phinstr     varchar(500),
  mdcosign     varchar(30),
  rnreview     varchar(30),
  phreview     varchar(30),
  freqlabel     varchar(16),
  action     varchar(20),
  state     varchar(20),
  stopstate     varchar(20),
  education     varchar(20),
  order_     varchar(900),
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  txid     numeric
);

create table view_for_freeformorders
as select freeformorders.oid as oid_for_freeformorders,
d_patients.case_id, round(freeformorders.pid, 0) as pid,
d_chartitems.label as itemid, freeformorders.itemid as
key_for_itemid, freeformorders.charttime as charttime,
round(freeformorders.chartdate, 0) as chartdate,
d_careunits.unitname as cuid, freeformorders.cuid as
key_for_cuid, freeformorders.verifiedtime as verifiedtime,
d_caregivers.cgid as verifiedby, freeformorders.verifiedby as
key_for_verifiedby, freeformorders.addtime as addtime,
d_caregivers0.cgid as addby, freeformorders.addby as
key_for_addby, freeformorders.addverifytime as addverifytime,
d_caregivers1.cgid as addverifyby, freeformorders.addverifyby
as key_for_addverifyby, round(freeformorders.duration, 0) as
duration, freeformorders.durationtype as durationtype,
freeformorders.orderedby as orderedby, freeformorders.starttime
as starttime, freeformorders.stoptime as stoptime,
freeformorders.schedcomments as schedcomments,
freeformorders.discontinuecomments as discontinuecomments,
freeformorders.mdinstr as mdinstr, freeformorders.rninstr as
rninstr, freeformorders.phinstr as phinstr,
freeformorders.mdcosign as mdcosign, freeformorders.rnreview as
rnreview, freeformorders.phreview as phreview,
freeformorders.freqlabel as freqlabel, freeformorders.action as
action, freeformorders.state as state, freeformorders.stopstate
as stopstate, freeformorders.education as education,
freeformorders.order_ as order_, round(freeformorders.scode, 0)
as scode, round(freeformorders.pcode, 0) as pcode,
d_sources.hospitalname as sid, freeformorders.sid as
key_for_sid, round(freeformorders.txid, 0) as txid from
freeformorders, d_patients, d_chartitems, d_careunits,
d_caregivers, d_caregivers d_caregivers0, d_caregivers
d_caregivers1, d_sources where
d_patients.pid=freeformorders.pid and
d_chartitems.itemid=freeformorders.itemid and
d_careunits.cuid=freeformorders.cuid and
d_caregivers.cgid=freeformorders.verifiedby and
d_caregivers0.cgid=freeformorders.addby and
d_caregivers1.cgid=freeformorders.addverifyby and
d_sources.sid=freeformorders.sid ;

create table infusionorders (
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cuid     numeric,
  verifiedtime     timestamp,
  verifiedby     numeric,
  addtime     timestamp,
  addby     numeric,
  addverifytime     timestamp,
  addverifyby     numeric,
  duration     numeric,
  durationtype     varchar(20),
  orderedby     varchar(30),
  starttime     timestamp,
  stoptime     timestamp,
```

```
  schedcomments     varchar(60),
  discontinuecomments     varchar(60),
  mdinstr     varchar(500),
  rninstr     varchar(500),
  phinstr     varchar(500),
  mdcosign     varchar(30),
  rnreview     varchar(30),
  phreview     varchar(30),
  freqlabel     varchar(16),
  action     varchar(20),
  state     varchar(20),
  stopstate     varchar(20),
  education     varchar(20),
  base     numeric,
  basevol     numeric,
  rate     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_infusionorders
as select infusionorders.oid as oid_for_infusionorders,
d_patients.case_id, round(infusionorders.pid, 0) as pid,
d_chartitems.label as itemid, infusionorders.itemid as
key_for_itemid, infusionorders.charttime as charttime,
round(infusionorders.chartdate, 0) as chartdate,
d_careunits.unitname as cuid, infusionorders.cuid as
key_for_cuid, infusionorders.verifiedtime as verifiedtime,
d_caregivers.cgid as verifiedby, infusionorders.verifiedby as
key_for_verifiedby, infusionorders.addtime as addtime,
d_caregivers0.cgid as addby, infusionorders.addby as
key_for_addby, infusionorders.addverifytime as addverifytime,
d_caregivers1.cgid as addverifyby, infusionorders.addverifyby
as key_for_addverifyby, round(infusionorders.duration, 0) as
duration, infusionorders.durationtype as durationtype,
infusionorders.orderedby as orderedby, infusionorders.starttime
as starttime, infusionorders.stoptime as stoptime,
infusionorders.schedcomments as schedcomments,
infusionorders.discontinuecomments as discontinuecomments,
infusionorders.mdinstr as mdinstr, infusionorders.rninstr as
rninstr, infusionorders.phinstr as phinstr,
infusionorders.mdcosign as mdcosign, infusionorders.rnreview as
rnreview, infusionorders.phreview as phreview,
infusionorders.freqlabel as freqlabel, infusionorders.action as
action, infusionorders.state as state, infusionorders.stopstate
as stopstate, infusionorders.education as education,
d_meditems2.label as base, infusionorders.base as key_for_base,
round(infusionorders.basevol, 2) as basevol,
round(infusionorders.rate, 2) as rate,
round(infusionorders.scode, 0) as scode,
round(infusionorders.pcode, 0) as pcode, d_sources.hospitalname
as sid, infusionorders.sid as key_for_sid,
round(infusionorders.elemid, 0) as elemid,
round(infusionorders.txid, 0) as txid from infusionorders,
d_patients, d_chartitems, d_careunits, d_caregivers,
d_caregivers d_caregivers0, d_caregivers d_caregivers1,
d_meditems d_meditems2, d_sources where
d_patients.pid=infusionorders.pid and
d_chartitems.itemid=infusionorders.itemid and
d_careunits.cuid=infusionorders.cuid and
d_caregivers.cgid=infusionorders.verifiedby and
d_caregivers0.cgid=infusionorders.addby and
d_caregivers1.cgid=infusionorders.addverifyby and
d_meditems2.itemid=infusionorders.base and
d_sources.sid=infusionorders.sid ;

create table medorders (
  pid     numeric,
  itemid     numeric,
  charttime     timestamp,
  chartdate     numeric,
  cuid     numeric,
  verifiedtime     timestamp,
  verifiedby     numeric,
  addtime     timestamp,
  addby     numeric,
  addverifytime     timestamp,
  addverifyby     numeric,
  duration     numeric,
  durationtype     varchar(20),
  orderedby     varchar(30),
  starttime     timestamp,
  stoptime     timestamp,
  schedcomments     varchar(60),
  discontinuecomments     varchar(60),
  mdinstr     varchar(500),
  rninstr     varchar(500),
  phinstr     varchar(500),
  mdcosign     varchar(30),
  rnreview     varchar(30),
  phreview     varchar(30),
```

```
  freqlabel     varchar(16),
  action     varchar(20),
  state     varchar(20),
  stopstate     varchar(20),
  education     varchar(20),
  renewtime     timestamp,
  dosemin     numeric,
  doseminuom     numeric,
  dosemax     numeric,
  dosemaxuom     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric,
  txid     numeric
);

create table view_for_medorders
as select medorders.oid as oid_for_medorders,
d_patients.case_id, round(medorders.pid, 0) as pid,
d_ioitems.label as itemid, medorders.itemid as key_for_itemid,
medorders.charttime as charttime, round(medorders.chartdate, 0)
as chartdate, d_careunits.unitname as cuid, medorders.cuid as
key_for_cuid, medorders.verifiedtime as verifiedtime,
d_caregivers.cgid as verifiedby, medorders.verifiedby as
key_for_verifiedby, medorders.addtime as addtime,
d_caregivers0.cgid as addby, medorders.addby as key_for_addby,
medorders.addverifytime as addverifytime, d_caregivers1.cgid as
addverifyby, medorders.addverifyby as key_for_addverifyby,
round(medorders.duration, 0) as duration,
medorders.durationtype as durationtype, medorders.orderedby as
orderedby, medorders.starttime as starttime, medorders.stoptime
as stoptime, medorders.schedcomments as schedcomments,
medorders.discontinuecomments as discontinuecomments,
medorders.mdinstr as mdinstr, medorders.rninstr as rninstr,
medorders.phinstr as phinstr, medorders.mdcosign as mdcosign,
medorders.rnreview as rnreview, medorders.phreview as phreview,
medorders.freqlabel as freqlabel, medorders.action as action,
medorders.state as state, medorders.stopstate as stopstate,
medorders.education as education, medorders.renewtime as
renewtime, round(medorders.dosemin, 2) as dosemin,
round(medorders.doseminuom, 2) as doseminuom,
round(medorders.dosemax, 2) as dosemax,
round(medorders.dosemaxuom, 2) as dosemaxuom,
round(medorders.scode, 0) as scode, round(medorders.pcode, 0)
as pcode, d_sources.hospitalname as sid, medorders.sid as
key_for_sid, round(medorders.elemid, 0) as elemid,
round(medorders.txid, 0) as txid from medorders, d_patients,
d_ioitems, d_careunits, d_caregivers, d_caregivers
d_caregivers0, d_caregivers d_caregivers1, d_sources where
d_patients.pid=medorders.pid and
d_ioitems.itemid=medorders.itemid and
d_careunits.cuid=medorders.cuid and
d_caregivers.cgid=medorders.verifiedby and
d_caregivers0.cgid=medorders.addby and
d_caregivers1.cgid=medorders.addverifyby and
d_sources.sid=medorders.sid ;

-- tables for Duration

create table a_chartdurations (
  pid     numeric,
  starttime     timestamp,
  endtime     timestamp,
  itemid     numeric,
  cuid     numeric,
  duration     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric
);

create table view_for_a_chartdurations
as select a_chartdurations.oid as oid_for_a_chartdurations,
d_patients.case_id, round(a_chartdurations.pid, 0) as pid,
a_chartdurations.starttime as starttime,
a_chartdurations.endtime as endtime, d_chartitems.label as
itemid, a_chartdurations.itemid as key_for_itemid,
d_careunits.unitname as cuid, a_chartdurations.cuid as
key_for_cuid, round(a_chartdurations.duration, 2) as duration,
round(a_chartdurations.scode, 0) as scode,
round(a_chartdurations.pcode, 0) as pcode,
d_sources.hospitalname as sid, a_chartdurations.sid as
key_for_sid, round(a_chartdurations.elemid, 0) as elemid from
a_chartdurations, d_patients, d_chartitems, d_careunits,
d_sources where d_patients.pid=a_chartdurations.pid and
d_chartitems.itemid=a_chartdurations.itemid and
d_careunits.cuid=a_chartdurations.cuid and
d_sources.sid=a_chartdurations.sid ;

create table a_iodurations (
  pid     numeric,
  itemid     numeric,
```

```
  starttime     timestamp,
  endtime     timestamp,
  cuid     numeric,
  duration     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric
);

create table view_for_a_iodurations
as select a_iodurations.oid as oid_for_a_iodurations,
d_patients.case_id, round(a_iodurations.pid, 0) as pid,
d_ioitems.label as itemid, a_iodurations.itemid as
key_for_itemid, a_iodurations.starttime as starttime,
a_iodurations.endtime as endtime, d_careunits.unitname as cuid,
a_iodurations.cuid as key_for_cuid,
round(a_iodurations.duration, 2) as duration,
round(a_iodurations.scode, 0) as scode,
round(a_iodurations.pcode, 0) as pcode, d_sources.hospitalname
as sid, a_iodurations.sid as key_for_sid,
round(a_iodurations.elemid, 0) as elemid from a_iodurations,
d_patients, d_ioitems, d_careunits, d_sources where
d_patients.pid=a_iodurations.pid and
d_ioitems.itemid=a_iodurations.itemid and
d_careunits.cuid=a_iodurations.cuid and
d_sources.sid=a_iodurations.sid ;

create table a_meddurations (
  pid     numeric,
  startrealtime     timestamp,
  starttime     timestamp,
  itemid     numeric,
  endtime     timestamp,
  cuid     numeric,
  duration     numeric,
  scode     numeric,
  pcode     numeric,
  sid     numeric,
  elemid     numeric
);

create table view_for_a_meddurations
as select a_meddurations.oid as oid_for_a_meddurations,
d_patients.case_id, round(a_meddurations.pid, 0) as pid,
a_meddurations.startrealtime as startrealtime,
a_meddurations.starttime as starttime, d_meditems.label as
itemid, a_meddurations.itemid as key_for_itemid,
a_meddurations.endtime as endtime, d_careunits.unitname as
cuid, a_meddurations.cuid as key_for_cuid,
round(a_meddurations.duration, 2) as duration,
round(a_meddurations.scode, 0) as scode,
round(a_meddurations.pcode, 0) as pcode, d_sources.hospitalname
as sid, a_meddurations.sid as key_for_sid,
round(a_meddurations.elemid, 0) as elemid from a_meddurations,
d_patients, d_meditems, d_careunits, d_sources where
d_patients.pid=a_meddurations.pid and
d_meditems.itemid=a_meddurations.itemid and
d_careunits.cuid=a_meddurations.cuid and
d_sources.sid=a_meddurations.sid ;
```

# APPENDIX C: MIMIC Index Definitions

```
-- indexes for all tables in MIMIC

-- for metatdata
create index mimic_table_columns on mimic_table_elements (column_name);
create index mimic_table_tables on mimic_table_elements (table_name);
create index mimic_table_cats on mimic_table_categories (category_name);
create index mimic_table_index on mimic_table_elements (table_name, include_in_view_p);
create index mimic_display_keys_index on mimic_display_keys(table_name);
create index mimic_time_keys_index on mimic_time_keys(table_name);
create index mimic_element_view on mimic_table_elements (table_name, include_in_view_p);
create index mimic_element_search on mimic_table_elements (table_name,
include_in_ctx_index_p);
create index mimic_display_keys_index on mimic_display_keys (table_name);

-- for data log
create index mimic_update_id on new_data_log (update_id);

-- for patient fact tables
create index case_id_for_problems on d_patients (case_id);
create index key_index_for_problems on problems (pid);
create index key_index_for_resultevents on resultevents (pid);
create index key_index_for_a_meddurations on a_meddurations (pid);
create index key_index_for_a_chartdurations on a_chartdurations (pid);
create index key_index_for_d_patients on d_patients (pid);
create index key_index_for_a_iodurations on a_iodurations (pid);
create index key_index_for_chartevents on chartevents (pid);
create index key_index_for_deliveries on deliveries (pid);
create index key_index_for_driporders on driporders (pid);
create index key_index_for_formevents on formevents (pid);
create index key_index_for_freeformorders on freeformorders (pid);
create index key_index_for_infusionorders on infusionorders (pid);
create index key_index_for_interventions on interventions (pid);
create index key_index_for_ioevents on ioevents (pid);
create index key_index_for_medevents on medevents (pid);
create index key_index_for_medorders on medorders (pid);
create index key_index_for_outcomes on outcomes (pid);
create index key_index_for_censusevents on censusevents (pid);
create index key_index_for_solutions on solutions (pid);
create index key_index_for_additives on additives (pid);
create index key_index_for_noteevents on noteevents (pid);
create index key_index_for_totalbalevents on totalbalevents (pid);

-- for dimension tables

create index key_index_for_d_caregivers on d_caregivers (cgid);
create index key_index_for_d_careunits on d_careunits (cuid);
create index key_index_for_d_chartitems on d_chartitems (itemid);
create index key_index_for_d_days on d_days (dayid);
create index key_index_for_d_days on d_days (calday);
create index cal_key_index_for_d_days on d_days (calday);
create index key_index_for_d_interventionitems on d_interventionitems (itemid);
create index key_index_for_d_ioitems on d_ioitems (itemid);
create index key_index_for_d_meditems on d_meditems (itemid);
create index key_index_for_d_outcomeitems on d_outcomeitems (itemid);
create index key_index_for_d_primarycodes on d_primarycodes (pcode);
create index key_index_for_d_problemitems on d_problemitems (itemid);
create index key_index_for_d_secondarycodes on d_secondarycodes (scode);
create index key_index_for_d_sources on d_sources (sid);

-- for time elements
create index time_index_for_a_chartdurations on a_chartdurations (starttime);
create index time_index_for_additives on additives (charttime);
create index time_index_for_a_iodurations on a_iodurations (starttime);
create index time_index_for_a_meddurations on a_meddurations (starttime);
create index time_index_for_censusevents on censusevents (intime);
create index time_index_for_chartevents on chartevents (charttime);
```

```
create index time_index_for_deliveries on deliveries (charttime);
create index time_index_for_driporders on driporders (charttime);
create index time_index_for_formevents on formevents (charttime);
create index time_index_for_freeformorders on freeformorders (charttime);
create index time_index_for_infusionorders on infusionorders (charttime);
create index time_index_for_interventions on interventions (charttime);
create index time_index_for_ioevents on ioevents (charttime);
create index time_index_for_medevents on medevents (charttime);
create index time_index_for_medorders on medorders (charttime);
create index time_index_for_noteevents on noteevents (charttime);
create index time_index_for_outcomes on outcomes (charttime);
create index time_index_for_problems on problems (charttime);
create index time_index_for_resultevents on resultevents (chartTime);
create index time_index_for_solutions on solutions (charttime);
create index time_index_for_totalbalevents on totalbalevents (charttime);

-- time and key elements
create index time_key_index_a_chartdurations on a_chartdurations (starttime, pid);
create index time_key_index_additives on additives (charttime, pid);
create index time_key_index_a_iodurations on a_iodurations (starttime, pid);
create index time_key_index_a_meddurations on a_meddurations (starttime, pid);
create index time_key_index_censusevents on censusevents (intime, pid);
create index time_key_index_chartevents on chartevents (charttime, pid);
create index time_key_index_deliveries on deliveries (charttime, pid);
create index time_key_index_driporders on driporders c(harttime, pid);
create index time_key_index_formevents on formevents (charttime, pid);
create index time_key_index_freeformorders on freeformorders (charttime, pid);
create index time_key_index_infusionorders on infusionorders (charttime, pid);
create index time_key_index_interventions on interventions (charttime, pid);
create index time_key_index_ioevents on ioevents (charttime, pid);
create index time_key_index_medevents on medevents (charttime, pid);
create index time_key_index_medorders on medorders (charttime, pid);
create index time_key_index_noteevents on noteevents (charttime, pid);
create index time_key_index_outcomes on outcomes (charttime, pid);
create index time_key_index_problems on problems (charttime, pid);
create index time_key_index_resultevents on resultevents (chartTime, pid);
create index time_key_index_solutions on solutions (charttime, pid);
create index time_key_index_totalbalevents on totalbalevents (charttime, pid);

-- for search display
create index itemid_index_chartevents on chartevents (itemid);

-- for menu item display
create index menu_item_totalbalevents on totalbalevents(itemid);
create index menu_item_censusevents on censusevents (careunit);
create index menu_item_chartevents on chartevents (itemid);
create index menu_item_a_chartdurations on a_chartdurations (itemid);
create index menu_item_a_iodurations on a_iodurations (itemid);
create index menu_item_deliveries on deliveries (ioitemid);
create index menu_item_formevents on formevents (sectiontitle);
create index menu_item_ioevents on ioevents (itemid);
create index menu_item_medevents on medevents (itemid);
create index menu_item_noteevents on noteevents (category);
create index menu_item_solutions on solutions (itemid);
create index menu_item_totalbalevents on totalbalevents (itemid);

-- for date_keys
create index view_date_for_additives on view_for_additives (chartdate);
create index view_date_for_censusevents on view_for_censusevents (indayid);
create index view_date_for_chartevents on view_for_chartevents (chartdate);
create index view_date_for_deliveries on view_for_deliveries (chartdate);
create index view_date_for_driporders on view_for_driporders (chartdate);
create index view_date_for_formevents on view_for_formevents (chartDay);
create index view_date_for_freeformorders on view_for_freeformorders (chartdate);
create index view_date_for_infusionorders on view_for_infusionorders (chartdate);
create index view_date_for_interventions on view_for_interventions (chartdate);
create index view_date_for_ioevents on view_for_ioevents (chartdate);
create index view_date_for_medevents on view_for_medevents (chartdate);
create index view_date_for_medorders on view_for_medorders (chartdate);
create index view_date_for_noteevents on view_for_noteevents (chartDate);
create index view_date_for_outcomes on view_for_outcomes (chartdate);
```

```
create index view_date_for_problems on view_for_problems (chartdate);
create index view_date_for_resultevents on view_for_resultevents (chartDate);
create index view_date_for_solutions on view_for_solutions (chartdate);
create index view_date_for_totalbalevents on view_for_totalbalevents (chartdate);

-- for case_ids
create index case_id_for_chartevents on view_for_chartevents (case_id);
create index case_id_for_a_meddurations on view_for_a_meddurations (case_id);
create index case_id_for_a_iodurations on view_for_a_iodurations (case_id);
create index case_id_for_a_chartdurations on view_for_a_chartdurations (case_id);
create index case_id_for_interventions on view_for_interventions (case_id);
create index case_id_for_outcomes on view_for_outcomes (case_id);
create index case_id_for_problems on view_for_problems (case_id);
create index case_id_for_medorders on view_for_medorders (case_id);
create index case_id_for_infusionorders on view_for_infusionorders (case_id);
create index case_id_for_freeformorders on view_for_freeformorders (case_id);
create index case_id_for_driporders on view_for_driporders (case_id);
create index case_id_for_solutions on view_for_solutions (case_id);
create index case_id_for_deliveries on view_for_deliveries (case_id);
create index case_id_for_ioevents on view_for_ioevents (case_id);
create index case_id_for_noteevents on view_for_noteevents (case_id);
create index case_id_for_formevents on view_for_formevents (case_id);
create index case_id_for_censusevents on view_for_censusevents (case_id);
create index case_id_for_additives on view_for_additives (case_id);
create index case_id_for_resultevents on view_for_resultevents (case_id);
create index case_id_for_medevents on view_for_medevents (case_id);
create index case_id_for_totalbalevents on view_for_totalbalevents (case_id);
create index case_id_for_d_patients on view_for_d_patients (case_id);
```

# APPENDIX D: MIMIC Keys

**mimic_date_keys**
**"table_name"**
**"date_key"**

| | |
|---|---|
| additives | chartdate |
| censusevents | indayid |
| chartevents | chartdate |
| deliveries | chartdate |
| driporders | chartdate |
| formevents | chartDay |
| freeformorders | chartdate |
| infusionorders | chartdate |
| interventions | chartdate |
| totalbalevents | chartdate |
| ioevents | chartdate |
| medevents | chartdate |
| medorders | chartdate |
| noteevents | chartDate |
| outcomes | chartdate |
| resultevents | chartDate |
| problems | chartdate |
| solutions | chartdate |

**mimic_item_keys**
**"table_name"**
**"item_key"**
**"include_in_view_p"**

| | | |
|---|---|---|
| d_secondarycodes | code | f |
| d_sources | siteid | f |
| d_caregivers | cgid | f |
| d_careunits | cuid | f |
| d_interventionitems | itemid | f |
| d_ioitems | itemid | f |
| d_meditems | itemid | f |
| d_outcomeitems | itemid | f |
| d_problemitems | itemid | f |
| additives | itemid | f |
| a_meddurations | itemid | f |
| d_chartitems | itemid | f |
| d_days | dayid | f |
| d_patients | pid | f |
| d_primarycodes | code | f |
| driporders | itemid | f |
| freeformorders | itemid | f |
| infusionorders | itemid | f |
| interventions | itemid | f |
| medorders | itemid | f |
| outcomes | itemid | f |
| problems | itemid | f |
| resultevents | resultid | f |
| a_iodurations | itemid | t |
| deliveries | ioitemid | t |
| ioevents | itemid | t |
| medevents | itemid | t |
| solutions | itemid | t |
| totalbalevents | itemid | t |
| censusevents | careunit | f |
| formevents | sectiontitle | t |
| chartevents | itemid | f |
| a_chartdurations | itemid | f |
| noteevents | title | f |

**mimic_time_keys**
**"table_name"**
**"time_key"**

| | |
|---|---|
| d_days | month |
| resultevents | chartTime |
| deliveries | charttime |
| driporders | charttime |
| freeformorders | charttime |
| infusionorders | charttime |
| medorders | charttime |
| problems | charttime |
| outcomes | charttime |
| interventions | charttime |
| a_iodurations | starttime |
| a_meddurations | starttime |
| censusevents | intime |
| solutions | charttime |
| a_chartdurations | starttime |
| chartevents | charttime |
| formevents | charttime |
| ioevents | charttime |
| medevents | charttime |
| noteevents | charttime |
| totalbalevents | charttime |
| d_patients | pid |
| additives | charttime |

```
mimic_display_keys
"table_name"
"display_key"
d_patients             pid
 d_careunits           unitname
 d_chartitems          label
 d_days        calDay
 d_interventionitems label
 d_ioitems             label
 d_meditems            label
 d_outcomeitems        label
 d_problemitems        label
 d_sources             hospitalname
 d_caregivers          cgid
 d_primarycodes        pcode
 d_secondarycodes      scode
 censusevents          pid
 chartevents           itemid
 formevents            itemid
 medevents             itemid
 noteevents            pid
 resultevents          objectid
 totalbalevents        pid
 ioevents              itemid
 additives             itemid
 deliveries            ioitemid
 solutions             itemid
 driporders            itemid
 freeformorders        itemid
 infusionorders        itemid
 medorders             itemid
 problems              itemid
 outcomes              itemid
 interventions itemid
 a_chartdurations      itemid
 a_iodurations        | itemid
 a_meddurations       | itemid
(34 rows)
```

# Appendix E MIMIC Administrator's Manual

## Background on MIMIC

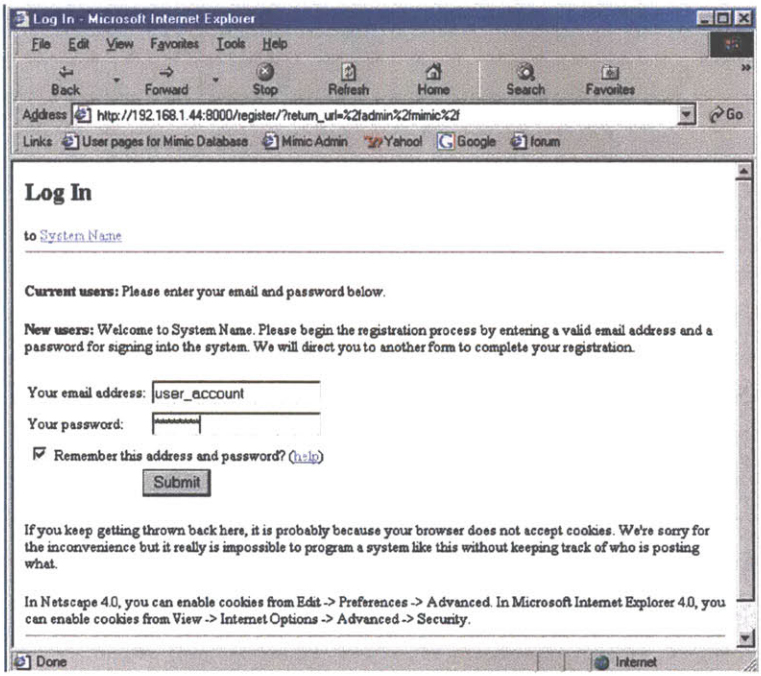
The Mimic Relational Database Management System (RDBMS) is designed for patient records taken from a hospital's clinical information system called CareVue. This automated system consolidates patient data and stores it in an Oracle database. Those records have been available to MIMIC through a partnership with a local hospital.

The purpose of the MIMIC RDBMS is to provide an interface to define tables and manage how they are accessed, displayed, and presented to the end user. The MIMIC RDBMS is used to select data to be downloaded from the partner hospital, parse, upload, and organize these records, and present them to the end user.

## Administrative Modules for MIMIC

Administrators to MIMIC have the ability to manage table definitions, add new data to the database, and control parameters that affect how data is displayed and managed.

Administrative tasks can be divided into two categories: making table definitions, and adding new data to the database. Table definitions need to be made only once, and updated as necessary. New data can be added at anytime. It is likely that there will be regular updates to add new data as it is collected.

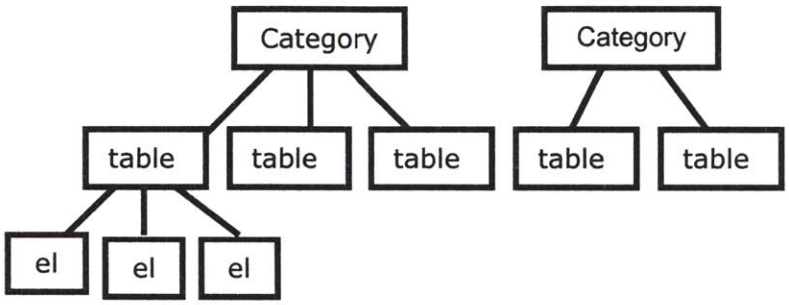



Administrators will need a username and password to log into MIMIC before being able to perform any of the administrative tasks described in this section.

## Database Management

The administrative database management module located in */server_path/www/admin/mimic/* allows administrators to enter table definitions for tables in MIMIC. These definitions are then used to generate SQL CREATE TABLE statements that the administrator can cut and paste into Postgres to create tables. The administrator does not need to have previous knowledge of SQL.

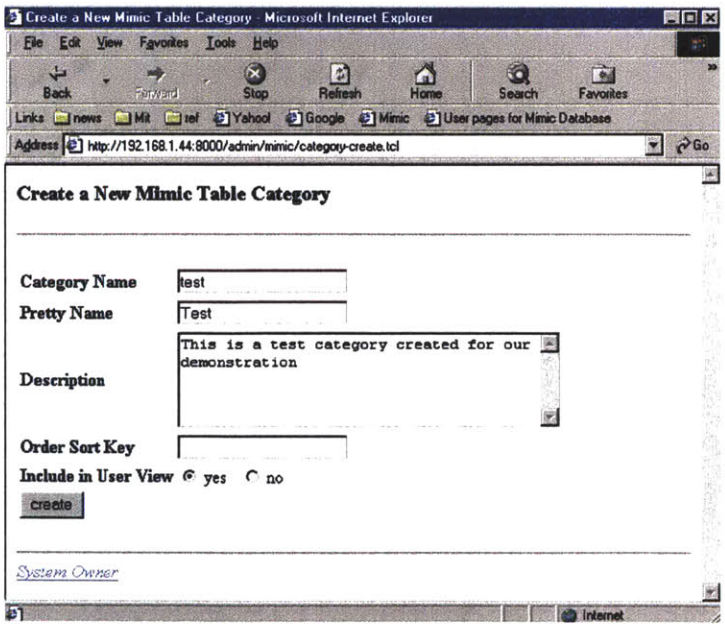
Tables are stored in categories and consist of a number of elements (columns) for that table.




*Hierarchy of Metadata*

## Step 1: Creating a Category

Tables are organized into categories. The section for normal users describes how category definitions affect how data is viewed. The administrator first needs to define a category for the table by clicking on "Create a New Category" link in */server_path/www/admin/mimic/.* The administrator can then enter the attributes associated with this category.




The **Category Name** is the name used in SQL for this category. Category names must be unique and should not contain any white space.

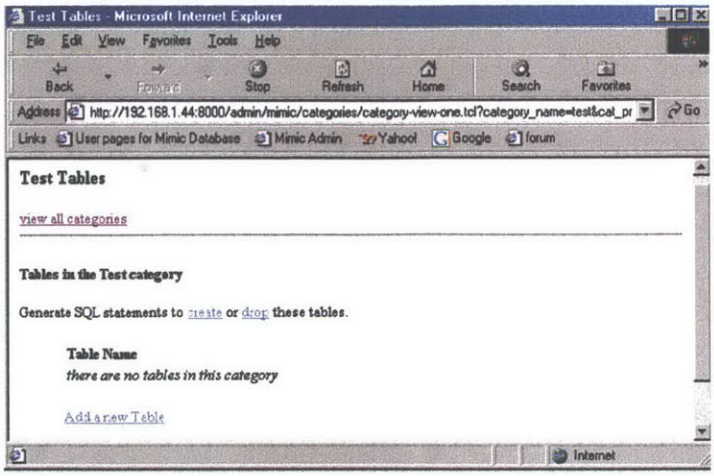
The **Pretty Name** is the name used when displaying this category name to normal users.

*(optional)* The **Description field** may be used to enter a short description about the usage or contents of this category.

The **Order Sort Key** is an integer used in sorting categories. For instance, a category with an Order Sort Key of 1 would appear first in

the list of categories. Order sort keys are constrained to be unique. The default value for order sort keys is the next largest value order sort key.

Lastly, the administrator may specify whether this category should be **Included in the User View**. This allows categories to be defined, but hidden from normal users. This is useful for administrative or dimension tables.



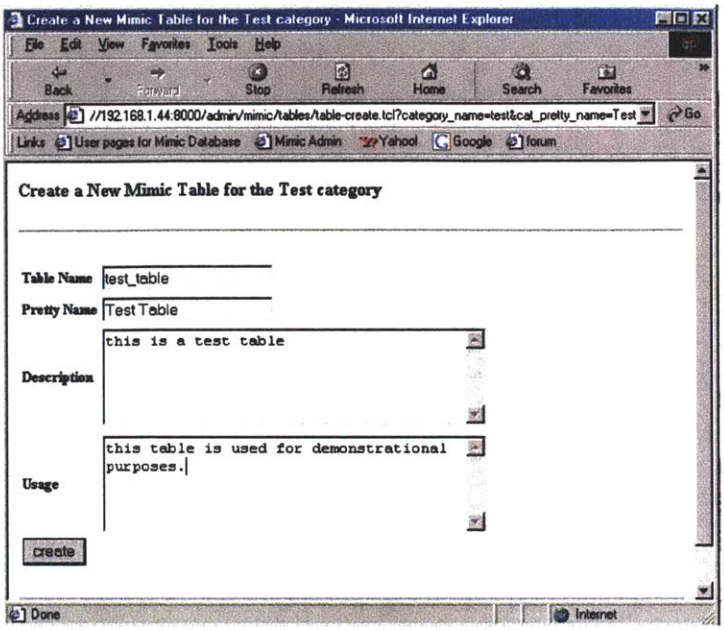

Once these attributes are entered, the administrator can create this category by clicking on the "*create*" button.

Once a category is created, the administrator is taken to a page that lists the tables for that category. To create a new table, click on "*Add a New Table*."

*Figure 1 New Category*

## Step 2: Creating a Table

Now that a category has been defined, tables can be added to that category. The attributes for tables are similar to those for categories.

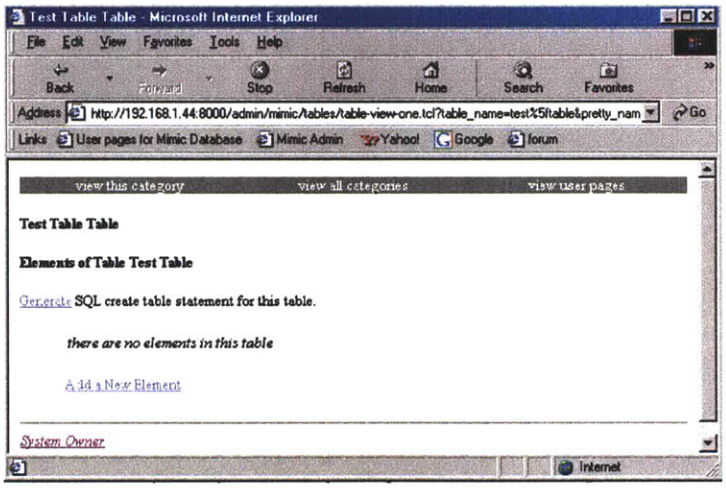



The **Table Name** is the name used in SQL for this table. The Table Name is constrained to be unique should not contain any white space.

The **Pretty Name** is the name used when displaying this table to normal users.

The **Description** field is used to store information about the table.

The **Usage** field is used to describe how this table should be used.




Once these attributes have been defined, the administrator can click on the "*create*" button to create this table. The table is not actually created on the database system at this time. Step 4 describes how to actually create the table in the database.

Now the administrator has defined a category and a table in that category. He can now add elements (columns) to that table by clicking on "*Add a New Element.*"

## Step 3: Adding Elements to a Table

Administrators can then enter attributes for an element in this table.

The **Column Name** is the name used in SQL for this table and should not contain any whitespace. The Column Name is constrained to be unique for this table.

The **Pretty Name** is the label used for displaying this column to normal users.

The **Abstract Data Type** specifies what type this column is, i.e. "boolean" or "integer" or "text." This value is used in formatting this column for display.

The **Postgres Data Type** is the type used for Postgres, i.e. "numeric" or "timestamp." This value is used in generating SQL to create and select from tables in Postgres.

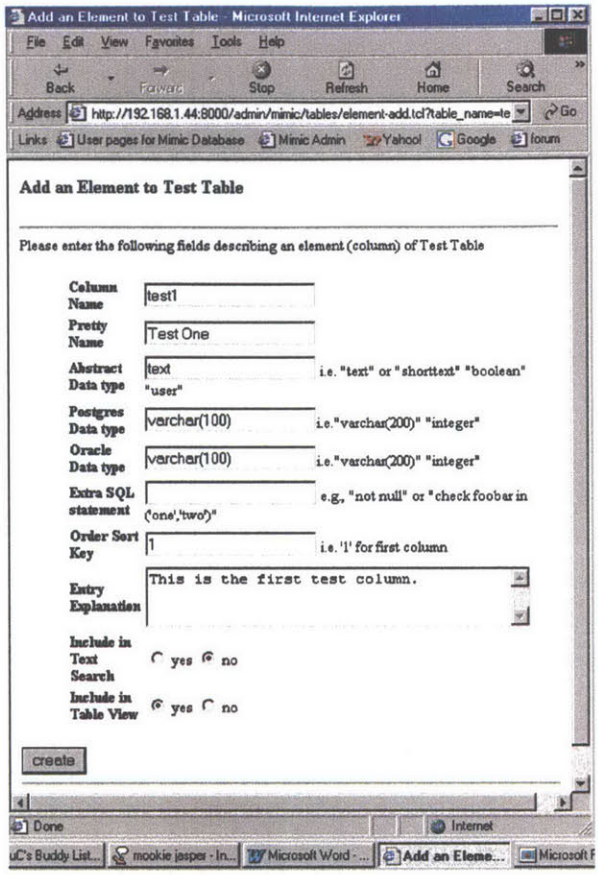
The **Oracle Data type** is the type used in Oracle, i.e. "integer" or "date." This value is used in selecting data from hospital tables that are stored in Oracle.

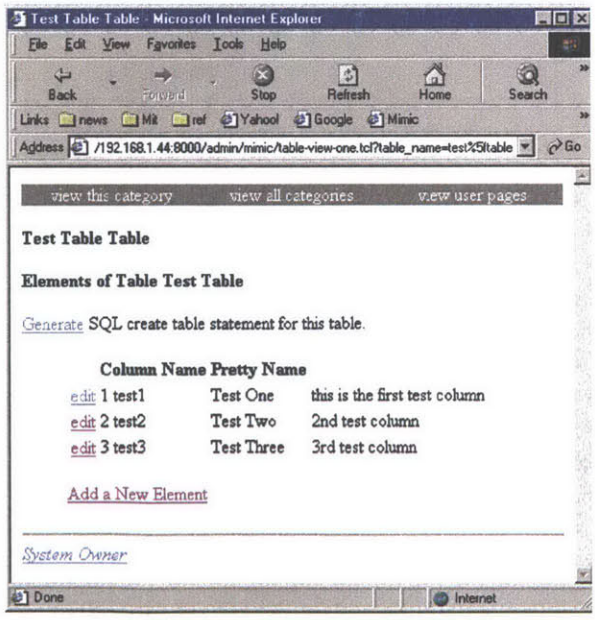
*(optional)* The **Extra SQL** field can be used for any extra SQL that should be included when creating this table, such as "not null" or "references d_tests.test."

The **Order Sort Key** is used to sort elements in this table. An element of order sort key of 1 would appear first in this table. The default value for this attribute is the maximum value +1 for this table.

*(optional)* The **Entry Explanation** field can be used to give a description of this column.

Administrators can specify whether this column should be included in a text



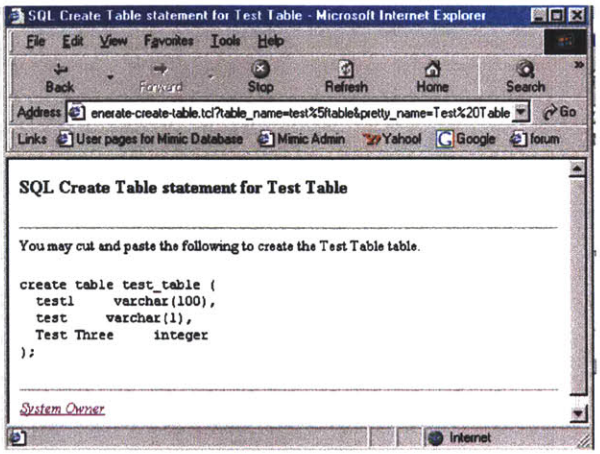
search of this table in the Include in Text Search field. Typically, this value is turned on for free text fields and turned off for fields that contain date or numeric information.

The **Include in Table View** field controls whether this field is visible to normal users. Some fields may not be significant to the end user and should be hidden.

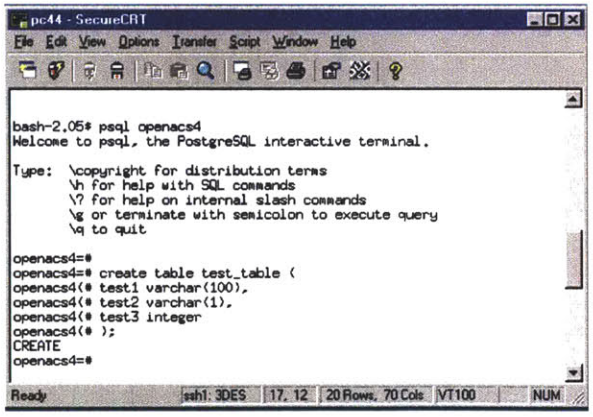
## Step 4: Generating SQL

Once all elements are defined for this table, the administrator can generate SQL to create this table by clicking on "Generate."

A code sample generated by this step appears below.




This will generate SQL create table statements that the administrator can cut and paste into the Postgres command line.




*Note: These administrative table management pages are also used to modify category, element, and table definitions. Each time an element's column name, Oracle data type, or Postgres data type is updated, the Administrator should first drop and then re-create the table in Postgres for the changes to take affect. Other modifications to columns take effect without having to recreate the table.*

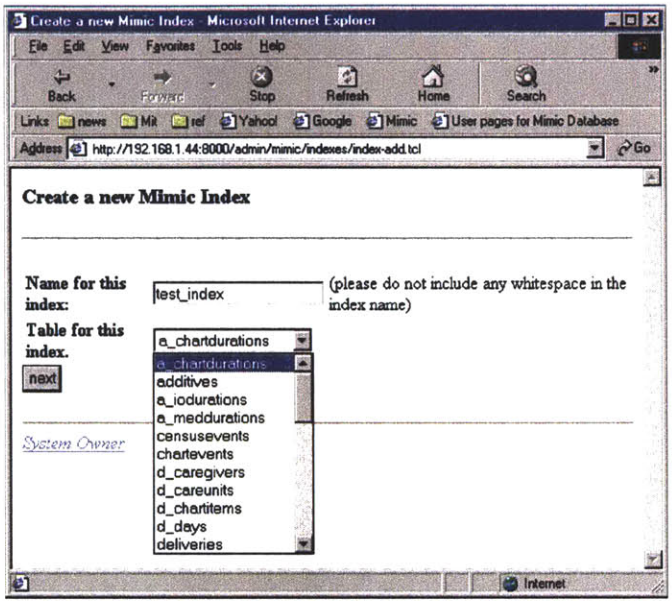
Currently, MIMIC contains table definitions for those tables defined in the CareVue ISM. A description of these table definitions appears in Appendix B.

## Index Management

SQL indexes are used to speed common queries for tables with indexable columns.

The index management module in */server_path/admin/mimic/indexes* allows users to create and manage indexes for tables defined in the table management module and created in Postgres as described in Step 4. Indexes are optional but can provide added performance.

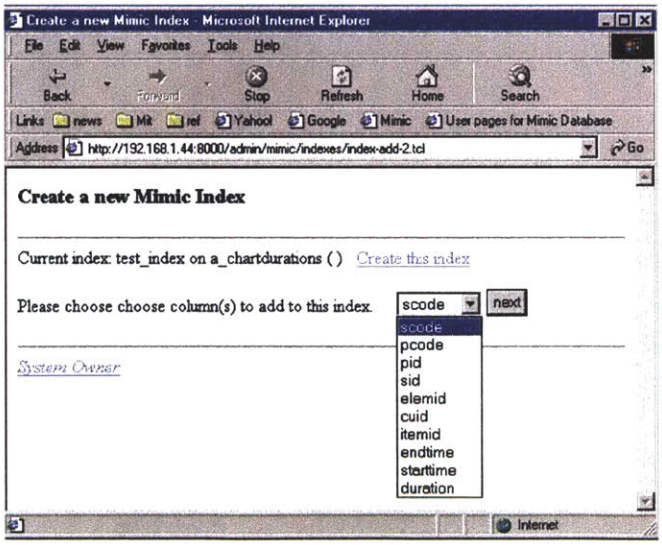
Administrators can create new indexes by clicking on "*Add a New Index*." He can then add attributes to define the index.




The **Name for the Index** is the name used in SQL for this index. It is constrained to be unique and should not contain any white space and.

The **Table for this Index** is the table that this index is created on. This can be selected from a pull-down menu of tables defined using the table management module. Once the index name and table have been chosen, click on "*next*."

The next step in creating a new index is to choose columns for the index. Columns can be chosen from a pull-down menu of previously defined columns (elements) for this table.




Clicking "*next*" will add a column to the index. Repeat to add more columns. Once all columns have been added, the Administrator can click on the link to "*Create this index*." The index will be added to the index management system and also created in the database.

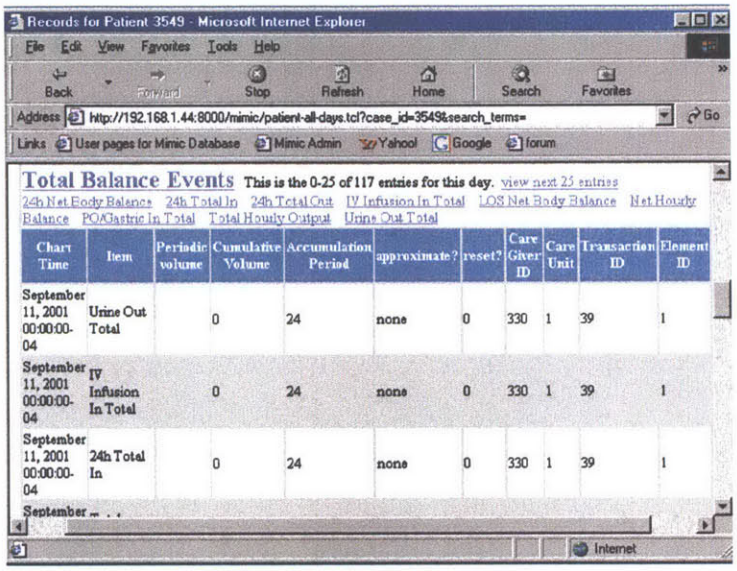
MIMIC contains indexes for the tables for tables that are currently defined. These indexes are described in more detail in Appendix C.

## Key Management

Keys store more information about how to display the data in tables. The key management module is in */server_path/admin/mimic/data/keys/*. These keys are simply columns of a table that are designated for a specific use.

A **Display Key** for a table is the column that should be used as a label for this table. In some views where only one column for a table is displayed, this label is used. For example, the `d_meditems` table, the `label` is the display key. Oftentimes other tables will reference the `d_meditems` table. When it does, the `label` is used to display that value.

**Menu Keys** are used in displaying data in a menu fashion. Administrators can specify a column for each table and whether this menu key should be visible. This controls the menus that are displayed to normal users. For instance, the `totalbalevents` table has `itemid` as its menu key. This means that the values of this column are available as menu options. One value for `itemid` is "24 Total Out." A user could click on this item to view only entries in that column with that menu value. For this example, the result would be a table of only "24 Total Out" entries.




**Time Keys** are used to select and order data. These are generally columns with timestamps. Each table is ordered by a time key so that results can be displayed in chronological order.

**Date Keys** are also used to select and order data. These are generally columns with integers that denote which day the item was stored. These date keys generally reference the d_days table, which has entries for every day from January 1, 1970 to December 30, 2030. Date keys are helpful in finding records for a specific day.

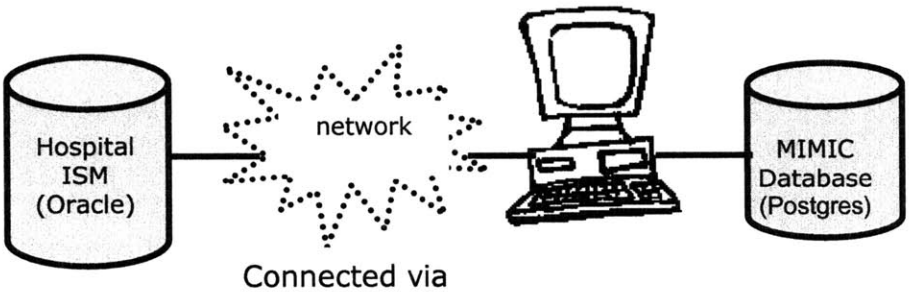
Current values for the keys described above can be found in Appendix D.

# Adding Data to MIMIC

Once tables have been defined in the table management module and created in Postgres, the system is ready to be populated with data. This section describes the steps involved in adding new data to MIMIC.

## *Overview*

The data comes from a hospital's Information Support Mart (ISM) that is stored in Oracle. Through a special partnership with the hospital, we are able to connect remotely to the hospital's network to download data.




Connected via
CareWeb

Currently, a separate system is used to initially collect and store waveform data. Patients are each assigned a case id (case_id). The hospital's ISM uses a different patient ID (pid) for each admission. Case_ids are assigned to be the same for each patient and reused for readmissions. The data download process for MIMIC consists of the following steps:

1. Find the hospital's patient ID for each patient by matching their names and/or MRN
2. Extract data with these patient ID's and de-identify records.
3. Upload data to the MIMIC database

## Tools Needed

- Web browser (i.e. Netscape or Internet Explorer)
- Access to hospital network (i.e. via CareWeb)
- Access to the hospital ISM (Oracle)
- Access to local server that MIMIC is on (pc44)
- List of patients
  This should be of the format:

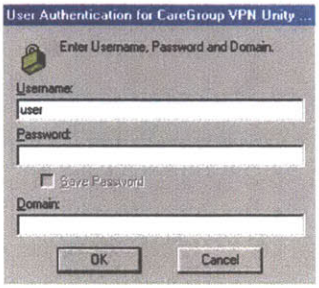
```
case_id|last_name|first_name|MRN
```

The MRN (Medical Record Number) is optional.

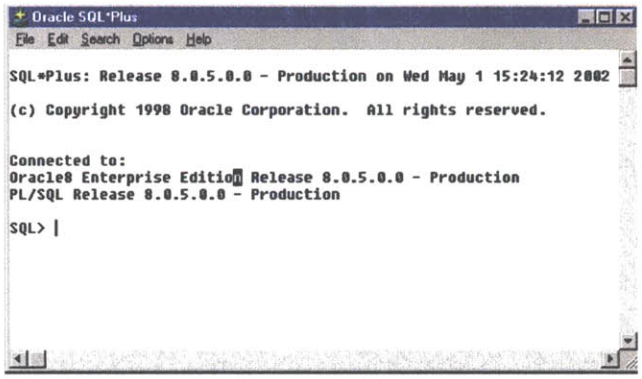
Samples of all files needed for and generated by the data update process can be found at the end of this document

## Getting New Data

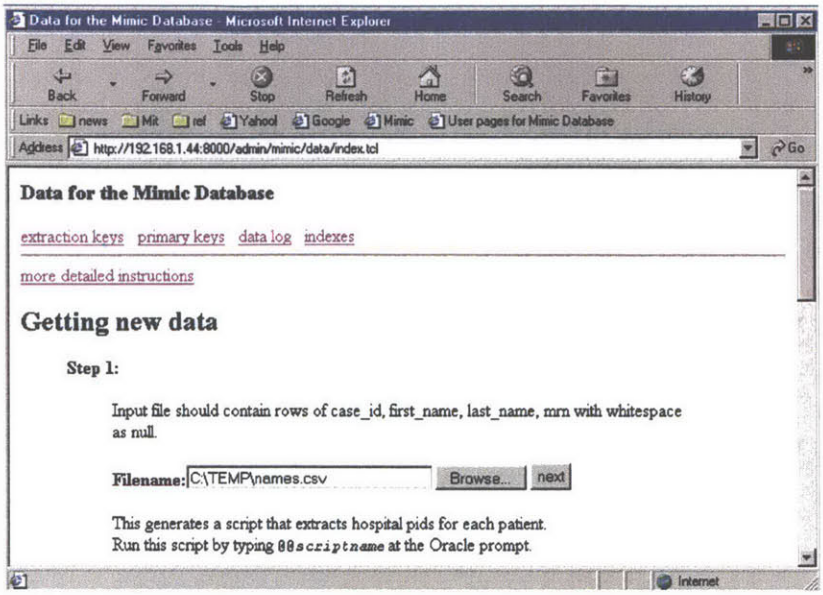
1. Connect to the CareGroup (eurkea.caregroup.org) system at the partner hospital via the CareGroup VPN Dialer.


2. Start Oracle SQL*Plus and connect to the hospital ISM. You should enter your username, password, and '**ISM**' as the host string. You are now connected to the hospital network and have access to patient records.



Go to the data upload page at */server_path/admin/mimic/data.*
Upload a file containing the list of patients for **Step #1**.
The list of patients should be a text file of the format:

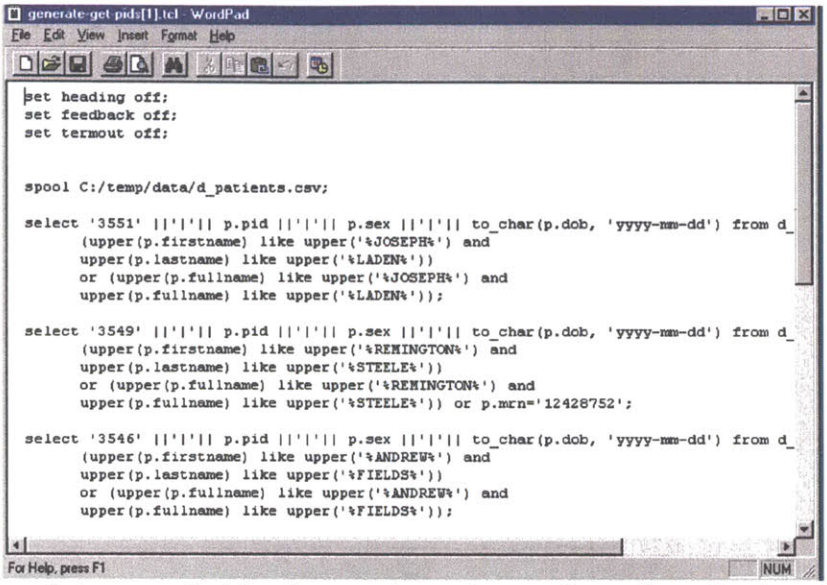



```
caseid|last_name|first_name|mrn
```

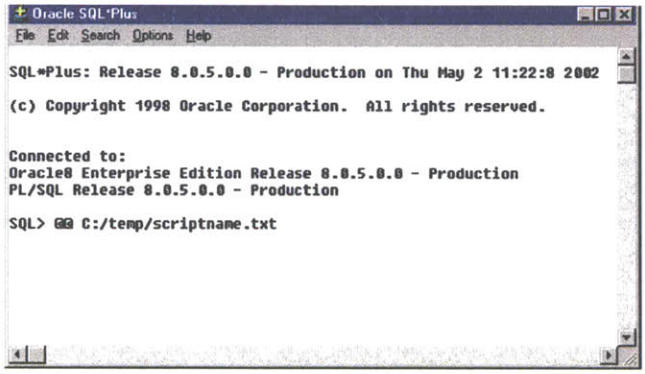
where each patient is listed on a separate line. A sample appears below.

```
3551|LADEN|JOSEPH|
3549|STEELE|REMINGTON|12428752
3546|FIELDS|ANDREW|
```

Here is a sample of a script generated by this step. Save this file as *scriptname.*




3. Take the script generated by the previous step and run it on the hospital system by typing: @@ *scriptname* at the Oracle prompt.




This step generates a file called d_patients.csv. This file will be of the format

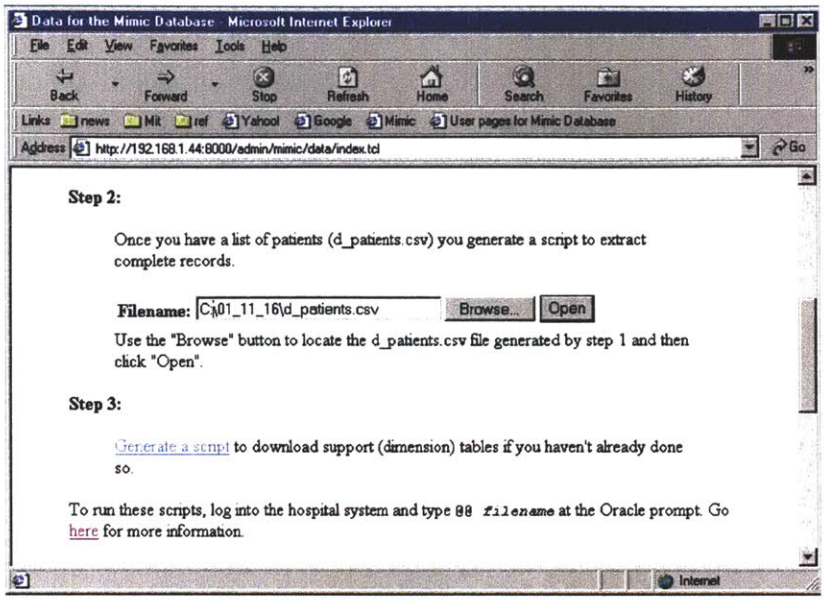
```
case_id|pid|sex|dob
```

A sample appears below:

```
3505|592|F|1936-09-09
3505|2316|F|1936-09-09
3509|1901|M|1927-07-05
```

No Patient names are used from this point forward.

4. Take the d_patients.csv file generated in the previous step and upload it in Step #2. This step generates a file that will extract complete patient records for the

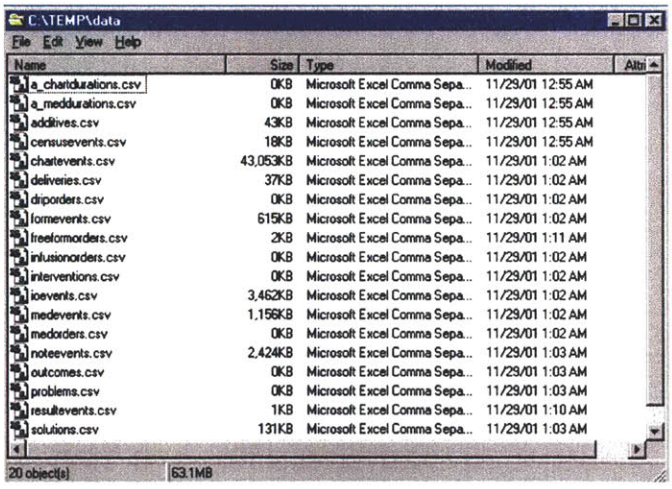



desired patients. Save this file as *scriptname.*

5. Run the script generated by the previous step by typing

```
@@ scriptname
```

at the Oracle prompt. This will create several .csv files in C:/temp/data on the local filesystem.




6. *(optional)* Mimic uses some support tables in storing patient data. If you have not already done so, you can generate a script to download support tables in Step #3 . Save this file as *scriptname.*

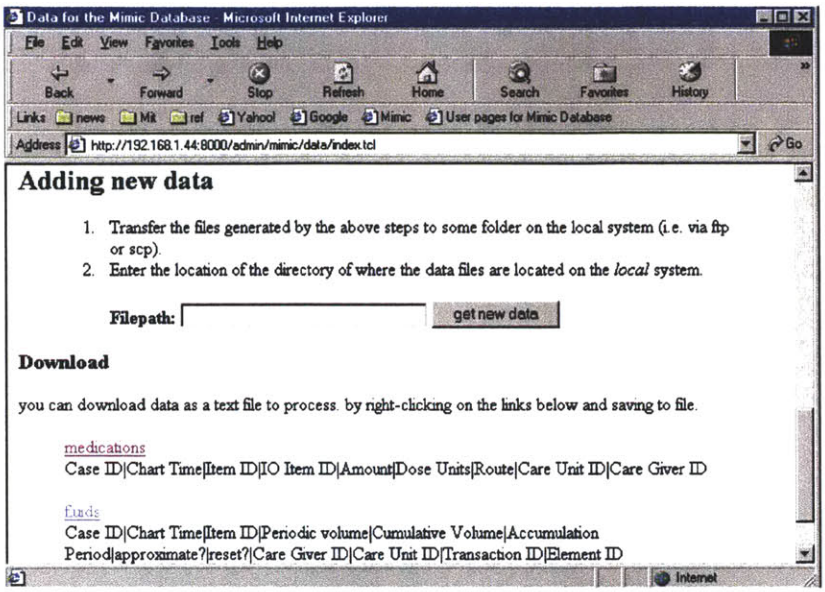
   Run this script by typing

```
@@ scriptname
```

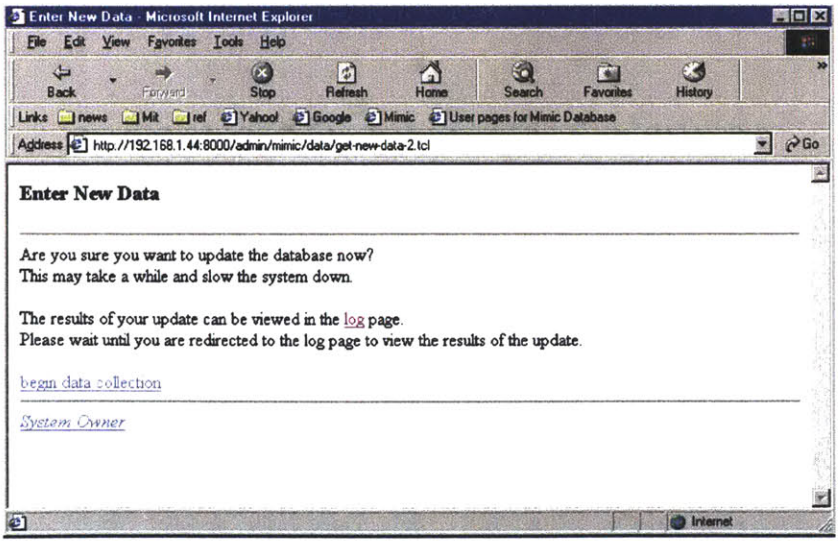
   at the Oracle prompt. This will generate more .csv files in C:/temp/data.

7. Move all .csv files generated by the previous steps to the local server (i.e. by ftp). Next, upload the data in Step 4 by specifying the location of the .csv files. Note that the files must be stored in a folder that is readable and writeable by the Postgres user..

   The next screen will ask for confirmation to begin the data upload process. The

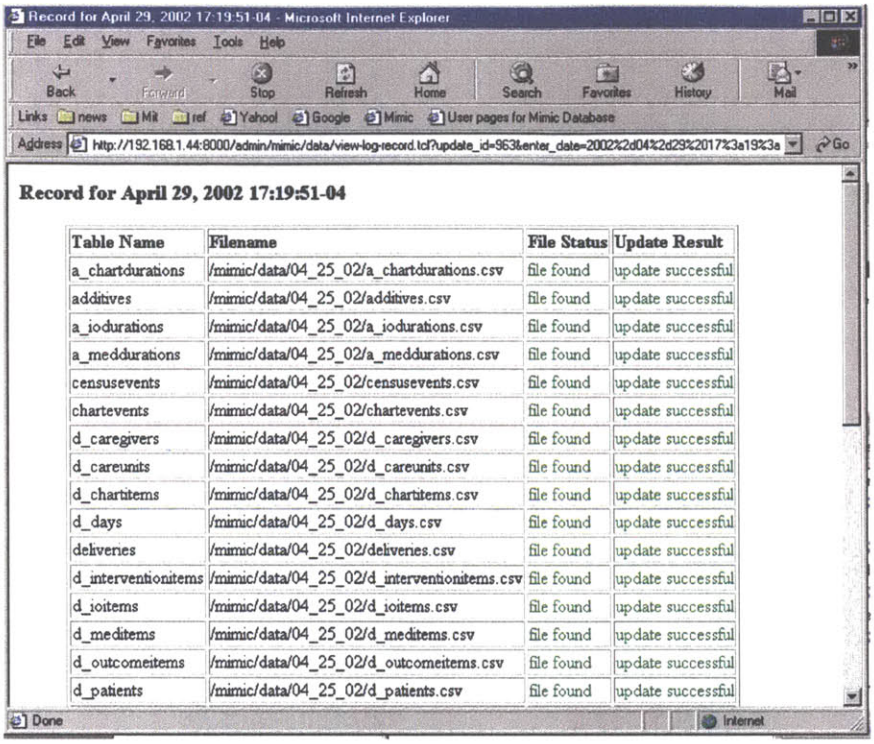



   time to upload new data depends on the number of records being downloaded and the size of each record.




   The data will then be parsed, de-identified, and entered into the database. Once this process is complete, the browser will be redirected to the data log where the results of the update can be viewed.

Normal users can now view these records in */server_path/mimic/.*



## *Sample Files*

### Sample input file of names for Step 1

```
1266|BROWN|JAMES|0033085
2106|THOMAS|SARAH|
```

### Sample Script generated by Step 1

```
set heading off;
set feedback off;
set termout off;


spool C:/temp/data/d_patients.csv;

select '1266' ||'|'|| p.pid ||'|'|| p.sex ||'|'|| to_char(p.dob,
'yyyy-mm-dd') from d_patients p, dual where
  (upper(p.firstname) like upper('%JAMES%') and
  upper(p.lastname) like upper('%BROWN%'))
  or (upper(p.fullname) like upper('%JAMES%') and
  upper(p.fullname) like upper('%BROWN%')) or p.mrn='0033085';

select '2106' ||'|'|| p.pid ||'|'|| p.sex ||'|'|| to_char(p.dob,
'yyyy-mm-dd') from d_patients p, dual where
  (upper(p.firstname) like upper('%SARAH%') and
  upper(p.lastname) like upper('%THOMAS%'))
  or (upper(p.fullname) like upper('%SARAH%') and
  upper(p.fullname) like upper('%THOMAS%'));

spool off;
```

### Sample d_patients.csv file generated by script from Step 1.

```
1266|1210|M|1912-11-05
2601|1172|F|1917-01-28
```

## Sample script generated by Step 2

```
set heading off;
set feedback off;
set termout off;

CREATE OR REPLACE FUNCTION replace_names
(find_pid in number,
 replace_string in char)
RETURN varchar
IS new_string varchar(5000);
temp_lastname varchar(100);
temp_firstname varchar(100);
temp_num number;

BEGIN
   SELECT fullname INTO temp_lastname from d_patients
WHERE pid =
find_pid;
   new_string := replace(lower(replace_string),
lower(temp_lastname),
'patient');
   temp_num:=instr(temp_lastname,' ');
   temp_firstname:=substr(temp_lastname, 0, temp_num);
   temp_lastname:=substr(temp_lastname, temp_num);
   new_string := replace(new_string, temp_firstname,
'patient ');
   new_string := replace(new_string, temp_lastname,
'patient ');

   RETURN (new_string);
END replace_names;
/

spool C:/temp/data/a_chartdurations.csv;

select pid||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(endtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   itemid||'|'||
   cuid||'|'||
   duration||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid
 from ism.a_chartdurations where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/additives.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   itemid||'|'||
   ioitemid||'|'||
   amount||'|'||
   replace_names(pid, doseunits)||'|'||
   mlperunit||'|'||
   replace_names(pid, route)||'|'||
   cuid||'|'||
   cgid||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.additives where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/a_iodurations.csv;

select pid||'|'||
   itemid||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(endtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   cuid||'|'||
   duration||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid
 from ism.a_iodurations where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/a_meddurations.csv;

select pid||'|'||
   to_char(startrealtime, 'YYYY-MM-DD
HH24:MI:SS')||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
```

```
   itemid||'|'||
   to_char(endtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   cuid||'|'||
   duration||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid
 from ism.a_meddurations where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/censusevents.csv;

select pid||'|'||
   to_char(intime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(outtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   careunit||'|'||
   destcareunit||'|'||
   replace_names(pid, dischstatus)||'|'||
   los||'|'||
   sid||'|'||
   indayid||'|'||
   outdayid
 from ism.censusevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/chartevents.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(realtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   itemid||'|'||
   replace_names(pid, value1)||'|'||
   value1num||'|'||
   replace_names(pid, value1uom)||'|'||
   replace_names(pid, value2)||'|'||
   value2num||'|'||
   replace_names(pid, value2uom)||'|'||
   replace_names(pid, stopped)||'|'||
   replace_names(pid, resultstatus)||'|'||
   replace_names(pid, annotation)||'|'||
   cgid||'|'||
   cuid||'|'||
   scode||'|'||
   pcode||'|'||
   chartdate||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.chartevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/deliveries.csv;

select pid||'|'||
   chartdate||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   ioitemid||'|'||
   replace_names(pid, site)||'|'||
   rate||'|'||
   cgid||'|'||
   cuid||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.deliveries where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/driporders.csv;

select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cuid||'|'||
   to_char(verifiedtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   verifiedby||'|'||
   to_char(addtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   addby||'|'||
   to_char(addverifytime, 'YYYY-MM-DD
HH24:MI:SS')||'|'||
   addverifyby||'|'||
   duration||'|'||
   replace_names(pid, durationtype)||'|'||
   replace_names(pid, orderedby)||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(stoptime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, schedcomments)||'|'||
   replace_names(pid, discontinuecomments)||'|'||
   replace_names(pid, mdinstr)||'|'||
```

```
   replace_names(pid, rninstr)||'|'||
   replace_names(pid, phinstr)||'|'||
   replace_names(pid, mdcosign)||'|'||
   replace_names(pid, rnreview)||'|'||
   replace_names(pid, phreview)||'|'||
   replace_names(pid, freqlabel)||'|'||
   replace_names(pid, action)||'|'||
   replace_names(pid, state)||'|'||
   replace_names(pid, stopstate)||'|'||
   replace_names(pid, education)||'|'||
   base||'|'||
   basevol||'|'||
   rate||'|'||
   dosemin||'|'||
   doseminuom||'|'||
   dosemax||'|'||
   dosemaxuom||'|'||
   replace_names(pid, doseunits)||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.driporders where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/formevents.csv;

select pid||'|'||
   to_char(chartTime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(realtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, formtitle)||'|'||
   replace_names(pid, sectiontitle)||'|'||
   replace_names(pid, subsectiontitle)||'|'||
   itemid||'|'||
   replace_names(pid, value_)||'|'||
   replace_names(pid, valuenum)||'|'||
   replace_names(pid, uom)||'|'||
   cgid||'|'||
   cuid||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid||'|'||
   chartDay||'|'||
   formid
 from ism.formevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/freeformorders.csv;

select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cuid||'|'||
   to_char(verifiedtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   verifiedby||'|'||
   to_char(addtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   addby||'|'||
   to_char(addverifytime, 'YYYY-MM-DD
HH24:MI:SS')||'|'||
   addverifyby||'|'||
   duration||'|'||
   replace_names(pid, durationtype)||'|'||
   replace_names(pid, orderedby)||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(stoptime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, schedcomments)||'|'||
   replace_names(pid, discontinuecomments)||'|'||
   replace_names(pid, mdinstr)||'|'||
   replace_names(pid, rninstr)||'|'||
   replace_names(pid, phinstr)||'|'||
   replace_names(pid, mdcosign)||'|'||
   replace_names(pid, rnreview)||'|'||
   replace_names(pid, phreview)||'|'||
   replace_names(pid, freqlabel)||'|'||
   replace_names(pid, action)||'|'||
   replace_names(pid, state)||'|'||
   replace_names(pid, stopstate)||'|'||
   replace_names(pid, education)||'|'||
   replace_names(pid, order_)||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   txid
 from ism.freeformorders where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/infusionorders.csv;
```

```
select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cuid||'|'||
   to_char(verifiedtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   verifiedby||'|'||
   to_char(addtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   addby||'|'||
   to_char(addverifytime, 'YYYY-MM-DD
HH24:MI:SS')||'|'||
   addverifyby||'|'||
   duration||'|'||
   replace_names(pid, durationtype)||'|'||
   replace_names(pid, orderedby)||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(stoptime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, schedcomments)||'|'||
   replace_names(pid, discontinuecomments)||'|'||
   replace_names(pid, mdinstr)||'|'||
   replace_names(pid, rninstr)||'|'||
   replace_names(pid, phinstr)||'|'||
   replace_names(pid, mdcosign)||'|'||
   replace_names(pid, rnreview)||'|'||
   replace_names(pid, phreview)||'|'||
   replace_names(pid, freqlabel)||'|'||
   replace_names(pid, action)||'|'||
   replace_names(pid, state)||'|'||
   replace_names(pid, stopstate)||'|'||
   replace_names(pid, education)||'|'||
   base||'|'||
   basevol||'|'||
   rate||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.infusionorders where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/interventions.csv;

select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cgid||'|'||
   cuid||'|'||
   ordercgid||'|'||
   to_char(ordertime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(dateadded, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   addcgid||'|'||
   targetdate||'|'||
   replace_names(pid, instructions)||'|'||
   replace_names(pid, orderstatus)||'|'||
   problem||'|'||
   to_char(probtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, guidelinename)||'|'||
   replace_names(pid, guideline)||'|'||
   replace_names(pid, chartstatus)||'|'||
   replace_names(pid, shift)||'|'||
   replace_names(pid, variancetype)||'|'||
   replace_names(pid, variancecause)||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.interventions where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/ioevents.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(realtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   itemid||'|'||
   altid||'|'||
   volume||'|'||
   replace_names(pid, volumeuom)||'|'||
   unitshung||'|'||
   replace_names(pid, unitshunguom)||'|'||
   newbottle||'|'||
   dressingchanged||'|'||
   tubingchanged||'|'||
   assessment||'|'||
   replace_names(pid, stopped)||'|'||
   replace_names(pid, estimate)||'|'||
   replace_names(pid, annotation)||'|'||
   cgid||'|'||
```

```
   cuid||'|'||
   scode||'|'||
   pcode||'|'||
   chartdate||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.ioevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/medevents.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   itemid||'|'||
   elemid||'|'||
   chartdate||'|'||
   to_char(realtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   volume||'|'||
   dose||'|'||
   replace_names(pid, doseuom)||'|'||
   solutionid||'|'||
   solvolume||'|'||
   replace_names(pid, route)||'|'||
   replace_names(pid, site)||'|'||
   replace_names(pid, stopped)||'|'||
   replace_names(pid, annotation)||'|'||
   cgid||'|'||
   cuid||'|'||
   pcode||'|'||
   scode||'|'||
   sid||'|'||
   txid
 from ism.medevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/medorders.csv;

select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cuid||'|'||
   to_char(verifiedtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   verifiedby||'|'||
```



```
   to_char(addtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   addby||'|'||
   to_char(addverifytime, 'YYYY-MM-DD
HH24:MI:SS')||'|'||
   addverifyby||'|'||
   duration||'|'||
   replace_names(pid, durationtype)||'|'||
   replace_names(pid, orderedby)||'|'||
   to_char(starttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(stoptime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, schedcomments)||'|'||
   replace_names(pid, discontinuecomments)||'|'||
   replace_names(pid, mdinstr)||'|'||
   replace_names(pid, rninstr)||'|'||
   replace_names(pid, phinstr)||'|'||
   replace_names(pid, mdcosign)||'|'||
   replace_names(pid, rnreview)||'|'||
   replace_names(pid, phreview)||'|'||
   replace_names(pid, freqlabel)||'|'||
   replace_names(pid, action)||'|'||
   replace_names(pid, state)||'|'||
   replace_names(pid, stopstate)||'|'||
   replace_names(pid, education)||'|'||
   to_char(renewtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   dosemin||'|'||
   doseminuom||'|'||
   dosemax||'|'||
   dosemaxuom||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.medorders where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/noteevents.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(realtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   replace_names(pid, category)||'|'||
   replace_names(pid, title)||'|'||
   replace_names(pid, notetext)||'|'||
   correction||'|'||
   cgid||'|'||
```

```
   cuid||'|'||
   chartDate||'|'||
   sid||'|'||
   noteid||'|'||
   elemid||'|'||
   txid
 from ism.noteevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/outcomes.csv;

select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cgid||'|'||
   cuid||'|'||
   replace_names(pid, comments)||'|'||
   targetdate||'|'||
   dateadded||'|'||
   addcgid||'|'||
   to_char(evaltime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   evalcgid||'|'||
   replace_names(pid, shift)||'|'||
   replace_names(pid, variancetype)||'|'||
   replace_names(pid, variancecause)||'|'||
   replace_names(pid, status)||'|'||
   problem||'|'||
   to_char(probtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.outcomes where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/problems.csv;

select pid||'|'||
   itemid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   cgid||'|'||
   addcgid||'|'||
   cuid||'|'||
   to_char(startdate, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   to_char(stopdate, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   dateadded||'|'||
   problemnum||'|'||
   replace_names(pid, status)||'|'||
   replace_names(pid, etiology)||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.problems where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/solutions.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   itemid||'|'||
   ioitemid||'|'||
   volume||'|'||
   replace_names(pid, doseunits)||'|'||
   replace_names(pid, route)||'|'||
   cgid||'|'||
   cuid||'|'||
   scode||'|'||
   pcode||'|'||
   chartdate||'|'||
   sid||'|'||
   elemid||'|'||
   txid
 from ism.solutions where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/totalbalevents.csv;

select pid||'|'||
   to_char(charttime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartdate||'|'||
   itemid||'|'||
   to_char(realtime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   pervolume||'|'||
   cumvolume||'|'||
   replace_names(pid, accumperiod)||'|'||
```

```
   replace_names(pid, approx)||'|'||
   reset||'|'||
   replace_names(pid, stopped)||'|'||
   replace_names(pid, annotation)||'|'||
   cgid||'|'||
   cuid||'|'||
   scode||'|'||
   pcode||'|'||
   sid||'|'||
   txid||'|'||
   elemid
 from ism.totalbalevents where  pid=1266 or pid=2106;

spool off;

spool C:/temp/data/resultevents.csv;

select pid||'|'||
   resultid||'|'||
   to_char(chartTime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   chartDate||'|'||
   cgid||'|'||
   cuid||'|'||
   replace_names(pid, resulttext)||'|'||
   to_char(sourcetime, 'YYYY-MM-DD HH24:MI:SS')||'|'||
   firstresult||'|'||
   nextresult||'|'||
   replace_names(pid, status)||'|'||
   sid||'|'||
   txid
 from ism.resultevents where  pid=1266 or pid=2106;

spool off;
```