PharmaSUG2010 - Paper AD15

Considerations for Building an Integrated Safety Database Using SAS

Denise J. Smith, Isis Pharmaceuticals, Carlsbad, CA
Daniel Schulz, Isis Pharmaceuticals, Carlsbad, CA
Gayle Kloss, Isis Pharmaceuticals, Carlsbad, CA
Wei Cheng, Isis Pharmaceuticals, Carlsbad, CA

ABSTRACT

As our company's pipeline of drugs, which all utilize the same unique technology, continues to expand, access to an Integrated Safety Database (ISDB) has become an invaluable resource. The ISDB includes all available clinical trial data collected across multiple compounds and trial phases. In practical terms, this means we have the ability to gain access to safety data across different protocols and subpopulations of patients. We have the ability to track single subjects through several studies, pool data for adverse event (AE) tracking, and gather lab results over longer durations of time than those shown in a single study. Using SAS®, we can quickly and easily access and analyze data from any of our nine core, harmonized datasets to create SAS datasets, SAS data views, tables, listings, and graphs. This paper will discuss the tools and procedures we used to build a solid, and continuously expanding, ISDB using SAS.

INTRODUCTION

As a compound moves through the various stages of clinical development, and more and more data are collected, SAS programmers can expect to analyze the safety data across various protocols and patient subpopulations. For certain indications, a company may even be interested in pooling data from different compounds. The slicing and dicing of the data can be a daunting task as the data are surely quite different between compounds, and between studies within a given compound. Thus, an ISDB is a rich, valuable and expandable resource, not only for the programming group but for other clinical functions as well.

In this paper we will refer to study level programming and integrated level programming. The former references work at the individual study level to prep and standardize the data so that it can be merged and utilized at the ISDB level. Dummy data was used for presentation purposes.

As of March, 2010, our ISDB contains data from six compounds, including 34 studies, 1,381 subjects, and 1,359,446 records. At Isis, we combine data from all trials and update the ISDB on a schedule of every two months. The following challenges had to be addressed before any programming work could begin:

- What core datasets do we want to create at the integrated level?
- What will the attributes of these datasets look like, and how should we document these attributes to ensure consistency between studies?
- How can we ensure that the attributes will be applied consistently across studies?
- Where should the datasets be created and how should we design the folder structure(s)?
- How should we handle the lab data?
- How can we identify and track those patients who participated in more than one of our clinical trials?
- How can we automate the process of updating data for ongoing studies?
- How should we track the status of data sets created?
- What about QC?
- How should we design the queries in terms of folder structure, output naming conventions?
- How often should we update the ISDB?

This paper will detail the actions taken to address the aforementioned challenges, including rules and tools established before any SAS programming was started and steps taken at the study level to harmonize the data and prepare for the integration. We will discuss the integration step(s), and touch briefly on the processes in place for creating summaries and queries. The most complex and interesting part of building the ISDB was working with the laboratory data, which contains a large number of unique lab tests coming to us from different laboratories and with different raw units attached. We've developed a solid work flow for processing this data, which we will discuss in this paper.

Our work is done using SAS 8.2 and 9.1 in a Windows® environment. We also utilize Microsoft Excel and Microsoft Word.

PRELIMINARY CONSIDERATIONS

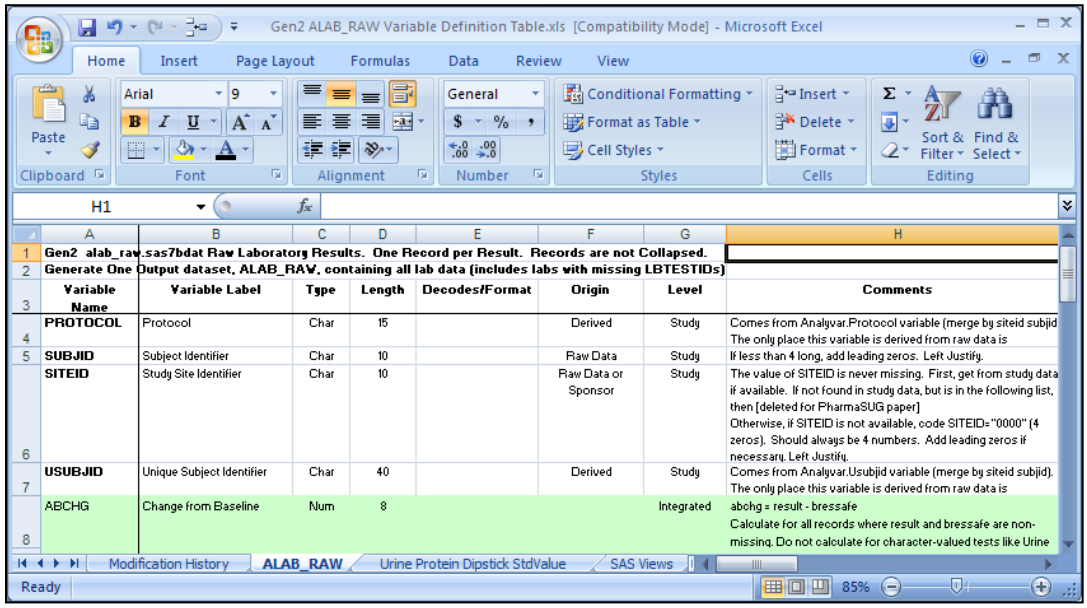
Based on our company's objectives we decided that we needed nine core datasets, including the following:

Dataset	Description
ANALYVAR	Baseline and demographic information
AALLAE	Adverse Events
ACONMEDS	Concomitant Medications
ADOSING	Dosing Information
AECG	Electrocardiogram Information
ALAB_RAW	Laboratory Tests
AMEDHX	Medical History
APE	Physical Examination
AVITALS	Vital Signs

The variables corresponding to the nine core datasets are defined in nine Excel spreadsheets called Variable Definition Tables. Each spreadsheet contains a tab with columns containing the variable name, label, type, length, decodes/format, origin (raw or derived), level (defined at the study level or the integrated level), and comments (any information to help in the derivation of the variable). There is an additional tab within each spreadsheet containing a modification history in the event that any variable attribute needs to be updated. For the laboratory dataset, ALAB_RAW, there is a third tab which includes documentation of the SAS Views created at the integrated level, and additional tabs are available for documenting standardized character result values for tests such as urine protein.

There are four key variables that exist in each of the datasets, and which are originally created in the ANALYVAR dataset. They are: USUBJID, PROTOCOL, SITEID, and SUBJID. The variable USUBJID (unique subject ID) guarantees a unique merge within the datasets at the integrated level and consists of a concatenation of protocol, site ID, and subject ID. For example, "123456-CS01 5001-1001" is the USUBJID for subject 1001, enrolled in protocol 123456-CS01 (study CS01 for compound 123456) at site 5001. At the integrated level, a variable called ISISID exists to help identify and track those subjects who enrolled in more than one of our trials, or were reused on the same trial. ISISID will be discussed later.

The following is a snapshot from the ALAB_RAW VDT:



The programmer follows the specifications in the VDT as he or she creates each of the nine datasets, but a method has been implemented to make certain that the attributes are applied directly and consistently for each of the studies, resulting in a seamless set of the 30+ studies at the integrated level. The following two macros are applied in the final step of each of the analysis dataset programs.

%MATTRIB(domain=XX): This macro call assigns the attributes for the "xx" dataset as provided in the VDT. The SAS code selected by the call begins with a dataset keep option, and includes all the required variable attributes for the dataset. Thus, the placement of this macro call is important. The SAS code generated by the MATTRIB macro call is as follows, using the AE dataset as an example:

```
(keep=AEENDT AEENDTM AENUM AESTDT AESTTM PROTOCOL SUBJID SITEID USUBJID AEACN AEBODSYS
AECOM1 AECOM2 AECOM3 AECOM4 AEDECOD AEENDTC AEFRQ AEOUT AEPRETRT AEREL AESER AESEV
AESTDTC AETERM AETOXGR AEVERS INJSITE SAE);
attrib
    PROTOCOL  label="Protocol"                              length=$15
    SUBJID    label="Subject Identifier"                    length=$10
    SITEID    label="Study Site Identifier"                 length=$10
    USUBJID   label="Unique Subject Identifier"             length=$40
    AEACN     label="Action Taken With Study Treatment"     length=$200
    AEBODSYS  label="Body System or Organ Class"            length=$200
    AECOM1    label="Comments 1"                            length=$200
    /*[code compressed]*/
```

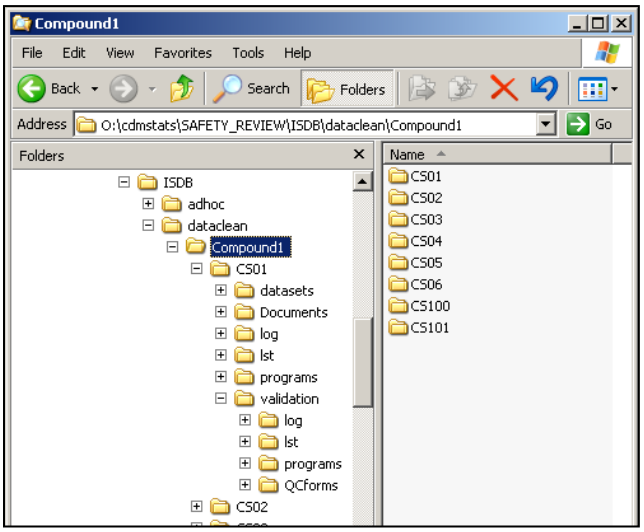
%MNMCH (domain=XX): This macro call performs some necessary "housekeeping" such as initializing all the required variables and taking care of the formats and informats for the variables. This call may be placed anywhere within the data step, as long as it is after the set statement. The SAS code generated by the MNMCH macro call is as follows, using the AE dataset as an example:

```
array nbrs AENUM AEENDT AEENDTM AESTDT AESTTM;
array chrs $ PROTOCOL SITEID SUBJID USUBJID AEACN AEBODYSY AEBODSYS AECOM1 AECOM2 AEDECOD
AEENDTC AEFRQ AEOUT AEPREF AEPREFV AEPRETRT
AEREL AESER AESEV AESTDTC AETERM AETOXGR AEVERS INJSITE SAE;

do i = 1 to dim(nbrs);
 if nbrs{i} = . then nbrs{i}=.;
end;
do i = 1 to dim(chrs);
 if chrs{i}='' then chrs{i}='';
end;

format _all_;
informat _all_;
format aestdt aeendt date9. aesttm aeendtm time5.;
```

The macros are applied during the final step in the analysis dataset creation program.

```
data datadir.AALLAE %mattrib(domain=AE);
set aefinal;
/*[code compressed]*/
%mnmch(domain=AE);
run;
```

The first step towards creating nine final, integrated datasets is to create the nine datasets for each individual study, following the specifications in the VDTs. All study-level programming is done under a subdirectory of the ISDB folder, called "dataclean." Each compound has its own directory, and each study has its own subdirectory containing folders for programming, validation, and the final analysis datasets.



PROGRAMMING THE ANALYSIS DATASETS FOR EACH STUDY

The creation of the analysis datasets is straightforward, but with room for creativity and personal programming style.

The VDT specifications should be followed closely to create the variables of interest and the MNMCH and MATTRIB macros should be called as the last step of each program to guarantee that all of the datasets have the same number of variables and the same variable attributes. The programmer is expected to output log files for each program. Outside of those 'rules', different programming styles and techniques are acceptable and expected as raw data can be quite different between compounds and studies. The QC programmer also follows the VDT. A log check macro is used to automatically check the SAS logs for warnings and errors.

Quick tip: Creating a SAS shortcut icon in each of your program directories makes life a little easier, especially when there is more than one programmer assigned to a project. The SAS shortcut should be set up to call a setup.sas (or something similar) in the same directory. The setup.sas program will include LIBNAME statements and some global macro variables if applicable to the study.

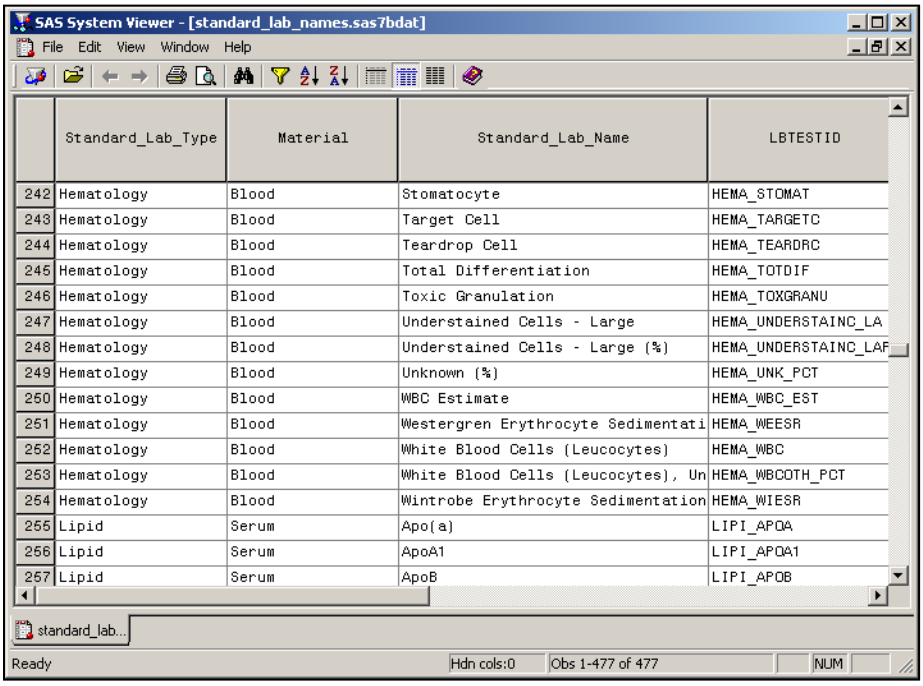
LABORATORY DATA

The most time consuming analysis dataset to build is ALAB_RAW. The laboratory tests collected differ from study to study, and sometimes more than one lab is used within a study. This means that different units may be attached to a given lab test across studies, or even within a study. Not to mention that naming conventions and other variable attributes can and will be quite different.

Because of the complexity of this data, we have implemented a number of rigid steps to ensure that the lab data are useable and the results standardized. The steps rely on two very important, internally maintained data files.

The first is a standard lab names file which contains standard lab names and their unique lab test ID (LBTESTID). The file is maintained by the programming group and periodically reviewed by a company clinician for accuracy. Since naming conventions vary between studies, the idea is to map all instances of a lab test to the same standard lab type and name. Lab types include: Chemistry, Hematology, and others. Lab test examples include Red Blood Cell Count, Cholesterol, and many others. As an illustration, let's say you have Hematology test names of "Lymphocyte (Absolute)"," Differential – Lymphocytes", "Lymphocytes". All three of these would map to the standard name "Absolute Lymphocyte Count".

A snapshot of the standard_lab_names.sas7bdat file follows:



There are three basic components to assigning LBTESTID values. The three parts are concatenated and separated by '_'. Only the third part may be missing, in which case there would not be a trailing '_'. LBTESTID is a character variable with a maximum length of 30.

Part 1: The lab type (max $5), which may not be missing.

Full Name	SAS ABBREVIATION
Antibody	ANTI
Chemistry	CHEM
Coagulation	COAG
(etc …)	

Part 2: The Core Test Name (max $15), which may not be missing. Source: Standard clinical abbreviation or Biometrics defined abbreviation.

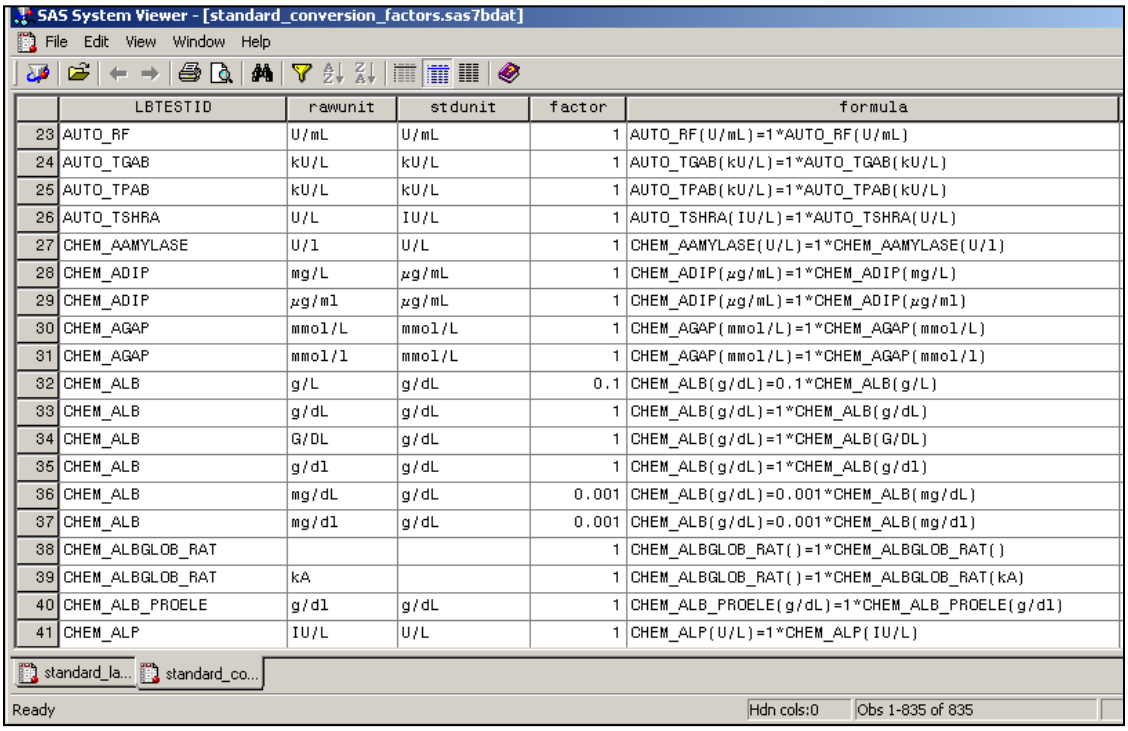
Full Name	CORE ABBREVIATION
Albumin	ALB
Blood Urea Nitrogen	BUN
Calcium	CA
(etc …)	

Part 3: Suffixes (max $8), which may be missing. Suffixes give supplemental information about the tests and help to maintain distinct LBTESTIDs.

Full Name	SUFFIX ABBREVIATION
Percent	PCT
Absolute Count	AB
Absolute Count Manual	AM
Total	TOT
(etc …)	

Following from the last example, the values which mapped to “Absolute Lymphocyte Count” are assigned a LBTESTID of HEMA_LYM_AB. The suffix “AB” distinguishes the lab test from HEMA_LYM_PCT, which maps to “Lymphocytes (%)”. We also use the suffix to distinguish between local and central laboratories for certain tests. We will talk more about assigning lab test units.

The second central file is the standard conversion factors dataset. It contains multiple observations per standard lab tests to represent all raw units found at the study levels, the preferred standard unit, and the appropriate factor for the conversion.



	LBTESTID	rawunit	stdunit	factor	formula
23	AUTO_RF	U/mL	U/mL	1	AUTO_RF(U/mL)=1*AUTO_RF(U/mL)
24	AUTO_TGAB	kU/L	kU/L	1	AUTO_TGAB(kU/L)=1*AUTO_TGAB(kU/L)
25	AUTO_TPAB	kU/L	kU/L	1	AUTO_TPAB(kU/L)=1*AUTO_TPAB(kU/L)
26	AUTO_TSHRA	U/L	IU/L	1	AUTO_TSHRA(IU/L)=1*AUTO_TSHRA(U/L)
27	CHEM_AAMYLASE	U/l	U/L	1	CHEM_AAMYLASE(U/L)=1*CHEM_AAMYLASE(U/l)
28	CHEM_ADIP	mg/L	µg/mL	1	CHEM_ADIP(µg/mL)=1*CHEM_ADIP(mg/L)
29	CHEM_ADIP	µg/ml	µg/mL	1	CHEM_ADIP(µg/mL)=1*CHEM_ADIP(µg/ml)
30	CHEM_AGAP	mmol/L	mmol/L	1	CHEM_AGAP(mmol/L)=1*CHEM_AGAP(mmol/L)
31	CHEM_AGAP	mmol/l	mmol/L	1	CHEM_AGAP(mmol/L)=1*CHEM_AGAP(mmol/l)
32	CHEM_ALB	g/L	g/dL	0.1	CHEM_ALB(g/dL)=0.1*CHEM_ALB(g/L)
33	CHEM_ALB	g/dL	g/dL	1	CHEM_ALB(g/dL)=1*CHEM_ALB(g/dL)
34	CHEM_ALB	G/DL	g/dL	1	CHEM_ALB(g/dL)=1*CHEM_ALB(G/DL)
35	CHEM_ALB	g/dl	g/dL	1	CHEM_ALB(g/dL)=1*CHEM_ALB(g/dl)
36	CHEM_ALB	mg/dL	g/dL	0.001	CHEM_ALB(g/dL)=0.001*CHEM_ALB(mg/dL)
37	CHEM_ALB	mg/dl	g/dL	0.001	CHEM_ALB(g/dL)=0.001*CHEM_ALB(mg/dl)
38	CHEM_ALBGLOB_RAT			1	CHEM_ALBGLOB_RAT()=1*CHEM_ALBGLOB_RAT()
39	CHEM_ALBGLOB_RAT	kA		1	CHEM_ALBGLOB_RAT()=1*CHEM_ALBGLOB_RAT(kA)
40	CHEM_ALB_PROELE	g/dl	g/dL	1	CHEM_ALB_PROELE(g/dL)=1*CHEM_ALB_PROELE(g/dl)
41	CHEM_ALP	IU/L	U/L	1	CHEM_ALP(U/L)=1*CHEM_ALP(IU/L)

Since there are always studies being added, we are continuously reviewing the data for new raw lab tests and/or units. Each time we find a new one, we consult and confirm with the team clinician. It is possible that at some point, there will be raw tests without a LBTESTID attached, or that are not yet converted to standard units. This is expected as the review process

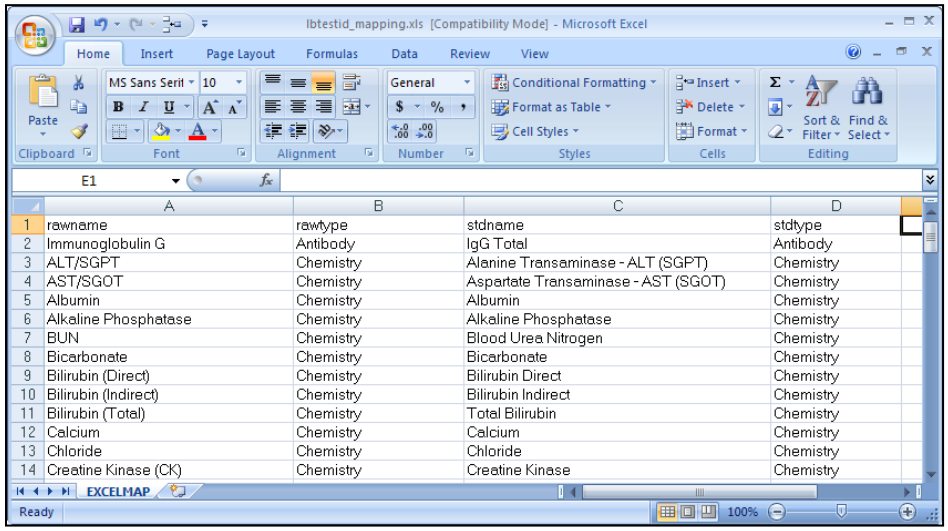
continues. Handling and the storage of these cases at the integrated level will be discussed later.

The following steps are followed to build a standard and useable lab file:

1. Read in raw lab data.

2. Output unique raw lab type and raw lab test name to an Excel file called lbtestid_mapping.xls.

3. Determine standard lab type and lab test based on the internal repository. The Excel sheet, to be stored in the dataclean/compound/study/datasets directory will contain 4 non-missing columns - rawtype, rawname, stdtype and stdname (and no additional columns) - and will be reviewed by clinical before being considered final.



	A	B	C	D
1	rawname	rawtype	stdname	stdtype
2	Immunoglobulin G	Antibody	IgG Total	Antibody
3	ALT/SGPT	Chemistry	Alanine Transaminase - ALT (SGPT)	Chemistry
4	AST/SGOT	Chemistry	Aspartate Transaminase - AST (SGOT)	Chemistry
5	Albumin	Chemistry	Albumin	Chemistry
6	Alkaline Phosphatase	Chemistry	Alkaline Phosphatase	Chemistry
7	BUN	Chemistry	Blood Urea Nitrogen	Chemistry
8	Bicarbonate	Chemistry	Bicarbonate	Chemistry
9	Bilirubin (Direct)	Chemistry	Bilirubin Direct	Chemistry
10	Bilirubin (Indirect)	Chemistry	Bilirubin Indirect	Chemistry
11	Bilirubin (Total)	Chemistry	Total Bilirubin	Chemistry
12	Calcium	Chemistry	Calcium	Chemistry
13	Chloride	Chemistry	Chloride	Chemistry
14	Creatine Kinase (CK)	Chemistry	Creatine Kinase	Chemistry

4. Import the completed mapping file and attach standard name and type to the working lab dataset.

```
proc import datafile="&datadir/lbtestid_mapping.xls"
            out=labmap
           dbms=excel replace;
run;
```

5. Import the standard mapping file and the standard conversion factors file.

```
proc sort data=mdatadir.standard_lab_names
            out=standardid (keep=standard_lab_name standard_lab_type lbtestid restype
                            rename=(standard_lab_name=labtest standard_lab_type=labtype));
  by lbtestid;
run;

proc sort data=mdatadir.standard_conversion_factors
          out=standardunit (drop=f6);
  by lbtestid;
run;
```

6. Process the data.

```
proc sql;
*-Attach LBTESTID to the unique list of raw lab type and name by merging on standard type
and name-*;
  create table lbf1 as
  select labmap.*, standardid.*
  from labmap left join standardid
  on labmap.stdname=standardid.labtest and labmap.stdtype=standardid.labtype
  order by stdtype, stdname;
```

```
*-Merge the above data (one unique observation per raw lab test) with the original, complete
raw dataset (LAB1 in this example) by merging on raw type and name. Now each observation has
a standard type, name, and LBTESTID attached-*;
  create table lbf2 as
  select a.*, b.lbtestid, b.restype
  from lab1 a left join lbf1 b
  on a.rawtest=b.rawname and a.rawtype=b.rawtype
  order by rawtype, rawtest;

*-Merge standard conversion factors to data on LBTESTID for further processing-*;
  create table lbf as
  select a.*,b.stdunit, b.factor, b.formula
  from lbf2 a left join standardunit b
  on a.lbtestid=b.lbtestid and a.rawunit=b.rawunit
  order by rawtype, rawtest;
quit;

data lab2;
set lbf (in=a);

  labtest=rawtest;
  labtype=rawtype;

  *-If factor>1, do convert to standard units-*;
    if factor not in (.,1) then do;
    convert="Y";
       unit=stdunit;
       *-Along with result, the values for the upper and lower limits of normal should also
       be converted-*;
       if nmiss(factor,result)=0 then do;
          result=result*factor;
         resultc=left(put(result,best.));
       end;
       if nmiss(factor,hhi)=0 then do;
           hhi=hhi*factor;
          uln=left(put(hhi,best.));
       end;
       if nmiss(factor,llo)=0 then do;
           llo=llo*factor;
          lln=left(put(llo,best.));
       end;
  end;

*-If factor=1, do not consider converted but use the standard unit label-*;
  if factor=1 then unit=stdunit;
    else if factor=. then unit=rawunit;
run;
```

Note that LABTEST and LABTYPE will always reflect the original naming convention at the study level. LBTESTID acts as our unique key at the integrated level. With the lab data standardized for most tests, we can easily run summaries and queries off of the data without further processing of the results. This has been extremely valuable for us.

CREATING THE INTEGRATED DATASETS

Let's assume the data have been harmonized for each of the studies, and have been validated. The next step is to create the integrated datasets. Every two months, an update macro (%MUPDATE) is run which serves any or all of the following functions:

1. Ongoing studies: Copy data from the working directories for ongoing studies to their counterpart raw data folders.
2 .Run all programs in the ISDB dataclean folders for selected studies. Selected studies may be:
    (a) All studies
    (b) Ongoing studies
    (c) Up to 10 selected studies (selected by protocol/study number)
3. Run the dataset creation at the integrated level
4. Run all existing queries [Queries to be discussed later]

SPECIAL PROCESSING FOR LAB DATA

Step 3 of the MUPDATE macro sets together all of the harmonized data to create one integrated dataset for eight of the nine core datasets defined, and defines any integrated-level variables as defined in the VDTs. There are separate domains for the integrated-level datasets for the MATTRIB and MNMCH macros (for example, DMI for integrated demographics dataset ANALYVAR). The 9th, ALAB_RAW, is processed a little differently. Due to the sheer number of observations and the need to easily pluck information for analysis requests, the lab data is set and then split out to the data warehouse by LBTESTID. Currently we have approximately 300 different unique lab datasets. Examples include: ALAB_RAW_CHEM_ALT, ALAB_RAW_COAG_APTT. A last dataset is ALAB_RAW_OTHER, which contains all data which are not assigned a LBTESTID value yet.

ISISID: SPECIAL CONSIDERATION FOR SUBJECTS WHO PARTICIPATED MORE THAN ONCE IN OUR CLINICAL TRIALS

There are a number of subjects who have participated in more than one of our clinical trials, or were reused in a trial, and this information is very important for analysis purposes. In order to track clinical subjects through successive ISIS clinical trials, two variables have been added to each record in the integrated ANALYVAR dataset (which contains one record per study per subject ID). These are ISISID, which is the first unique subject ID (USUBJID) ever assigned to a subject, and STUDYSEQ, which shows the progression of the subject through ISIS studies (or cohorts or occurrences within the same study in some cases). A typical sequence might look like:

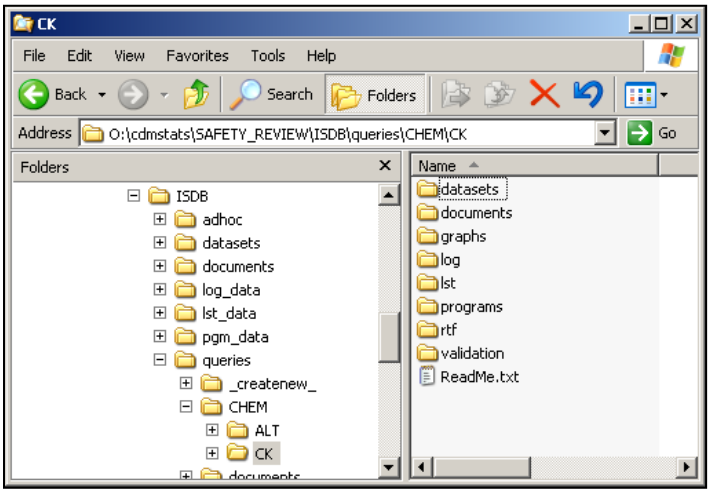
PROTOCOL	SUBJID	SITEID	USUBJID	ISISID	COHORT	STUDYSEQ
COMPOUND01-CS01	1006	1503	COMPOUND01-CS01 1503-1006	COMPOUND01-CS01 1503-1006	A	1
COMPOUND01-CS01	1011	1503	COMPOUND01-CS01 1503-1011	COMPOUND01-CS01 1503-1006	D	2
COMPOUND01-CS03	1203	1503	COMPOUND01-CS03 1503-1203	COMPOUND01-CS01 1503-1006	B	3

The information needed to track these subjects either comes from internally maintained Excel spreadsheets or from prior study information included in the SAS raw data received from data management.

QUERIES

Queries are requested from the clinical team. In order to keep everything organized and easily reproducible each time the ISDB is updated, we rely on the following: organized folder structure, standard naming conventions, and query analysis request documents.

Each time a new query is requested, a new folder is created to house all programming work, output, and documentation. If a similar query is requested later (say a new table or graph for a particular lab test), then the work is done under the existing folder. The folder structure is as follows:



For each new Chemistry lab test query request, a new subfolder will be created under CHEM.

The documents folder houses the original request documentation. The document specifies the analysis populations and subpopulations of interest and the datasets, table, listing, figures and/or patient profile specifications, including layout, titles, footnotes, and other details. The ReadMe file includes the output file naming conventions, designed for easy navigation. For example, the naming conventions for the CK queries are as follows:

Tables: Tck_&s_&period_&subpopulation
Graphs: Gck_&statistic_&period_&subpopulation
Patient Profiles: ptprofile_&study_&site-&subject

Where:
S: S=Shift table
SA=Shift table (absolute value)
D=Descriptive table

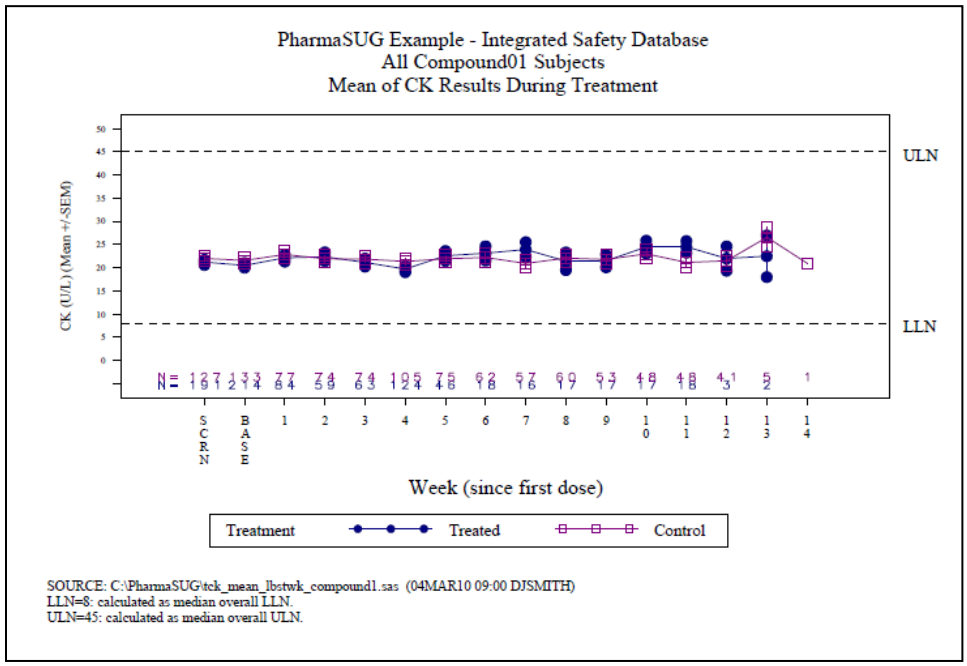
Period: LBSTWKTRT=During Dosing Period
LBSTWK=During Study
LBPOSTWK=During Post-Treatment Period

Statistic: MEAN, ABCHG, PCHG

Subpopulation: (all are Safety population)
1. Compound01: All Compound01 (exclude CS02)
2. nonCompound01: All non-Compound01
3. Compound01_bmi: Compound01 BMI>=25
4. nonCompound01_bmi: non- Compound01 BMI>=25
5. Compound01_bmi1: Compound01 BMI<25
6. nonCompound01_bmi1: non-Compound01 BMI<25

Study: Study ID
Site: Site ID
Subject: Subject ID

Once a request is received, programming and QC practices begin and follow the normal internally mandated programming standards and SOPs. The following is an example of a file named Tck_mean_lbstwk_compound1.pdf. It displays mean CK results during treatment for the 400mg/wk and Control groups for Compound01.



We have in place a number of report tools to help us quickly create output, including macros for:

- Creating standard mean and percent change graph
- Creating patient profiles
- Creating standard tables
- Assigning table numbers and titles dynamically
- Convert .lst output to .rtf

PRACTICAL CONCERNS

There is no clear roadmap for building an across-study, across-compound integrated clinical database. When a programmer generates datasets, listings, tables and graphs in a clinical study there is a definite beginning and end. There is no end to the integrated database. It continually grows as studies are completed, and more data is added to it.

If your company has a need for you to create such a database, how will you proceed?

Because of the wide scope of this task, it is critical that you clearly understand the role this database will have in your company before you even think about starting any programming activities. You will not have enough time and resources to exhaustively integrate and use every piece of available study-level data. Talk to the clinicians and department heads in your company who will later be asking you for query results. Try to find out in advance what kinds of questions they are hoping to answer from this database. What are their needs? This will help prioritize and guide your work. At Isis, we have found that others in the company are most interested in analyzing lab data, so we spend a disproportionate amount of time integrating lab data.

You will undoubtedly be dedicating large amounts of time and energy to this project, so realize in the beginning that you will probably want this database to be useful (and not become obsolete) for years to come. Over time the database size may become extremely large, so get some understanding of any storage space limitations. Discuss your database with your company's IT department. You don't want to build the database and discover that you don't have any storage space to add future studies.

As you think about how you will organize the work, keep these two words in mind: "standardize" and "automate". As you work on the database and it starts to take shape, if you don't look for ways to standardize and automate processes, you risk ending up with a database that requires many repetitive, tedious manual changes to use, or worse yet, one that is practically unusable. We have found it useful to create standardized directory structures and standardized SAS program names. Standardization naturally lends itself to creating macros and automating general processes (like updating the data in the entire database), which would not be possible otherwise.

FUTURE OF THE ISDB

To further develop the ISDB, we plan to implement two approaches simultaneously. First, we will add more data to the database, including clinical pharmacokinetics, pre-clinical, and clinical data from other compounds. Second, we'll utilize new tools from SAS 9.2 such as Business Intelligence server version, which contains Microsoft Office integration and web-based report building and data exploration functionalities, to streamline the process of generating clinical and pre-clinical queries and reports, and allow clinicians to submit updated reports without programming in SAS.

REFERENCES

SUGI 28 Paper 109-28: Build a SAS® Development Environment under Windows®. Wei Cheng.
http://www2.sas.com/proceedings/sugi28/109-28.pdf

ACKNOWLEDGEMENTS

Along with the authors, our team includes: Di Yu, Doreen Chen, Nelson Salgado, Shuting Xia, and John Su.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Denise J. Smith
Isis Pharmaceuticals
1896 Rutherford Road
Carlsbad, CA 92008
djsmith@isisph.com