Paper PO02

Enriching decision making through data integration: Creating a project database (PDb)

Jennifer Lomax, AstraZeneca, Alderley Park, UK
Nicki Noble, AstraZeneca, Alderley Park, UK

INTRODUCTION

Have you ever been asked to provide pooled data from multiple studies to help make quick decisions? Sometimes this can be quite a frantic job - collating data from different sources in different formats to provide the right answer at the right time.

Creation of a database to hold all project data, early in the development of a project can be invaluable as an accessible source of accurate information that is vital for making key decisions. By integrating data from all studies into a standard structure, it is possible to interrogate the data quickly and efficiently. This paper presents how we have developed a project database of derived data to aid our project team in making key decisions. It describes the processes used and the key considerations for collating data to create meaningful, accurate reports.

PLANNING AND PREPARATION

- Identify the key stakeholders of the project database and establish their requirements. The stakeholders need to think about what they want from the PDb and order their priorities. There may be several groups of stakeholders and their requirements need to be assessed together. Involvement from project team members will be vital to determine how the stakeholders' priorities align with the overall project priorities.
- Consider what studies will be included. The PDb should include as many studies as possible. Assess the feasibility of integrating data from in-house, CRO and collaborative group studies.
- When reports are created the users will need to know what data is included in order to put the information into context. Think about what data is available when the database is updated.
- It is important that studies are created according to the project standards in order to aid the integration of data. Assess which studies adhere to the standards and to what extent. Consider how much pre-processing is needed to prepare legacy studies for integration.
- Consider whether it will be beneficial to the project to create a PDb. Early resource investment is key to the success of a PDb, however it is also important to think about the stage of the project and processes already in place.
- The stakeholder requirements will form the basis to the scope of work, and the priorities will define the order in which work must be progressed. Depending on the aims of the PDb for the project the inclusion of datasets and variables should be considered e.g. whether it is appropriate to include study specific datasets and variables, exclude datasets that are not required for analysis and the creation of derived variables that aren't available on specific studies.

SET-UP

The following 3 key items should be set-up to act as the main building blocks for the PDb:

1. Create a specifications document in excel, to include the key data attributes for each variable in each dataset. This will include variable name, type, length, label, format for every variable that should be loaded into the PDb. Import the document into a SAS® dataset so data attributes are available in the dictionary tables for processing.

   This document will drive the inclusion of datasets and variables into the PDb as only the contents of this file will be loaded. This document can be used to easily control the contents of the PDb. If a deliverable requires only a subset of data, a new version of the specification document can be created. When the PDb is updated, an abbreviated version will be loaded with only the data of interest. This is particularly useful if the processing time to update the PDb is large, as it allows unnecessary processing to be reduced.

2. Create a study information document in excel to contain details relating to individual studies, and include anything you consider to be relevant to the use of the pooled database. Include one line per study with columns including the study design, status of the study (open label, blinded or unblinded), location of study data, whether data is final or ongoing and any useful data to group reports that is not already available in other variables. This will continue to develop as requests are made from the project team.

   By importing the document into SAS useful metadata is made available for use to allow easy subsetting for reports. For example, if a report including only phase I studies is required, the data can easily be

extracted by using a simple where clause (e.g. where phase = 'I') rather than needing to state each study that is required.

The study information spreadsheet should be updated whenever new studies are added to the project, or the status of the study and location of data changes, to maintain the integrity of the PDb.

3. Create a SAS program to assign LIBREFs to reference all studies to be included in the PDb. Use parameters from the study information document to dynamically define the library for each study. Protect the source data using ACCESS=READONLY on each LIBNAME statement. For ease of future processing name the LIBREF consistently for each study e.g. STUDY01, STUDY02 etc.

   By setting up a dynamic program based on the contents of the study information document no future maintenance of the program is required.

Once all libraries are set up, it is possible to determine which datasets are available in the project and create an index. The creation of this can be automated by utilising the SQL dictionary tables. Store a list of standard dataset names in a temporary dataset (in this case WORK.TEMP) and compare these to each of the study datasets available in dictionary.TABLES. Use the resulting dataset to provide a useful report for users on the availability of data. A code example of how to create the initial dataset is given below, which is then transposed to create a useful output.

```
proc sql noprint;
create table work.std as
   select distinct libname, memname,
          put(input(compress(libname,'STUDY'),2.),z2.) as studynum,
          'Y' as instudy
   from dictionary.tables
   where memname in (select memname from work.temp)
         and libname like 'STUDY%'
   order by memname, studynum;
quit;
```

Original dataset:

Library Name	Member name	STUDYNUM	INSTUDY
STUDY01	AE	01	Y
STUDY05	AE	05	Y
StUDY10	AE	10	Y
STUDY03	LAB	03	Y

Transposed version:

Dataset Name	Study ID: 01	Study ID: 03	Study ID: 05	Study ID: 10
AE	Y		Y	Y
LAB		Y		

If the data standards include formats, consider those applied across all studies. It is possible that similarly named formats can exist in studies, but have different decodes for values. Differences need to be rectified for a complete catalog to be created for the PDb data. Consolidate all study formats into a unique format catalog and output all discrepancies in values and decodes to a report. The report should then be reviewed in order to finalise the formats in the catalog.

CREATION OF THE PROJECT DATABASE

Before data can be loaded, legacy study data should be mapped to the project standards. This should be performed in a separate pre-processing program for any datasets and variables where it is deemed appropriate for the project. This may include changing the structure of datasets to meet the standards.
It may not be appropriate to map the entire dataset to the project standard if only a portion of that dataset will be utilised in the PDb. Consider the requirements before undertaking this task to save resource where possible.

Load data into the PDb using the specifications file to drive inclusion – only datasets and variables that exist in the file should be included. Below is a code example to dynamically load data as specified in the set-up documentation.

```
** Select the data to load using SQL dictionary tables **;
      create table work.sbj_vars as
        select distinct libname, memname, name, label, npos
        from dictionary.columns
        where upcase(memname) like ('AE%') and
         (upcase(LIBNAME) LIKE 'STUDY%') and name in
         (select distinct name from dictionary.columns
            where upcase(libname) = 'SPEC' and upcase(memname) = 'AE')
        order by memname,name;

** Make libnames and datasets available in macro variables  **;
** Count datasets to load                                   **;
      select libname, memname, count(*)
        into :libs separated by ' ', :data separated by ' ', :count
        from sbj_data
        order by libname;

** Create an empty table of the spec **;
       create table work.shell like spec.AE;

** Use a macro to load the data. Cycle through each dataset in &LIBS and **;
** &DATA **;
      %do i = 1 %to &count;
        data work.%scan(&data,&I);
          set %scan(&libs,&i).%scan(&data,&i);

          .....Any common processing needed before the load.....

        run;

** Dynamically create INSERT statements to load variables that exist in **; ** the
shell **;
        proc sql;
          select distinct name into :values separated by ',' from sbj_vars
          where (memname="%scan(&data,&i)" and libname="%scan(&libs,&i)");
          insert into work.shell (&values)
            (select &values from work.%scan(&data,&i));
        quit;
```

New requests from the project team, may mean that new variables need to be created within the PDb that are not already defined in the project standards. New variables may aid the review of the combined data or make increase the efficiency of the data processing. We recommend that a common identifier be used to distinguish these variables from the study data, for example a preceding P_. E.g. P_DOSE.

Metadata for each study should be collated throughout the process in order to track and document the contents of the PDb. Important information to capture are:

- Original source dataset – for legacy studies this should also include the name of pre-processing program.
- Date of last extract – the date that the original source dataset was last updated.
- Number of observations / variables – state what has been loaded into each dataset.

The code example below shows how to extract the required information from dictionary.tables and dictionary.members

```
      %macro mk_meta(_study = , _LIB =, _data = ,_prog=);
       proc sql noprint;
        create table m_meta&_study as
        select distinct tables.libname,
           'Drug X Study '||PUT(&_study,Z2.) as study,
           tables.memname, tables.modate, tables.nobs, tables.nvar,
           case
             when index(upcase(lib.path),'DEV') then 'DEV'
             when index(upcase(lib.path),'PROD') then 'PROD'
             else 'UNK'
           end as source, "&_prog" as prog
        from dictionary.tables tables,dictionary.members lib
        where tables.memname = "&_DATA" and
             tables.libname = "&_LIB"||PUT(&_STUDY,Z2.) and
             tables.libname = lib.libname;
       quit;
      %mend;
```

It is useful to create a cover sheet for each report generated from the PDb, to show what data has contributed to the output. This would be produced using the metadata and study information document.

Study	Indication	Date last modified	Status of data	Number of contributing rows	Blinded Yes / No / Open
1	SAD study	15/04/2007	Final	98	Open
2	MAD study	28/11/2007	Final	43	Open
3	AML	18/08/2008	Ongoing	36	Yes

THINGS TO LOOK OUT FOR

• Make sure the source data for each study is as up to date as possible.
• Studies that are blinded will not have a treatment group available. Consider how this will impact analysis and whether the data should be included.
• The same variable may be derived differently on different studies so it may not be sensible to combine the data.
• Ensure the TYPE attribute is the same as standard specifications or results may not be as expected.
• Output from the PDb should reflect conclusions within the original CSR so be careful not to re-derive variables.
• Where study specific anomalies exist input will be require from the lead study programmer. These differences should be highlighted within individual study dataset specification documentation.
• If changes are made to the standards consider the impact on the PDb. Something that appears to be insignificant on a study may have a big impact when pooling data.

USES OF A PROJECT DATABASE

Patient Safety have been the main customer utilising information from the PDb for input to safety evaluation and patient risk management. Also the Clinical Project Team have used the data to aid decision making on the project such as reviewing data from multiple studies to determine new study requirements. As the PDb contains all current data it has also been possible to use the data for input to the annual IND report. In addition plans are in place to utilise the PDb for an upcoming submission. Analysis of the collated data is possible using SAS outputs, Data Visualisation or other applications such as excel.

CONCLUSION

Before undertaking development of a project database, consider resource requirements against the benefits it could provide. Start with data which is prioritised by stakeholders and then let additional requests for information drive future development. Set up dynamic programs to allow for new data and studies to be added in an efficient way. To enable all the programmers on the project to utilise the project database a comprehensive user guide will need to be written to explain the processes and complex macros used. The documentation created from the metadata will also provide users detailed information about the content of the project database each time it is updated.

Early planning and resource investment towards creating a project database can provide huge rewards.

CONTACT INFORMATION

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

Jennifer Lomax & Nicki Noble
AstraZeneca
Parklands
Alderley Park
Macclesfield
Cheshire
SK10 4TG

Jennifer.Lomax@astrazeneca.com
Nicki.Noble@astrazeneca.com