Introduction and Statement of the Problem

Pharmaceutical companies conduct clinical trials to demonstrate that their products are safe and effective for their intended use. In processing data collected during clinical trials of pharmaceutical products, it is very important to assure that the data are complete, internally consistent and consistent with the requirements of the trial protocol. Although many data entry systems are capable of checking for valid data on a field by field basis at the time of data entry, it usually is necessary to conduct some post-entry, cross-field and cross-record data comparisons to assure that the data are consistent. This paper presents a system that uses Ingres® and SAS® to produce patient profiles of clinical research data with inconsistent or incorrect data that have been flagged for review.

Berlex Data Editing Process

At Berlex, Case Report Forms (CRFs) are retrieved by Clinical Research Associates (CRAs), reviewed for obvious errors, and forwarded to Clinical Data Administration for data entry. The data are entered directly into an Ingres database by using a database application that includes double-key entry validation. The system also permits batch loading of data, such as data provided electronically by central clinical laboratories. Following data entry, Clinical Data Administration staff produce and review patient oriented edit listings of the data and distribute them to the CRAs. The CRAs resolve the flagged and questionable data, if necessary by referring a query to the study investigator. The corrections are noted on the edit listing pages which are returned to Clinical Data Administration so that the corrections may be entered into the database. This cycle of printing edit listings for CRA review may be repeated, if necessary. When the data from all patients in a trial have been entered into the database, reviewed and corrected, the database is subjected to a "locking" process and the data are released for statistical analysis and reporting.

In the past, the production of edit listings has been a programming intensive process because of the evolution of CRFs and because of variations in data structure and protocol requirements from protocol to protocol. With conversion from a per-protocol database system to a per-project, multi-protocol database, it has become possible to reduce the programming effort required to produce the edit listings.

The Clinical Trials Database

The core structure of the Berlex Clinical Trials Database is depicted in the Entity-Relationship diagram of Figure 1. This consists of one Ingres database per clinical development project. Data from all protocols supporting the project are stored in a common set of tables such that data from common CRF pages are stored in a single table or set of related tables. For example, all adverse experience reports are inserted into a single table indexed by protocol, subject, schedule (visit) and an adverse experience sequence number. Likewise all physical exam data are stored in a single table, clinical laboratory data in a single set of tables, etc. Some data may be stored in protocol specific tables, if necessary because the data are specific to a single protocol or to a small number of protocols.

![Figure 1. Clinical Trials Database](image-url)
An essential part of the database design is the use of code tables that define the universe of valid values of coded data fields. There are three versions of code tables: 1) the general code table contains standard codes used consistently across all protocols, 2) protocol specific code tables contain codes that may be specific to protocols including those that define the categories used for medical history, physical exams, etc., and 3) protocol specific clinical laboratory code tables identify the specific lab tests used in each protocol. We have also created a special code table, the RANGE table, that defines ranges for non-coded fields (e.g., age, blood pressure, etc.) and special codes that control edit flag definitions and the printing of the listings. Although these code tables (except the RANGE table) were originally designed as a part of the mechanism to drive the data entry applications, they are also used to create SAS format libraries for reporting purposes.

The Edit Listing User Interface

The Edit Listing User Interface is comprised of three control edit flag definitions and the printing.

These choices, except the edit listing pages, are stored in the ED_SELCTT table and are automatically indexed by a sequential number stored in the ED_SEQ column. This value of ED_SEQ uniquely identifies the current user request to produce edit listings and it is the key element that links the request to the SAS program and MACROS that generate the listings. One row for each edit listing page selected is inserted into the ED_PAGES table and is indexed by ED_SEQ and PAGE_NO. The application then creates a VMS DCL command file to execute the edit listing programs and submits it for either immediate or delayed execution.

The ED_PROGS table contains the names and locations of SAS MACROS that produce the listings for each data type (e.g., subject demography and medical history, physical exams, adverse experiences, etc.). In general, there is a MACRO for each unique page in the subject case book plus one for CRF login and tracking information. Thus, for each MACRO there is one row in the ED_PROGS table per protocol indexed by PROTOCOL and PAGE_NO. The ED_PROGS table is maintained by the Database Administrator.

The EDITLIST Program

The edit listing output is actually produced by submitting to a batch queue the DCL command file created by the edit user interface application. The command file sets the execution environment, defines DCL symbols containing the name of the database and the current value of ED_SEQ and executes the EDITLIST SAS program.

```
DATA _NULL_; * Set up Edit Listing by * 
FILE = "\DB\GSAS\"; /* Macro Library in */ DATA Về * 
LIBNAME GV "\DB\GSAS\"; /* the production */ OPTIONS SASAUTOS=ML MPRINT MACROGEN SYMBOLGEN; /* database */ * 
STOP; /* Establish link to */ 
RUN; /* SAS/ACCESS VIEW */ 
CALL SYMPUT('DB',GETSYM('PROJNAME')); /* Descriptors in the */ 
CALL SYMPUT('VDDT',GETSYM('VDDT')); /* Production Data- */ 
CALL SYMPUT('DE',GETSYM('DE')); /* base */ 
CALL SYMPUT('D1',GETSYM('D1')); /* Set up Edit Listing by */ 
%_E-SETUP(_DB) /* creating Global MACRO */ 
%_E-FORMAT /* variables to select patients */ 
%_E_DSETS /* and visits and to control the */ 
%_E_PSETS /* printing of edit flags */ 
%_E_PAGES /* Create Edit Listing Formats from */ 
%_E_DSETS /* Ingres Code Tables and Range */ 
%_E_DSETS /* Tables */ 
%_E_PSETS /* Set up commonly used data sets */ 
%_E_PSETS /* Execute the Edit Listing MACROS */
```
EDITLIST is a driver program that manages the interface to the Ingres database, sets up the FORMAT library, retrieves commonly used data from the database, and executes the user's request for edit listings. EDITLIST first retrieves the name of the database from the DCL symbol by using the GETSYM function within a DATA _NULL_ step. A MACRO variable, DRUG, is assigned the product name for use in the listing titles. The remaining functions are accomplished by successively calling four MACROs: E_SETUP, E_FORMAT, E_PAGES, and E_DSETS.

The E_SETUP MACRO

The E_SETUP MACRO initializes a number of SAS MACRO variables that contain printer control specifications, flag printing specification, and the patient and CRF section selection criteria. The set of edit listing SAS MACROs to be executed are stored in a SAS dataset, E_PAGES, that is subsequently used by the E_PAGE MACRO.

E_SETUP

%MACRO E_SETUP(DB);
%LOCAL CLAUSE ED_SEQ CC_OPT DB;
%GLOBAL PROT PROJNO PROTNO FLAGSON EFORMAT WCLSl
WCLS2 WCLSJ WCLS4 WCLS5 WCLS6 PAGSIZE
LINENUM N_PAGES;
OPTIONS MISSING=' ', SYMBOLGEN;
DATA _NULL_; /* Get Edit Request Sequence */
/* Number from DCL Symbol */
CALL SYMPUTC 'ED_SEQ', GETSYMC 'ED_SEQ';
STOP;
DATA E_PAGES;
SET GV. V_PAGES (WHERE=(ED_SEQ=ED_SEQ));
NR + 1;
CALL SYMPUTC ('N_PAGES', NR);
RUN;
DATA _NULL_;
SET GV. ED_SELECT (WHERE=(ED_SEQ=ED_SEQ));
CALL SYMPUTC ('ED_SELECT', WHERE-&CLAUSE);
CALL SYMPUTC ('PROT', PROTOCOL);
CALL SYMPUTC ('WCLS1', TRIM(WRCLS1));
CALL SYMPUTC ('WCLS2', TRIM(WRCLS2));
CALL SYMPUTC ('WCLS3', TRIM(WRCLS3));
CALL SYMPUTC ('WCLS4', TRIM(WRCLS4));
RUN;
%LET PROTO = %STR(&PROT,1,3); /* Protocol specific SAS/ACCESS VIEWS */
%LET DB = %STR([P&PROT.. SAS]);
/* Set composite patient selection criteria */
/* Other conditions in an existing WHERE clause */
/* Add Schedule selection conditions to */
/* complete WHERE clause to be used as the only dataset option */
%END;
%IF %LENGTH(&CLAUSE)=0 %THEN %DO;
%LET WCLS1=WHERE-&CLAUSE;
/* WHERE conditions to be added to */
/* other conditions in an existing */
/* WHERE clause */
%LET WCLS2=WHERE-&CLAUSE;
/* complete WHERE clause to be */
/* added to other dataset options */
%LET WCLS3=WHERE-&CLAUSE;
/* complete WHERE clause to be */
%END;
%ELSE %IF %LENGTH(&CLAUSE)=0 %THEN %LET
CLAUSE=&ACLUS;
/* Set composite patient selection criteria */
/* Other conditions in an existing WHERE clause */
/* Add Schedule selection conditions to */
/* complete WHERE clause to be used as the only dataset option */
%END;
%ELSE %IF %LENGTH(&ACLUS)=0 %THEN %LET
CLAUSE=&ACLUS;
/* Set composite patient selection criteria */
/* Other conditions in an existing WHERE clause */
/* Add Schedule selection conditions to */
/* complete WHERE clause to be used as the only dataset option */
%END;
/* Set Composite Patient selection criteria */
/* Other conditions in an existing WHERE clause */
/* Add Schedule selection conditions to */
/* complete WHERE clause to be used as the only dataset option */
%END;
%ELSE %IF %LENGTH(&ACLUS)=0 %THEN %LET
CLAUSE=&ACLUS;
/* Set composite patient selection criteria */
/* Other conditions in an existing WHERE clause */
/* Add Schedule selection conditions to */
/* complete WHERE clause to be used as the only dataset option */
%END;
%ELSE %IF %LENGTH(&ACLUS)=0 %THEN %LET
CLAUSE=&ACLUS;
/* Set composite patient selection criteria */
/* Other conditions in an existing WHERE clause */
/* Add Schedule selection conditions to */
/* complete WHERE clause to be used as the only dataset option */
%END;
First, the value of ED_SEQ is retrieved from the DCL symbol, ED_SEQ, and assigned to a SAS MACRO variable, ED_SEQ, to establish the link to the Ingres edit listing interface tables. Then the E_PAGES dataset is created with one observation (row) for each edit listing page to be produced. The variables include the SAS MACRO name and the page number. Next MACRO variables are defined to set the carriage control option, the page size, the top margin and a variable to control printing of data integrity flags. Finally the row indexed by the current value of ED_SEQ in the Ingres ED_SELCT table is read to retrieve the patient and CRF section selection criteria. These criteria had been stored in character fields of up to 200 characters in the form of SQL Where clause specifications (eg. WHERE PAT BETWEEN 1001 AND 1999). E_SETUP concatenates the criteria with AND or OR operators as appropriate and assigns the results to a set of MACRO variables that are used by the edit listing MACROs in the WHERE dataset option to select subsets of the data in the Ingres database.

The E_FORMAT MACRO

The E_FORMAT MACRO reads and merges the Ingres Code Tables and the Range table to derive a SAS dataset that can be used with the CNTLIN option of PROC FORMAT. The resulting FORMATs are used within the edit listing MACROs to define and print flags that identify inconsistent or invalid data. The three code tables, V_CD_##, V_LCD_##, and V_CD_TBL, contain standard codes used by the Ingres data entry and modification applications to identify and label the acceptable values of variables (columns) in the database. The RANGE table contains additional codes not included in the Ingres Code Tables and some values that replace rows from the code tables in certain cases for which it is necessary to replace the value label.

The E_DSETS MACRO

Some of the Ingres tables contain data that are used in all or most of the edit listing MACROs. Rather than have each MACRO retrieve the same data from the Ingres database, the E_DSETS MACRO creates temporary SAS datasets corresponding to those tables once for the entire EDITLIST execution. The patient and CRF section selection criteria are applied in the creation of these datasets.

E_DSETS

%MACRO E_DSETS;
%GLOBAL DB PROT PROJNO FLASGON WCLS1 WCLS2 WCLS3 WCLS4 WCLS5 WCLS6 PAGSIZE LINENUM;
%-----------------------
| Get Patient records and Investigator |
| information |
%-----------------------;
DATA PATIENTS;
MERGE PV.PATIENT(IN=P WCLS2) PV.INV;
BY INV;
IF P;
INV_INIT = SUBSTR(FRSTNAME,1,1) || ' ';
IF FLASGON THEN DO;
IF INITS = ' ' THEN INITS = '**';
END;
RUN;

NFSUG '93 Proceedings
The E_PAGES MACRO

The E_PAGES MACRO simply establishes a %DO loop that processes a number of iterations equal to the number of observations in the E_PAGES dataset. On each iteration, an output print file is specified with a filename of "PAGE##.LJS" where ## is the page number from the Ingres ED_PAGES table. Also the MACRO, the name of which is the value of the PROGNAME variable, is executed.

```
E_PAGES
%MACRO E_PAGES;
%GLOBAL PROT PROTNO PROJNO FLAGSON WCLS1 WCLS2 WCLS3 WCLS4 WCLS5 WCLS6 PAGSIZE LINENUM;
%/*********************************************************/
%* CALL IN SET-UP MACRO FOR GENERAL COMMENTS */
%* FROM THE AUTOCALL LIBRARY. */
%/*********************************************************/
%&PG_PROG
%/*********************************************************/
%* INPUT AND MERGE THE PHY AND SCHEDULE */
%* TABLES, KEEPING THOSE OBSERVATIONS THAT */
%* APPEAR IN THE PHY TABLE. CREATE TWO */
%* DATASETS: PHY AND PHYDATES. THE PHY */
%* DATASET CONTAINS ALL OF THE PHY INFO. THE */
%* PHYDATES DATASET WILL CONTAIN 1 RECORD FOR */
%* EACH PT VISIT AT WHICH EXAMS WERE DONE. */
%* IT WILL ALSO CONTAIN ALL TPINFO INFO AND, */
%* IF TPINFO IS ≤ 7 OR 9, A FLAG IS SET. */
%/*********************************************************/
%&PG_NDX = &PG_NDX;
SET E_PAGES POINT-PAGE_NDX;
CALL SYMPUT('PAGE_NO', PAGE_NO);
CALL SYMPUT('PG_PROG', PROGNAME);
STOP;
%/*********************************************************/
%* ASSIGN file for DATA_NULL output */
FILENAME PAGE&PAGE_NO "!PAGE&PAGE_NO..LJS";*
%&PG_PROG
%/*********************************************************/
%MEND E_PAGES;
```

A Sample Edit Listing Page: The Physical Exam

Figure 3 is a sample output of the physical exam edit listing page. The following SAS MACRO sets up the data, defines the data integrity flags and prints the listing:

```
E_PHY
%MACRO E_PHY;
%GLOBAL DB PROT PROTNO PROJNO FLAGSON WCLS1 WCLS2 WCLS3 WCLS4 WCLS5 WCLS6 PAGSIZE LINENUM;
%&PG_PROG
%/*********************************************************/
%* CALL IN SET-UP MACRO FOR GENERAL COMMENTS */
%* FROM THE AUTOCALL LIBRARY. */
%/*********************************************************/
%&PG_NDX = &PG_NDX;
SET E_PAGES POINT-PAGE_NDX;
CALL SYMPUT('PAGE_NO', PAGE_NO);
CALL SYMPUT('PG_PROG', PROGNAME);
STOP;
%/*********************************************************/
%* ASSIGN file for DATA_NULL output */
FILENAME PAGE&PAGE_NO "!PAGE&PAGE_NO..LJS";*
%&PG_PROG
%MEND E_PAGES;
```

The E_PAGES MACRO

The E_PAGES MACRO simply establishes a %DO loop that processes a number of iterations equal to the number of observations in the E_PAGES dataset. On each iteration, an output print file is specified with a filename of "PAGE##.LJS" where ## is the page number from the Ingres ED_PAGES table. Also the MACRO, the name of which is the value of the PROGNAME variable, is executed.

```
E_PAGES
%MACRO E_PAGES;
%GLOBAL PROT PROTNO PROJNO FLAGSON WCLS1 WCLS2 WCLS3 WCLS4 WCLS5 WCLS6 PAGSIZE LINENUM
%&PG_PROG
%/*********************************************************/
%* CALL IN SET-UP MACRO FOR GENERAL COMMENTS */
%* FROM THE AUTOCALL LIBRARY. */
%/*********************************************************/
%&PG_NDX = &PG_NDX;
SET E_PAGES POINT-PAGE_NDX;
CALL SYMPUT('PAGE_NO', PAGE_NO);
CALL SYMPUT('PG_PROG', PROGNAME);
STOP;
%/*********************************************************/
%* ASSIGN file for DATA_NULL output */
FILENAME PAGE&PAGE_NO "!PAGE&PAGE_NO..LJS";*
%&PG_PROG
%MEND E_PAGES;
```

Nesug '93 Proceedings
PATIENT: 123456 /ABC

FIRS'T INTERV INT. DATE: 01/14/93 START TIME: 15:00
LAST INT. DATE: 01/14/93 START TIME: 15:05

PHYSICAL EXAMINATIONS

SCHEDULE
DATE
TIME
POINT
TPINFO
----------
1.00 Baseline
01/14/93
14:00
-52.0 min

7.00 24 Hr
01/15/93
08:00
15.9 hr ??

BASELINE
SYSTEM
RESULTS
ANN. CHANGE
SPECIAL ABNORMALITY
----------
Baseline
(1) GENERAL APPEARANCE
Abs
NO
OBESE, SLURRED SPEECH, FACIAL DROOP

Baseline
(2) SKIN
Abs
NO
LT GREAT TOE ULCER

Baseline
(3) EYES
Norm
Norm

Baseline
(4) HEAD AND NECK
Norm
Norm

Baseline
(5) LUNGS
Abs
NO
FEW BIBASILAR CRACKLES

Baseline
(6) HEART
Norm
Norm

Baseline
(7) ABDOMEN
Abs
NO
OBSESE, MONTENDER MIDLINE SCAR

Baseline
(8) LYMPH NODES
Norm
Norm

Baseline
(9) MUSCULOSKELETAL
Abs
NO
SECONDARY LT LE FITTING EDMA

_FLAGS: ?? = Documentable data
?? = Missing Value
#? = Does not meet data/cross check requirements
** = Does not meet criteria specified in protocol

NOTE: REVIEW AS PAGE.

Figure 3. Physical Examination Edit Listing Page

/* First record number per patient */
IF FIRST.PAT THEN PATNDX = RECNDX;
IF LAST.PAT THEN DO;
/* Last record number per patient */
LVNDX = RECNDX;
/* Number of visits per patient */
NVISIT = LVNDX - PATNDX + 1;
OUTPUT;

E N D ;

****** MERGE THE PATIENTS, MED_DATE AND PBYDATES 
****** DATASETS. USE FOR PRINTING THE EDIT 
****** LISTING HEADER. CALCULATE TPHR. DO PHY 
****** DATE/TIME CHECKS.
****** DATA HEADER;
DATA HEADER;
MERGE PHYDATES(IN=P) MED_DATE 
PATIENTS(KEEP=INV INV_INIT LSTNAME 
PAT INITS);

BY PAT;
IF P;
PHYIDATE = DATEPART(PHYDATE);
PHYSIME = TIMEPART(PHYDATE);
IF PHYDATE < INPUT(PUT(SCHEDULE,LTPHRRR.),4.) 
& TPHR > INPUT(PUT(SCHEDULE,UTPHRRR.),4.) 
THEN TPFLAG = '??';
END;
IF PHYIDATE > INPUT(PUT(SCHEDULE,LTPHRRR.),4.) 
& TPHR < INPUT(PUT(SCHEDULE,UTPHRRR.),4.) 
THEN TPFLAG = '??';
ELSE TPHR = (PHYDATE - PPRDATE) / 3600;
ELSE TPHR = (PHYDATE - PRSDATE) / 3600;

****** FLAG INCONSISTencies IN THE PHYSICAL 
****** EXAMINATION DATES/TIMES.
****** Flag missing value for physical exam date. */

******* SPEcIFICATION PHY.1, PHY.2 >> Baseline */

******* physical exam between >72 and 0 hours */

******* before the 1st injection date. Post- */

******* injection physical exam within +/- 1/4 of */

******* intended interval. */

******* IF ((TPHR <INPUT(PUT(SCHEDULE, 
LTPHRRR.),4.) | (TPHR > INPUT(PUT(SCHEDULE,UTPHRRR.),4.)) 
& (TPHR NE .) ) THEN TPFLAG = '??'
END;
IF (ABS(TPHR) LE 2) & (TPHR NE .) THEN DO;
TPHR = 60 * TPHR;
TPU = 'min';
END;
ELSE IF TPHR NE . THEN TPU = 'hr';

******* SOFT THE COMMENTS DATASET. KEEP THE */

******* PHYSICAL EXAM SPECIFIC COMMENTS. TRANPOSE */

******* COMMENTS INTO ONE VARIABLE (COMM). SORT */
/* TBE PBY DATASET. SORT THE PBY DATASET AND */ /* MERGE WITH THE COM_PHY DATASET. //* *//* **********************************************/ /* MERGE WITH THE COM_PHY DATASET. */PROC SORT DATA-COMMENTS(WHERE=(COMTABLE = 'phy' AND COMGEN = 0)) OUT=COM_PHY(KEEP=PAT COMID COMKEY COMID-COM5) RENAME=(COMKEY-PHYCD)); BY PAT COMKEY COMID;

DATA COM_PHY(DROP=X COM1-COM5);
SET COM_PHY;
ARRAY COMMS (5) $ COM1-COMS;
SET COM...PBY;
DO X = 1 TO S;
END;
MERGE COM...PHY PHY;
BY COMID;
BY PAT PHYCD COMID SCHEDULE SEQ;
RETAIN COMOK 1;
MERGE PHY PHYNDX CRF_PHY COMNDX;
SET CRFEVAL(WHERE=(TBLNAME='phy' & COMID > 0));
BY PAT;
RETAIN COMOK 1;
IF LAST.PAT THEN DO;
IF _N_ > 1 & FIRST.PAT TBEN PUT _PAGE_; ELSE
END;
***COMBINE PHY DATASET WITH PHYNDX (PHYDATES */ /* POINTERS), COMNDX AND CRF_PHY DATASETS. DO */ /* EDIT CHECKS AND SET FLAG VALUES. */ ***-----------------------------------------------------------------------*/

DATA PHY:
MERGE PHYNDX CRF_PHY COMNDX;
BY PAT;
IF &FLAGSON THEN DO;
PHTABFLG = 99;
CHGFLG = 99;
IF PHYABN IN(2,9) & (PHYCBG = .) THEN
ELSE
IF PHYABN = 9 THEN DO;
CHGFLG - 99;
PHTABFLG = 99;
IF PHYABN = . THEN PHYABN = .T;
IF PHYCHG = . THEN PHYCHG = .T;
END;
END;

(* If: 1) result is normal or missing and */ /* change is missing or 2) result is abnormal */ /* and scheduled visit is baseline or 3) */ /* result is nd/unk and scheduled visit is */ /* baseline then set change to blank when it */ /* decodes. */ ***-----------------------------------------------------------------------*/

IF (PHYABN IN(1,9) & (PHYABFLG = 11)) | */ (PHYABN IN(2,9) & PHYCHG = .T) THEN */ PHYCHG = .T;

(* DETERMINE THE TOTAL # OF VISITS FOR EACH */ /* PHYSICAL EXAMINATION, FOR EACH PATIENT. */ /* THIS FREQUENCY WILL HELP IN THE CORRECT */ /* PAGINATION OF THE EDIT LISTING. MERGE THIS */ /* INFORMATION WITH THE PHY DATASET. */ ***-----------------------------------------------------------------------*/

PROC MEANS DATA-PHY N NOPRINT;
BY PAT PHYCD;
VAR SCHEDULE;
OUTPUT OUT=N_PHY N-CNTR;
SET PBY END-EOF;
CURRENT = TODAY();
FILE PAGE&PAGE_NO PRINT HEADER-TOP N-PS LINESLEFT=LL NOTITLES;
CURRENT = TODAY();
FORMAT PHTABDATE INJDATE MDMDY97.
PHTAB PHYF. PHTABDATE INJDATE MDMDY97.
PHTAB INF. PHTABPHF PHTABFLG CHGFLG */ ABNF. PHTABINF. PHTABINZP. PSER 6.1 */ SCHEDULE HSCHED. COMOK AAF. ;
IF _N_ > 1 & FIRST.PAT THEN PUT _PAGE_; CHECK = 6 & (&FLAGSON=1) + 2 * */ (&FLAGSON=0) ;
IF FIRST.PHYCD THEN DO;
IF (LL < CHECK) | (CNTR > (LL-CHECK)) THEN DO;
IF &FLAGSON THEN LINK LEGEND;
PUT / @20 ('PHYCD 2. ') @25 PHYCD $;
END;
ELSE
PUT / @20 ('PHYCD 2. ') @25 PHYCD $;
*/
The E_PHY MACRO illustrates most of the techniques used in these edit listing programs. Figure 3 may be divided into five sections: 1) the general header with patient identification, investigator and protocol identification, dosing information and the date of the edit listing, 2) the schedule header with observation dates, times and validity information, 3) the body of the listing containing patient data ordered by schedule (visit) and data specific keys (eg. body system code), 4) general comments (not depicted in this example), and 5) legend information. The MACRO may be considered organized into three functions: 1) data collection and setup, 2) data integrity flag definition, and 3) producing the formatted output.

The first step in data collection is execution of the GCOMSU MACRO that selects the appropriate comment records to be printed by the GCOMPT MACRO. Then the physical exam data is divided into two sets. Data to be used in the schedule header goes into a dataset with one record per patient per visit. Data used in the body of the listing goes into a dataset with one record per patient per visit per body system. Subsequently abnormality comments are merged with these data.

For both sets of data, data integrity flag definition is controlled by the &FLAGSON MACRO variable that is defined in the E_SETUP MACRO. As an example of a header flag, if the time of the 24 hour visit is not within 18 to 30 hours following the procedure, the time point is flagged. Such criteria are stored in the RANGE table by the GCOMPRT MACRO. Then the physical exam data is divided into two sets. Data to be used in the schedule header goes into a dataset with one record per patient per visit. Data used in the body of the listing goes into a dataset with one record per patient per visit per body system. Subsequently abnormality comments are merged with these data.

In printing the listing, the data in the body is processed observation-by-observation in a straight-forward manner.
However, the header and general comments are handled differently. The specifications of the listing require that the schedule header be printed at the top of each page even if a patient's listing spans more than one physical page because of the number of visits and body systems. Therefore as part of the header setup, indices are created pointing to the first and last header record for each patient and these indices are merged with the patient physical exam data. That makes possible a direct access read of the header dataset each time there is a page skip. The same technique is employed to handle general comments. The GCOMSU MACRO creates a general comments dataset and a pair of indices that is merged with the patient data. On LAST.PAT, the GCOMPRT MACRO performs a direct access read of the general comments dataset and prints all of the general comments for that patient following the body of the listing.

Summary and Future Directions

Although we have succeeded in reducing the programming effort required to perform data integrity checks by creating modular SAS MACROs to produce the data edit listings and by storing in the RANGE table integrity criteria that may change from protocol to protocol, this approach may be extended to achieve greater flexibility while further reducing repetitive programming activities. Two of the functional elements of each edit listing MACRO mentioned above are the data setup and the data integrity flag definitions. It should be possible to construct classes of data integrity definition MACROs such that the MACROs of a given class could be associated with one data setup MACRO. For example, there may be an Adverse Event setup MACRO that would join all the data necessary to process the Adverse Event data integrity checks. Then one could choose from the complete set of available Adverse Event data integrity MACROs those appropriate to the current protocol. Rather than only print the results of the checks, they could be stored in the database. Thus they would be available for analysis to detect patterns of errors so misunderstandings with investigators and/or CRAs could be corrected and CRFs could be improved.