Paper 5

SAS® RDBMS!
A GUI Approach to Data Entry Using SAS/AF®
Enforced With Relational Model Integrity Constraints from ORACLE®

Annie Guo, Ischemia Research and Education Foundation, San Francisco, California

Abstract

For the long term follow-up of an international, epidemiological study, 4000 cardiac surgery patients' responses to interview questions are recorded on interview answer sheets. The responses are processed and saved in SAS® database via a SAS/AF® data entry application. This application has the following features.

- It adopts a GUI approach to data entry using Data Form and Data Table objects in SAS/AF FRAME® software, version 6.12. The two objects provide a graphical interface and more Screen Control Language flexibility when browsing and editing data than the FSVIEW and FSEDIT procedures in SAS/FSP® software allow.

- User authentication is checked automatically upon login and a set of privileges is granted to the user accordingly. This determines whether the SAS data sets are open at editing mode for data entry or only at browsing mode.

- Integrity constraints such as primary key and foreign key inherited from an ORACLE® database are enforced into the SAS database at application level. These constraints help eliminate the need for error check after data entry.

1.0 Introduction

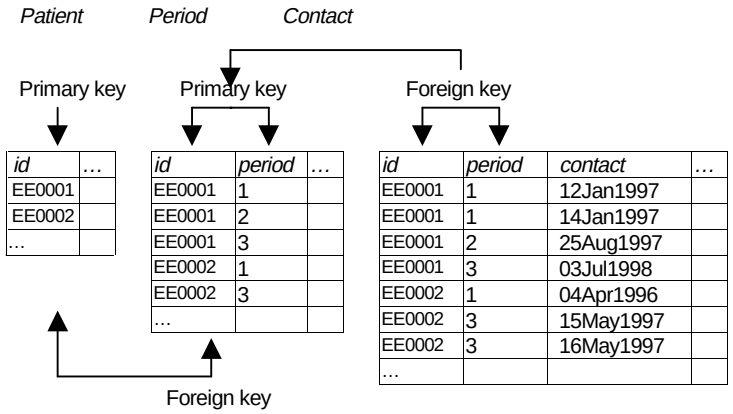
In an international, multi-center epidemiological study, 4000 cardiac surgery patients are enrolled over 3 years. The Long Term Follow-up (LTFU) of the study is designed to follow up each patient at three time points after the date of hospital discharge: 6 weeks, 6 months and 12 months. Before each time point is due, a nurse will contact the patient by phone to schedule a follow-up interview. In most cases, there are one or more contact attempts before an interview is scheduled. Sometimes a follow-up interview can not be scheduled because the patient is not well, died, etc. The result of each telephone contact attempt is recorded on interview answer sheets.

The SAS application presented in this paper has been implemented to process the data collected on the interview answer sheets for the telephone contacts.

2.0 Database Design

A database is set up to store the result of the telephone contacts at the three time points. It consists of 3 tables, *Patient*, *Period* and *Contact* as shown in the Entity Relationship Diagram (ERD) in Table 1.1.

Table 1.1: Entity Relationship Diagram (ERD) of the three sample tables *Patient*, *Period* and *Contact*.



id	…
EE0001	
EE0002	
…	

id	*period*	…
EE0001	1	
EE0001	2	
EE0001	3	
EE0002	1	
EE0002	3	
…		

id	*period*	*contact*	…
EE0001	1	12Jan1997	
EE0001	1	14Jan1997	
EE0001	2	25Aug1997	
EE0001	3	03Jul1998	
EE0002	1	04Apr1996	
EE0002	3	15May1997	
EE0002	3	16May1997	
…			

The table *Patient* stores the study patients' demographic data such as id, initials and date of birth. The first 2 letters in the id indicate the hospital code where the patient was enrolled into the study.

The table *Period* stores the information of which the data collection unit is the combination of patient and period. An example of such data type is the date when a patient was contacted at the 6-week time point. Table 1.1 shows an example of table *Period. T*he column *period* has the values 1, 2 and 3 indicating 6 Week Follow-up, 6 Month Follow-up and 12 Month Follow-up, respectively. Patient EE0002 missed the 6 month follow-up, so does not this period's record in the table.

The table *Contact* consists of detailed information of each telephone contact. In Table 1.1, the values on column *contact* are the actual dates when the nurses called the patients. There are as many records as the number of telephone contacts made to the patient at the 3 time period. For example, in Table 1.1, there are 2 records for patient EE0002's 12 month follow-up, and they are made on 15May1997 and 16May1997, respectively. In addition, within a patient and at the same time period, there can be more than one record of the same dates. This is because the patient might be contacted more than once on the same date.

3.0 Business Rules

The following rules have been identified to maintain the integrity among the 3 tables. They are illustrated in Table 1.1.

- User's roles
  Only the designated data entry technicians are authorized to edit the data. Study team members can only browse the data. Other people are not allowed to access the data.

- Primary Key Constraints
  - The values on column *id* in table *Patient* must be unique and not null.
  - The combination of the values on columns *id* and *period* in table *Period* must be unique and not null; there is at most one record per period and per patient.

- Referential Integrity Constraints
  - The values on column *id* (i.e., foreign key) in the child table *Period* must exist on column *id* (i.e., referenced key) in the parent table *Patient*.
  - The combination of the values on columns *id* and *period* in the child table *Contact* must exist on columns *id* and *period* in the parent table *Period*.

4.0 Objectives

The study's primary database is an ORACLE relational database. It includes the table *Patient* described in Section 2.0. Note that the primary key constraint on patient id described in Section 3.0 is in place in the ORACLE table.

The database for the study's Long Term Follow-up could have been integrated with the primary database in Oracle. However, because of time constraint, the SAS System has been chosen to store the data and to build an application for data entry purposes.

Two objectives of this data entry application are identified.

- Prevent the business rules described in Section 3.0 from violation during data entry process. For example, the data entry technician can not add a record to table *Contact* where the combination of id and period being added already exists.

- A GUI approach to data entry will be more efficient than a text based environment such as SAS/FSP. There are 2 reasons. Firstly, the interview answer sheets contain mostly check boxes. Secondly, there are no values printed in the boxes to instruct the data entry technicians what values the boxes checked should be transcribed to and entered to the database. In this case, a text based system is not convenient.

5.0 Design and Implementation

To build a GUI data entry environment, the Data Form and Data Table objects in SAS/AF Frame software, version 6.12 on HP-UX are used. The objects provide a graphical interface and more SCL flexibility when editing and browsing data than FSEDIT and FSVIEW procedures in SAS/FSP software.

The enforcement of the business rules to the data entry process is illustrated by Flowcharts 1, 2 and 3, and Screen Prints 1, 1A, 2, 2A, 3 and 3A.

Flowchart 1 (Screen Prints 1 and 1A, table *Patient)*

- User roles

The application resides on UNIX platform, and uses the UNIX security system to authenticate the user. The UNIX login name is captured by the SAS application to identify the user's roles. Then a set of privileges (i.e., edit and browse, browse only, or none) is granted to the user by _set_dataset_ method. Only authorized users are allowed to go further from here. Note the assumption for this user authentication mechanism is that the user who invokes the SAS application is the same person who logs on to UNIX with valid UNIX login name and password.

- Primary key constraint on column *id* in table *Patient*

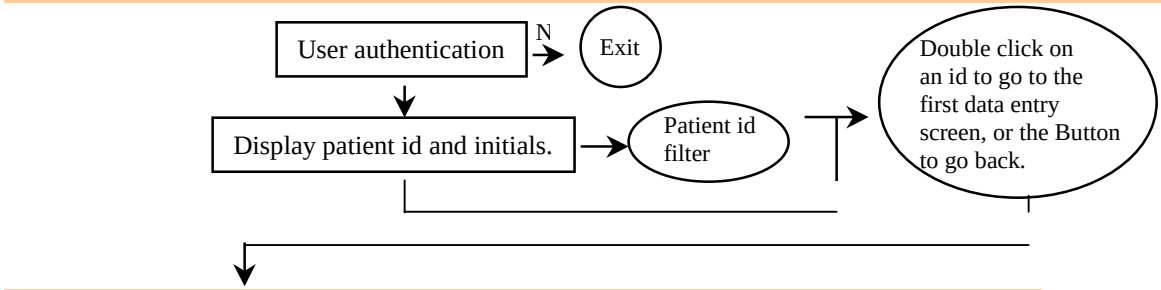
The primary key constraint on column *id* is defined in the Oracle table *Patient.* This table is converted to SAS prior to data entry. The converted data, i.e., patient id and initials are displayed through the Data Table object in SAS/Frame and are read only. Therefore the primary key constraint implemented in the ORACLE database remains intact and is enforced indirectly in the SAS table *Patient.*

- Additional feature : patient id filter

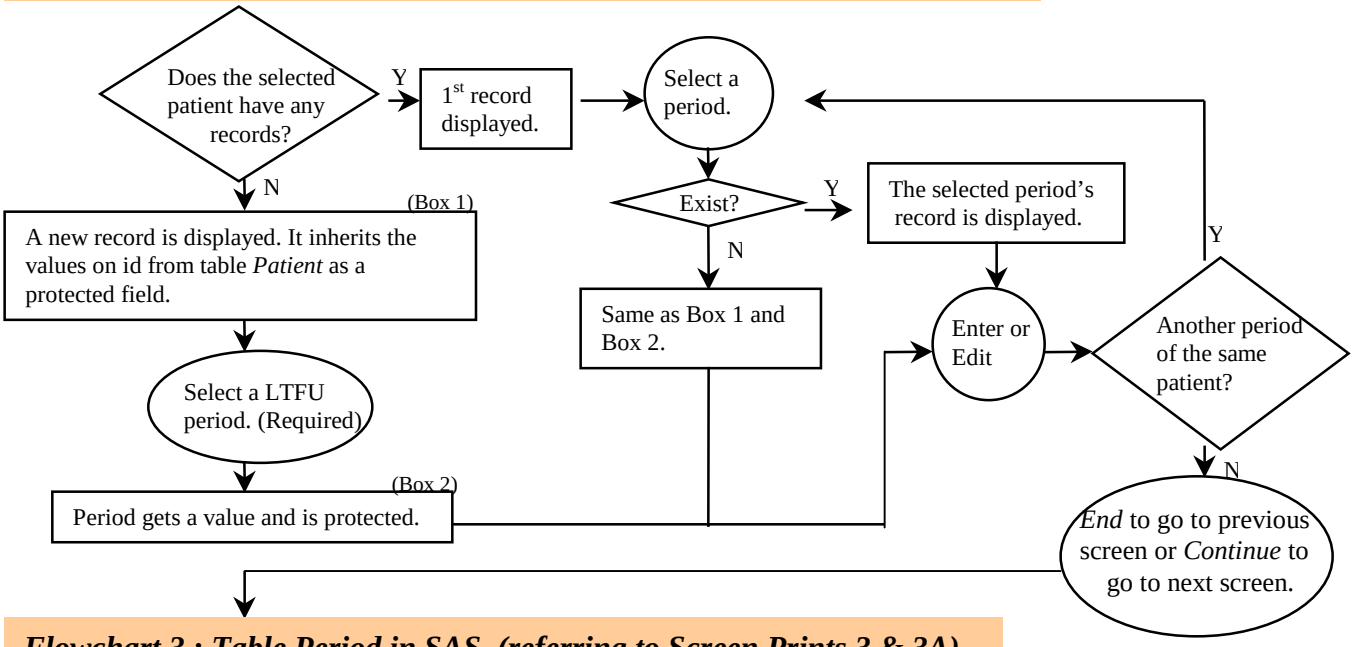
The patient id filter combines Control Box, Extended Text Entry field and Data Table to subset the 4000 patient ids displayed in the Data Table. It works in two ways. (1) To display all the patients from a specific study site, select a site code from the list of study site codes by clicking on the Control Box. The site code selected also appears in the Extended Text Entry field. (2) Enter a complete or partial patient id in the Extended Text Entry field to subset the data. For example in Screen Print 1A, the user enters 'EE000', and only those ids starting with EE000 are displayed in the Data Table.

The communication among the three classes is made through event processing. The Control Box sends the selected site code via _send_event_ method to the Extended Text Entry field. The Extended Text Entry field receives the site code via _set_event_handler method and displays it as a text. The text displayed in the Extended Text Entry field can be the site

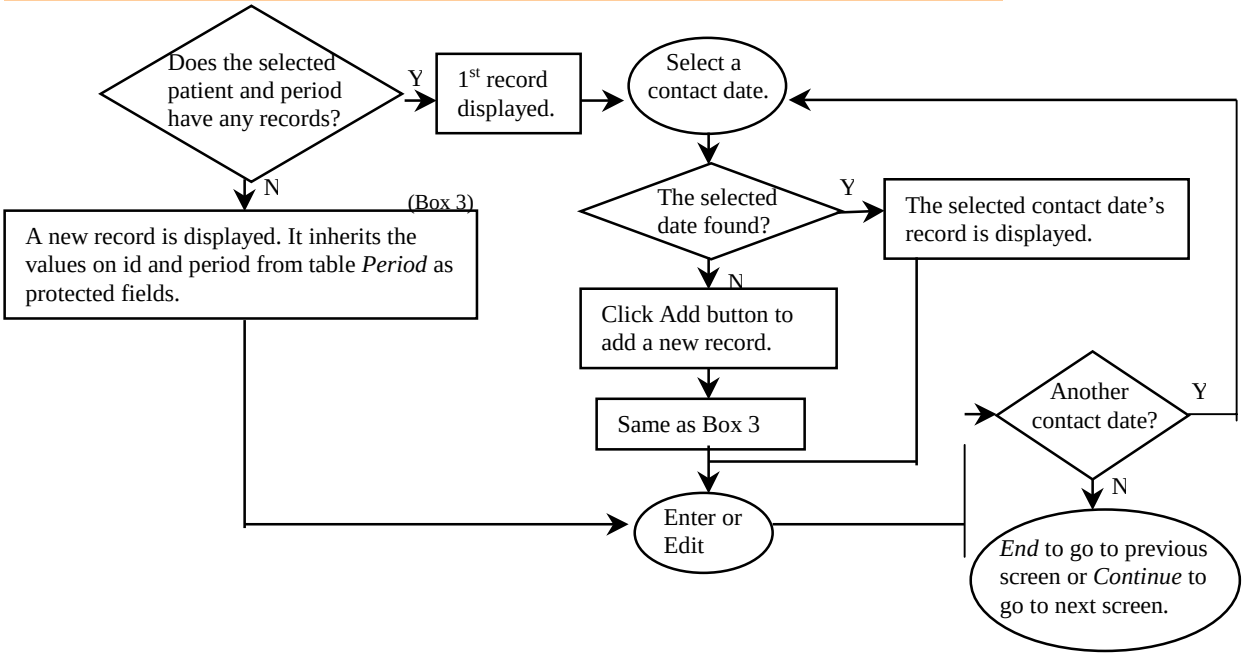
Flowchart 1 : Table Patient in SAS and ORACLE (referring to Screen Prints 1 & 1A)



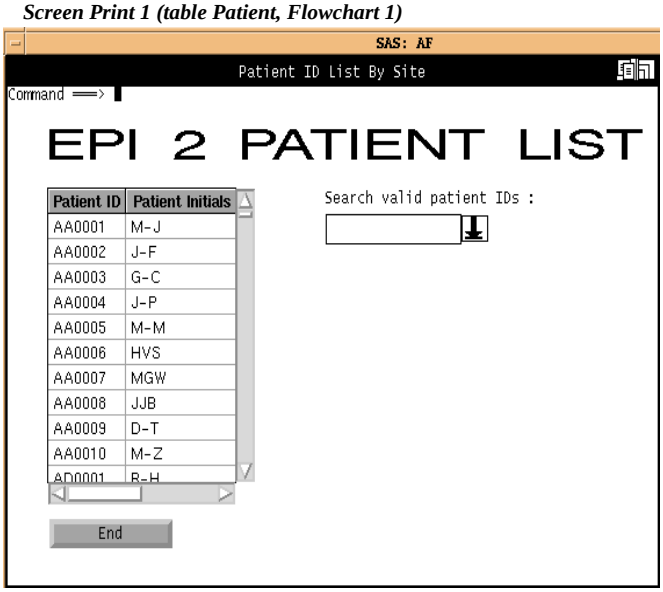
Flowchart 2: Table Contact in SAS (referring to Screen Prints 2 and 2A)



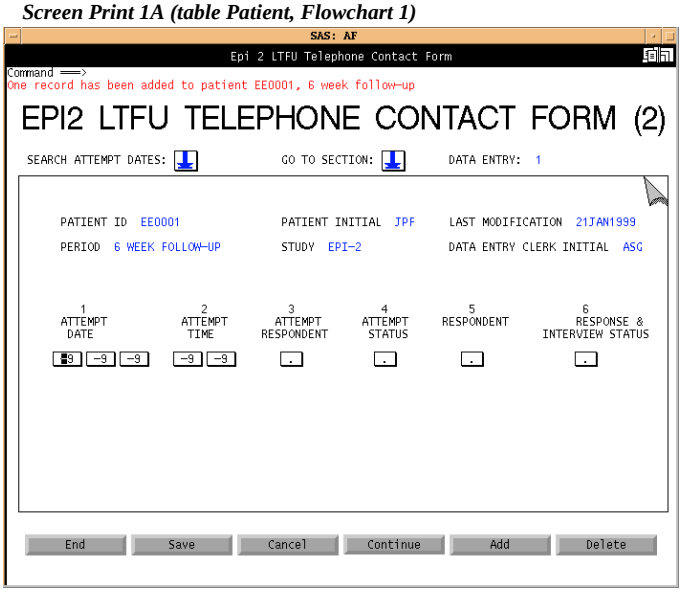
Flowchart 3 : Table Period in SAS (referring to Screen Prints 3 & 3A)



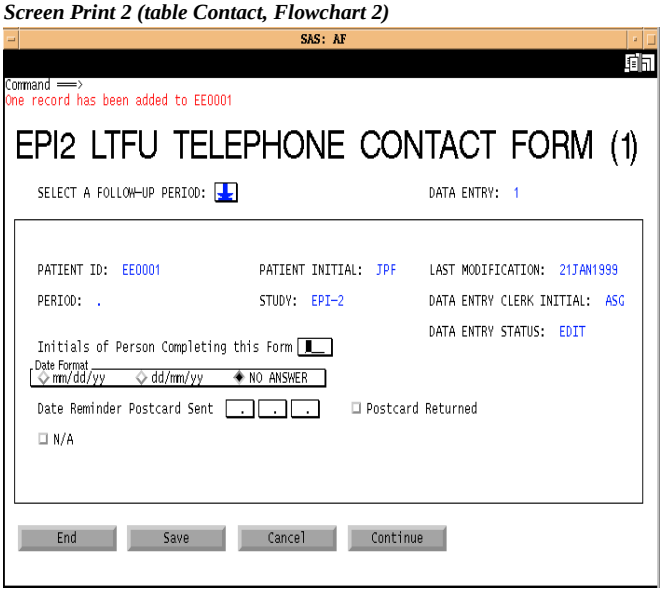
Screen Print 1 (table Patient, Flowchart 1)



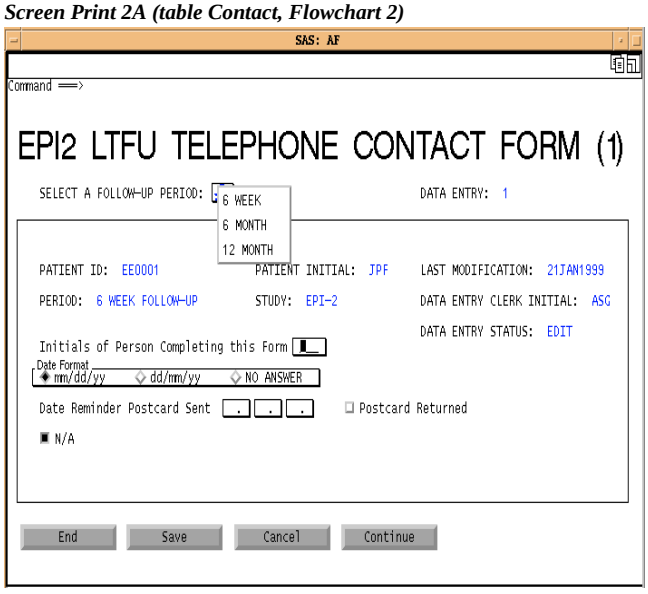
Screen Print 1A (table Patient, Flowchart 1)



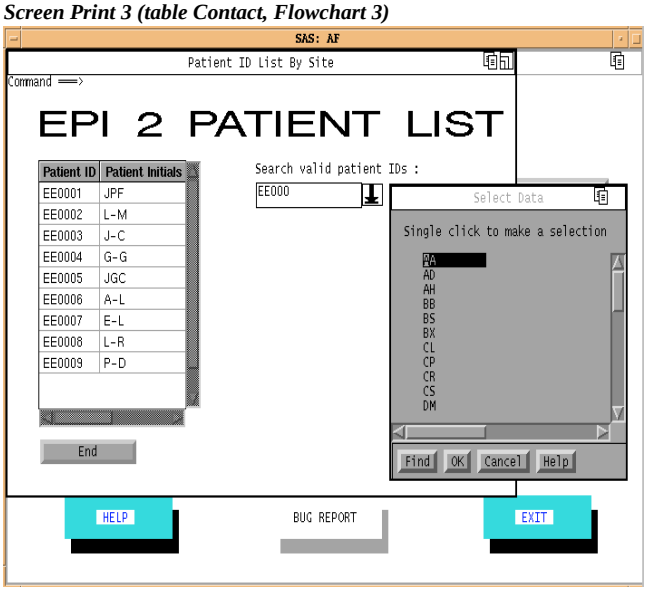
Screen Print 2 (table Contact, Flowchart 2)



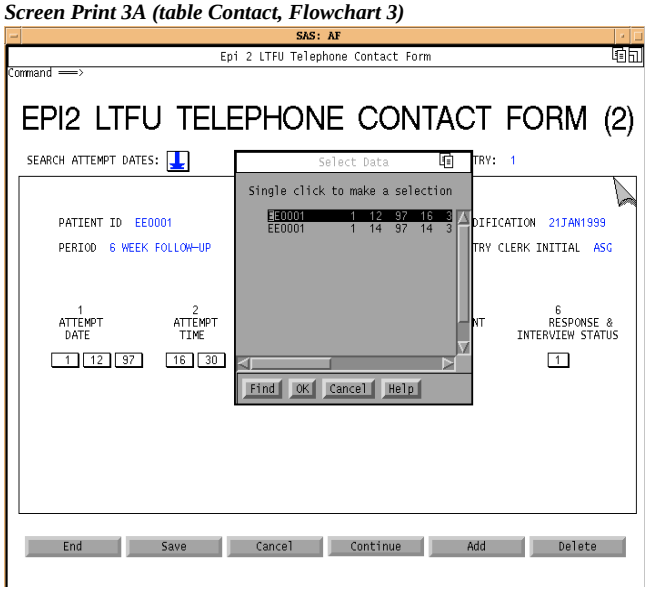
Screen Print 2A (table Contact, Flowchart 2)



Screen Print 3 (table Contact, Flowchart 3)



Screen Print 3A (table Contact, Flowchart 3)



code sent from the Control Box, or a complete or partial id entered directly by the user. It is then sent to the Data Table. The Data Table receives the text and subsets the list of patient ids accordingly.

Going from Flowchart 1 (Screen Print 1A, table *Patient*) to Flowchart 2 (Screen Print 2, table *Period*)

- Foreign key constraint on column *id* in table *Period*

The user has to double click on a desired patient id, e.g., EE0001 in Screen Print 1A, via the Data Table to access the patient's records in table *Period* in Screen Print 2. If the selected patient does not have any records in table *Period*, a new record is created and the field *id* is filled out with the value from table *Patient*. Once launching Screen Print 2, the field *id* is are protected from editing. Thus the foreign key constraint on *id* is created.

In the background, when the user double clicks on a patient id in Screen Print 1A, a custom _select_ method associated with the Data Table is executed. The method takes the selected patient id to form a WHERE clause, and then sends the WHERE clause to the Data Form in Screen Print 2. The Data Form receives the WHERE clause, subsets the data in table *Period* using _find_row_ method, and displays the first record found using _goto_row_ method or creates a new record using _add_row_ method. Note that because of the WHERE clause, if the user needs to access another patient's records, he has to go back to Screen Print 1 and then select another patient id.

Flowchart 2 (Screen Print 2 and 2A) – table *Period*

- Primary key constraint on the combination of columns *id* and *period* in table *Period*

The Control Box where it says 'Select a follow-up period' on the top of Screen Print 2A displays the 3 follow-up periods. The user clicks on one of them to add a record for the selected period, or display the period's record if it exists. For example in Screen Print 2, the very first time when patient EE0001 is added to table *Period*, the field *period* is blank until the user clicks on the Control Box and selects 6 Week Follow-up in Screen Print 2A. After that, every time the user clicks on the Control Box and selects a period, the SCL code in the background constructs a WHERE clause and searches via _find_row_ method if the combination of the selected id and period has a record in table *Period*. If it finds one, the record is displayed via _goto_row_ method. If not, a new record is created via _add_row_ method and the field *period* and *id* are filled out. In all cases, the field *period* is protected from editing. With such a mechanism the user can not enter an incorrect period value, or add a second record of any period, and thus the primary key constraint is implemented.

- Additional feature: data editing commands

To ensure the user goes through the designed route to edit data and avoid violating the rules, the popup menu of editing commands at the Data Form object is turned off. Instead, only the following customized commands necessary for the data entry process are provided as Command Push buttons.

(1) End : to save all the changes made after the last Save command was executed and go back to Screen Print 1.
(2) Save : to save all the changes made after the last Save command was executed.
(3) Cancel : to cancel all the changes made after the last Save command was executed.
(4) Continue : to save all the changes made after the last Save command was executed and go to Screen Print 3.

Note that, to prevent the referential integrity constraint between table *Period* and table *Contact*, the user is not allowed to delete an existing record from table *Period* via the application. Take Table 1.1 as an example, if the record for patient EE0001 and period 6 Week is deleted from table *Period* and the 2 records in table *Contact* for patient EE0001 and period 6 Week stay, the referential integrity constraint is violated. Therefore, the designated database manager needs to confirm this data change request, and then delete all the 3 records in order to maintain the integrity.

Going from Flowchart 2 (Screen Print 2A, table *Period*) to Flowchart 3 (Screen Print 3, table *Contact)*

- Foreign key constraint on the combination of columns *id* and *period* between tables *Period* and *Contact*

To access the table *Contact* in Screen Print 3, first of all the user must go through Screen Prints 1A and 2A to have valid patient id and follow-up period in place in table *Period*. Next, the user presses the button Continue in Screen Print 2A to go to Screen Print 3. As soon as the Continue button is clicked, SCL code checks if the field *period* in Screen Print 2A is blank. If yes, an error message is displayed and the user can not go to the next screen. If it is not blank, a WHERE clause using the selected patient id and period is constructed. The WHERE clause is passed to table *Contact* in Screen Print 3. If the combination of the id and period has one or more records in table *Contact*, the first record is displayed. If the combination of the id and period is added to table *Contact* for the very first time, e.g., patient EE0001 and period 6 Week shown in Screen Print 3, a new record is created via _add_row_ method. The values on columns *id* and *period* of the new record come directly from Screen Print 2A. The fields id and period are always protected from editing, thus the foreign key constraint is created.

Flowchart 3 (Screen Print 3 and 3A) - table *Contact*

- Additional feature : data editing commands

In addition to the 4 editing commands available in table *Period* (Screen Print 2 and 2A), Add and Delete commands are available for table *Contact* to add and delete a record, respectively. This is because there is no constraint required within table *Contact*. Thus, given a selected patient id and period, data entry technicians can add records according to interview answer sheets, and delete any records necessary as part of the data entry process.

- Additional feature : record searching

The Control Box where it says 'Search attempt dates' on the top of Screen Print 3 will display a list of entered and saved records by id, period, contact date and contact time. By clicking on one of them, e.g., EE0001, 6 Week Follow-up (value 1), contact date 1/12/97, and contact time 16:29 in Screen Print 3A, this record is displayed.

Note that, because of the WHERE clause, once launching Screen Print 3, only those records of the selected patient id and period are accessible. If the user wants to access another period's records of the currently selected patient, he has to go back to Screen Print 2 and select another period. Similarly, if the user wants to access another patient's records, he has to go all the way back to Screen Print 1 and select another patient.

6.0 Conclusion

The SAS/AF Frame application provides a GUI environment to data entry, but in terms of performance it is more resource consuming than SAS/FSP. Secondly, the business rules itemized in Section 3.0 help maintain the integrity of the SAS database. However, they are defined only at application level rather than at database level like what ORACLE can do. So when someone who knows SAS programming manipulates the data sets through direct interaction with the SAS System, these constraints are not there any more.

7.0 Reference

ORACLE7® Server Concepts Manual (1992), Oracle Corporation, CA.

SAS/AF Software: FRAME Class Dictionary, Version 6, First Edition (1995), SAS Institute Inc., NC.

Solutions @ Work ™ : Rapid Applications Development Examples (1997), SAS Institute Inc., NC.

8.0 Acknowledgements

The author wishes to thank Ischemia Research and Education Foundation for funding and support of this paper presented at SUGI 24.

Author

Annie Guo
Ischemia Research and Education Foundation
250 Executive Park Blvd. #3400, San Francisco, CA 94134
(415) 715-2300
asg@crucis.iref.org