CHAPTER 1

Introduction to Code-free Design

Problems in the application development cycle

Today's business world increasingly demands quality applications. How can an application meet future needs when the business situation constantly changes? How fast can new conditions be met? What is the cost of any change in the application? The answers to these questions depend on application development technology, which determines the quality of the application.

Here are some problems that technology must solve to improve application quality:

- Many people must be involved in the construction of a complex application, and regular communication becomes impossible because it consumes too much time.
- The useful life of an application begins when it is delivered to the user. From this point, application modifications are often required. These modifications may consume a great deal of effort in recoding and even redesigning.
- Generally, the better an application, the longer its life. Modifications and maintenance of the application are usually performed by people other than the original implementers. This problem is very difficult to solve because the application internals cannot be easily understood.

The code-free approach described in this book addresses these problems. Generally, the code-free approach addresses the most fundamental problems of existing application development technologies–the failure to recognize users' true needs and the inability to develop applications quickly to meet these needs.

Principles of code-free design

The massive proliferation of the SAS System has created new opportunities and new challenges for application designers to deliver graphical, event-driven, client/server database applications. Application user expectations for new applications include short delivery lead-times, ease of use, and increased flexibility. In response to these challenges, application designers are adopting rapid application development

techniques. Code-free design is one of these techniques. Code-free design lets the application designer describe applications visually, in terms of their functionality, and permits the development of mission-critical applications without programming.

The main principles of code-free design are

- Application design is perceived as data and nothing but data. This means that the application design is defined in a set of specially structured tables and is stored, updated, and managed in the same way as ordinary data.
- Application activities are stated in terms of what must be done but not how to do it. An application activity can be imagined as a definition of states and messages allowing transition from one state to another. Of course, the states and messages are stored, updated, and managed as ordinary data. The application designer defines full application functionality by use of its states and messages.
- The application is managed from a single control point.

How code-free design can automate application development

Automating an application development process from beginning to end means that:

- requirements feed seamlessly into design
- design automatically writes code
- code is deployed automatically into production
- maintenance changes automatically rewrite code.

Code-free design can realize this dream of end-to-end automation through the table-driven environment.

The table-driven environment is the most fundamental aspect of the code-free approach. The heart of this environment is the set of specially structured tables forming the data dictionary. The data dictionary contains a variety of information concerning application objects and operations, such as data structures, application activities, and so on. In other words, the data dictionary contains the application design, and the data dictionary programs transform it into appropriate, functional working applications. The data dictionary also provides modifications to the application according to changes in application design. Different structures of data dictionaries can be defined. The designer is a person who defines the data dictionary structure. Chapter 2 describes the data dictionary in detail.

The table-driven environment is the software tool created by the programmer. This tool supports the data dictionary architecture and provides the designer with a convenient and reliable environment for code-free application design. The table-driven environment provides the capability to determine quickly the impact of requested modifications, which can be made by the designer, programmer, or

application users. This ability enhances communication among designers, programmers, and application users and increases their productivity. This is very important because productivity depends on these people.

The role of the SAS System in code-free design

The SAS System is ideally suited to implement the code-free approach to application development.

First, SAS stores data in data sets that are very similar to tables. In turn, it can operate with tables rather than records and can implement the relational data model easily.

Second, SAS supplies tools for creating any application facility and capability that we may have envisioned, such as:

- data access and management
- powerful analysis and computation tools with distributed computing capability
- cross-platform data transfer capabilities and transparent access to files and to different databases
- client/server capabilities
- data entry and retrieval facility, along with concurrent multi-user updates to remote data
- high-resolution graphics tools.

Third, SAS is a continuously developing system that follows up-to-date achievements in hardware and software technology so that any SAS application keeps pace with advances in the computer world.

Different views on an application

The application user, application designer, and SAS programmer are all involved in the process of an application's "birth."

In order to visualize the code-free approach to application development, we have created a sample application that will be developed throughout the book. It is a database client/server application that supports clinical experiments in the pharmaceutical industry.

How a user imagines an application

Here are three types of users of the clinical experiments application:

1. The pharmaceutical chemist, who creates the new drug and wants to test it thoroughly.
2. The physician, who uses the new drug to treat patients.
3. The manager of clinical experiments, who defines treatment policy and requirements for data and analysis results quality.

These users may generate the following requirements for the application:

Pharmaceutical chemist

- The pharmaceutical chemist analyzes data, and this analysis must produce reliable results. The analysis results in such an application are very sensitive to the quality of the data. The pharmaceutical chemist wants to be able to define new and different tests of data validation and correctness each time data are updated and/or analyzed.
- The pharmaceutical chemist analyzes data from clinical experiments through several steps:
  a. Pre-analysis: This step shows data outliers.
  b. Outliers processing: This step submits correct data for the next step.
  c. Main analysis: An interpretation of the results of the main analysis may lead to additional analyses.
- Each clinical experiment requires specific analysis methods for each step. Moreover, the pharmaceutical chemist may wish to change an analysis method while an experiment is in progress.
- The pharmaceutical chemist needs access to all data for analysis purposes or to present the results of analysis to the manager.

Physician

- The physician needs a distributed, networked data entry facility so that several physicians involved in the clinical experiment can enter data simultaneously from different remote work places.
- The physician needs to be sure that any data entered are correct in terms of values and relationship to other data.

Manager

- The manager needs a very advanced reporting facility, a feature that is especially important for such an application. Because the professional's decision about a drug must be well-explained and proven, the reporting facility must have unlimited ability to generate reports (both tabular and graphic).

- The manager needs to define what kind of relationships will appear in graphical or tabular reports and must be able to change the reports' appearance and presentation of information at any time. This requirement is very important for the manager because correct and flexible data presentation can often demonstrate previously invisible data features.

To summarize the users' requirements for this application, it is a client/server application with a distributed database and with a high degree of flexibility in application features definition.

Organization of the clinical experiment

How will the clinical experiment be organized?

- The physician has to select patients for the clinical experiment. Each patient has unique characteristics (for example, identification number, etc.).
- The physician has to classify the patients according to characteristics that seem to be relevant for the drug checking (for example, age, sex, etc.)
- The manager has to define conditions of drug acceptance (for example, dosage, time intervals, and so on) for each group of patients according to their characteristics. The manager must define the type and frequency of medical trials to be done and how the treatment policy will change depending on the results of the medical trials.

How a designer defines an application

The designer's task is to design an application for clinical experiments that supports changing application requirements. Everyone who has worked with application development knows that it is an iterative process, in which requirements are constantly being refined and made more precise. To design the right application, the designer must define the application objects and operations that satisfy the requirements.

Defining objects

The main objects of our sample application are

1. *patient*
2. *medicine*
3. *trial*
4. *result*

These objects are described below.

The patient object

The ***patient object*** is represented by the Patient and Group tables. The Patient table lists patients who participate in the clinical experiment. Each row in the table is uniquely identified by a patient identification number so that we can access immediately each of the patients by his or her unique identification number. In addition, the table contains patient first names, addresses, and other related information. The columns of the Patient table are defined as follows:

Columns of the Patient table

Column name	Type	Length	Description
PATIENT	Numeric	8	Patient identification number
NAME	Character	20	First name
SURNAME	Character	20	Last name
SEX	Character	1	Sex
BIRTH	Character	10	Birth date
ADDRESS	Character	80	Address
PHONEHOM	Numeric	8	Phone number at home
PHONEWRK	Numeric	8	Phone number at work

During a clinical experiment, the Patient table is filled in with information about patients, as in the following example:

Patient table

PATIENT	NAME	SURNAME	SEX	BIRTH	ADDRESS	PHONEHOM	PHONEWRK
10001	Tanya	Green	f	09FEB1963	14/26 Red str.	9196004252	9196004254
10002	Samuel	Brown	m	25APR1956	26/15 Green str.	9196723678	9196795487
10003	Lisa	Howell	f	22MAY1961	14/27 Red str.	9196004567	9196004253
10004	John	West	m	21APR1951	14/67 Red str.	9196906756	9196004255
10005	Nancy	David	f	12DEC1968	26/18 Green str.	9196907876	9196790389
10006	Anne	Brown	f	11NOV1955	14/18 Red str.	9197806786	9196789078
10007	Rick	Weeks	m	12JAN1971	22/12 Green str.	9196789567	9198905678

The Group table collects the patients into groups, so that each patient is a member of only one group. The columns of the Group table are defined as follows:

Columns of the Group table

Column name	Type	Length	Description
GROUP	Numeric	8	Group identification number
ORDERNO	Numeric	8	Order number of the patient in the group
PATIENT	Numeric	8	Patient identification number

During a clinical experiment, the Group table is filled in with information about patient groups, such as:

Group table

GROUP	ORDERNO	PATIENT
1	1	10001
1	2	10002
2	1	10003
2	2	10004

The medicine object

The *medicine object* is represented by the Medicine and Dose tables. The Medicine table lists the medicines that are used in the clinical experiment. Each row in the table is uniquely identified by a medicine identification number so that we can immediately access each medicine by its unique identification number.

Columns of the Medicine table

Column name	Type	Length	Description
MEDICINE	Numeric	8	Medicine identification number
MEDNAME	Character	80	Medicine name

The Medicine table, with information filled in, looks like this:

Medicine table

MEDICINE	MEDNAME
1	Paracetamol
2	Bricalin
3	Histafed
4	Pramin

The Dose table contains assignments of the medicines according to groups of patients so that each medicine has a specified dose for each group.

Columns of the Dose table

Column name	Type	Length	Description
MEDICINE	Numeric	8	Medicine identification number
GROUP	Numeric	8	Group identification number
L_DOSE	Numeric	8	Lowest daily dose of the medicine for the patients' group
H_DOSE	Numeric	8	Highest daily dose of the medicine for the patients' group

The Dose table, with information filled in, looks like this:

Dose table

MEDICINE	GROUP	L_DOSE	H_DOSE
1	1	200	500
1	2	250	450
2	1	7	8
2	2	6	9
3	1	12.5	12.8
4	1	1200	1300

The trial object

The ***trial object*** is represented by the Trial and Monitor tables. The Trial table describes the trials that are applied in the clinical experiment. It contains the names and tolerance levels of the trials' results, according to trial identification number, so that we can immediately access each of the of the table rows by its unique trial identification number.

Columns of the Trial table

Column name	Type	Length	Description
TRIAL	Numeric	8	Trial identification number
TRIALNAM	Character	10	Trial name
L_NORMAL	Numeric	8	The lowest normal value of the test's result
H_NORMAL	Numeric	8	The highest normal value of the test's result

Columns of the Trial table (continued)

L_WARN	Numeric	8	The lowest warning value of the test's result; it is still normal
H_WARN	Numeric	8	The highest warning value of the test's result; it is still normal

The Trial table, with information filled in, looks like this:

Trial table

TRIAL	TRIALNAM	L_NORMAL	H_NORMAL	L_WARN	H_WARN
1	WBC	0.08	0.35	0.1	0.3
2	HRF	18.8	32.2	20.1	30.1
3	HLC	0.0	1.0	0.001	0.008
56	HGB	11.7	17.3	13.2	15.5
63	HCT	35.0	49.0	39.0	45.0
88	RBC	3.8	5.7	4.3	5.1
93	MCV	76.0	120.0	80.0	100.0

The Monitor table defines the trial policy for each patient's group. Every row in the table is uniquely identified by group and trial identification numbers.

Columns of the Monitor table

Column name	Type	Length	Description
GROUP	Numeric	8	Group identification number
TRIAL	Numeric	8	Trial identification number
REPEAT	Numeric	8	Time interval in days; it is recommended to repeat this test with the same interval for this patient's group

During a clinical experiment, the Monitor table is filled in with information about performing trials, such as:

Monitor table

GROUP	TRIAL	REPEAT
1	1	12
1	56	10
1	63	5
2	2	8
2	56	8
2	63	5

The result object

The ***result object*** is represented by the Result table. It contains trial results for each patient. The result is uniquely identified by patient and trial identification numbers and the date when the trial was performed.

Columns of the Result table

Column name	Type	Length	Description
PATIENT	Numeric	8	Patient identification number
TRIAL	Numeric	8	Trial identification number
DATE	Character	10	Date of trial
RESULT	Numeric	8	Trial result

During a clinical experiment, the Result table is filled in with information about trial results, such as:

Result table

PATIENT	TRIAL	DATE	RESULT
10001	1	10MAY1994	8.12
10002	1	10MAY1994	10.00
10003	1	10MAY1994	8.04
10001	1	17MAY1994	7.54
10002	1	17MAY1994	7.58
10003	2	23MAY1994	11.2
10004	2	23MAY1994	13.2
10005	2	22MAY1994	11.5
10001	3	12JUN1994	20.1
10002	3	12JUN1994	23.5

The objects, described above, constitute the core of the application.

Defining operations

Besides the application object's definitions, the designer needs to define operations that are allowed with the objects. For example, the Patient table lists all legal values for the PATIENT column in any other table. It will be used, for instance, for validation of the values of the PATIENT column from the Result table. We can also validate the value of the RESULT column from the Result table versus the values of the L_WARN and H_WARN columns from the Trial table. The set of such operations, together with the objects themselves, constitutes the initial pass of the application design.

Changing objects or operations

Any time the designer changes or adds objects or operations, it can be done easily with the table-driven environment. Any changes the designer makes in the design through this environment produce immediate changes in the application without any programming. This is the main principle of code-free design that ultimately provides data independence, i.e., separation of code and information. This magic environment contains a set of specially structured tables intended for the definition of application objects and operations on the one hand and the software that processes the meta data (definitions of objects and operations) from these tables on the other hand. The designer has to define the structure of the tables of table-driven environment. The same environment can be used to create different applications that can communicate with each other.

In this book, we describe possible sets of tables that constitute the table-driven environment. Consider the following example. The Location table, that belongs to such a set, lists the tables of the patient object and their locations in SAS libraries. The Location table looks like this:

Location table

TABLE	LIBRARY
Patient	patdb2
Group	patappl

If we want to move the Patient table from the current library patdb2 to another library, we just have to update this value in the LIBRARY column. The application will "know" about this change and will invoke the suitable event-driven process for performing this change. (Processes are discussed in Chapter 3.)

What the programmer does

Code-free design changes the role of the programmer. The programmer is no longer responsible for implementation of the specific application, and he or she is independent from any changes in the application's requirements or application design. Instead, the programmer builds an environment in which the specific application will be generated. The programmer receives from the designer the strictly defined architecture of the data dictionary. This architecture is described in terms of the relational data model.

In order to implement the environment for application generation, programmers need to know SAS language, SAS macro language, and many of the base SAS procedures, such as PROC SQL, PROC FORMAT, and PROC PMENU. In addition to base SAS software, programmers have to use at least the following SAS software products:

- SAS/AF, SAS/FSP, and SAS Screen Control Language—for data entry facilities and user interface generation
- SAS/ACCESS—for transparent interfaces to databases and files not in SAS
- SAS/CONNECT—for cooperative and distributed data processing
- SAS/SHARE—for concurrent access to data
- SAS/GRAPH—for graphical data presentation.
- SAS/STAT, SAS/OR, SAS/QC, and SAS/ETS—for statistical analysis, operation research and project management, quality control and improvement, time series analysis, forecasting, econometrics, and business planning.

In this book, there are many working programming examples that support the table-driven environment for application generation, such as data set generation, data entry generation, report generation, and data processing generation.

For example, the Library table, where the SAS library references are defined, can be presented like this:

Library table

LIBRARY	LOCATION
patdb2	c:\clinic\db2
patappl	c:\clinic\appl

The %LIBREF maro, which belongs to the class of reusable programs, processes data from the Library table and implements the SAS library reference definitions. This macro looks like this:

```
/*
PROGRAM      LIBREF.
DESCRIPTION  Assigns SAS library references according to the Library table
             meta data.
USAGE        %libref(libname);
PARAMETERS   libname - is the name of the library storing the library data
             set.
REQUIRES     The library data set corresponding to the Library table.
AUTHORS      T.Kolosova and S.Berestizhevsky.
*/

%macro libref (libname) ;


/*
The following DATA step creates macro variables and fills them with data
from the library data set:
libs - contains the number of libraries, defined in the library data set
lib - is a series of macro variables containing names of the libraries
loc - is a series of macro variables containing physical locations of
these libraries.
*/

         %let libs=0;
         data _null_ ;
            set &libname..Library ;
            call symput("libs", _n_) ;
            call symput("lib" || left(_n_), trim(library)) ;
            call symput("loc" || left(_n_),trim(location)) ;
         run ;


/*
The following loop generates required LIBNAME statements.
*/

         %if &libs>0 %then
         %do i = 1 %to &libs ;
            libname &&lib&i "&&loc&i" ;
         %end ;;
         %mend libref ;
```

If the %LIBREF macro reads data from the Library table, the SAS System will see these statements:

```
LIBNAME PATDB2 "C:\CLINIC\DB2";
LIBNAME PATAPPL "C:\CLINIC\APPL";
```

The %LIBREF macro is a very simple example of a program that converts data into action. In this book the programmer will find many examples of programs that use this power method.

Summary

This chapter has introduced the basic ideas of code-free design and its main features. It has analyzed the application example and shown how the user imagines the application, how the designer defines main objects of the application, and what the programmer has to do to support the table-driven environment. Most of the examples in this book use the application tables described in this chapter.

The next chapter focuses on the ideas and usage of the data dictionary, including:

- For the designer—how to use relational technology to define an application data model.
- For the programmer—how to implement software tools that support the table-driven environment.
- For the user—how to cultivate an application data model through the table-driven environment.