A PROGRESS REPORT ON THE ACTIVITIES OF THE CODASYL END USER FACILITY TASK GROUP

JUNE 1975

Chairman:	Henry C. Lefkovits	Blue Cross of Massachusetts, Inc.
Members:	Thomas H. Baumgartner	Defense Communications Agency, Joint Technical Support Activity
	J. M. de La Beaujardiere	Control Data Corporation
	John C. Clymer	Bell Telephone Laboratories, Inc.
	Joseph T. Connolly	The MITRE Corporation
	Timothy L. Grimes	Sperry-Univac
	George W. Potts	Dean, Witter and Company
	Jonas Rabin	Western Electric Engineering Research Center
	Robert S. Rosenthal	MRI Systems Corporation
	Matthew C. Schein	International Business Machines Corp.
	Harry B. Towne	Naval Command Systems Support Activity
Contributors:		
	James L. Black	Sperry Research Center
	Charles R. Blose	United Bank Service Company
	Byron Moore	Defense Communications Agency, Joint Technical Support Activity
	Michael W. O'Reilly	Bell-Northern Research
	Arla Weinert	Naval Command Systems Support Activity

CONTENTS

SECTION 1

INTRODUCTION

The purpose of this report is to document the progress which has been made by the CODASYL End User Facility Task Group since its inception and to explain the direction in which current work is proceeding. The report is addressed primarily to a segment of data processing professionals who are interested in the subject, and we are also soliciting feedback from this audience.

The CODASYL Systems Committee decided to establish an End User Facility Task Group (EUFTG) and this decision was ratified by the CODASYL Executive Committee.

The Tasking Statement given to this group is:

1. To define and describe, including functional capabilities required, one or more classes of end-users of a complex structured database.

2. To analyze relationships between complexity of language, users' view of the database, and other factors influencing total capabilities of systems satisfying the above class(es) of users.

3. To describe a set of functions with a reasonable degree of data independence which may be invoked by end users.

4. To develop the syntactic and semantic framework for one or more prototype languages for the above.

Early in its work, the Task Group decided on certain interpretations of this tasking statement which were to govern the manner in which the work was to be carried out. The interpretation of a "Complex-Structured Database" was that of a database of the DBTG-type* and the assumption also was made that a generalized database management system to manipulate this database would be available. It was also decided that the exact syntax of an End User Facility to be specified was initially of minor importance, as long as the semantics of the language were well defined and specified.

The need for a means of accessing a database by End Users has long been recognized; in fact, a specific mention of this is made in the original DBTG report, and many systems are in existence today which have addressed themselves to this problem. The Task Group, in its early days of existence, has extensively reviewed many of these efforts, and has developed certain ideas which are being presented in this progress report. Of paramount importance has been the idea that it is necessary to have a good match between the language and the user of the language; consequently, a substantial effort was made in studying characteristics of certain user classes, and in developing a definition of the class which is currently being addressed. An assumption was made that in all likelihood no single End User Facility (EUF) could satisfy the entire universe of End Users, therefore, the direction currently taken by the Task Group is intended to satisfy the requirements of only one segment of that universe. Tentative plans are to address other segments upon completion of the current phase of work.

In the course of its work, the Task Group considered a number of different approaches regarding the basic orientation which the language to be specified would take. The decision of using a forms-oriented approach was made on the basis that it appeared that such a language could have the simplicity desirable for the kind of user being addressed, deal with concepts with which the user was familiar, as well as have sufficient power to allow the user to solve non-trivial problems. For purposes of the present report then, the EUF activities are based on this decision. Whereas it does appear to the Task Group that this approach may be good in the present context, no claim is intended that this approach would be the proper one in other environments.

The basic plan on which the Task Group has worked has been that the initial effort should concentrate heavily on the definition of user-visible objects, with a lower priority being assigned initially to the specification of manipulative functions. As such, the present report will contain more material on DDL than on DML.

The remaining sections of the report present an overview of the EUF, followed by a technical discussion of some of the characteristics of the EUF, and ends by identifying some of the areas of future work of the Task Group.

The Task Group currently meets for three days every two months. Meetings are hosted in rotation by Task Group members. Persons interested in more information about the Task Group, or in possible becoming members, should contact the Chairman of the Task Group.*

*CODASYL Database Task Group Report, April 1971

* Mailing Address: P. O. Box 297, Harvard, Mass. 01451

SECTION 2

DESCRIPTION OF THE FORMS APPROACH

This section provides the rationale for the approach which the Task Group has taken for an EUF; i.e., a forms approach for the End User's perception of data. From a data processing point of view, the forms approach consists of defining virtual structures which the End User thinks of as forms, and giving him the means of manipulating data in terms of these forms.

The section includes a gross definition of potential End Users, establishes and emphasizes the need for users to perceive data in familiar terms, introduces the selected forms approach, the rationale for its selection, and the anticipated benefits of the approach.

2.1 THE END USER

The EUFTG has spent much effort attempting to classify End Users so that we might specify a facility that is appropriate for the End User class. We feel it is essential that the EUF be defined with the End User for whom it is intended clearly in mind. The data processing industry has used the term End User to cover a wide spectrum of people - our definition has focused on a specific class of End User which covers a sizeable subset of the users of data processing equipment and its output.

First, and foremost, an End User is a person. That person is engaged in a job that directs, or supports the direction of, one or more of the activities which are necessary to operate any organization that can perform useful work. Every End User is engaged in one or more of the following activities: (1) Planning - The setting of goals, the identification of requirements, and the determination of the framework within which requirements will be satisfied; (2) Direction - The assignment of particular responsibilities to insure that requirements are satisfied; (3) Execution - Using organized resources in actual operations to achieve established objectives, including positive intervention to insure satisfactory achievement; (4) Monitoring - Using available communications to maintain positive control of execution, including all methods of reporting, collecting, and maintaining information on resource status and performance; (5) Evaluation - Analysis of all activity to determine the degree to which objectives are achieved and to determine the success or failure of the organization in support of its operations. Our End User will succeed or fail as he can direct or support these processes.

Our End User is assumed competent in his job, without benefit of an EUF. He may have a specialty occupation requiring extensive education or training. We assume that the specialty is not data processing or related specialties. An End User proficient in using this EUF is not expected to become proficient in programming or any other data processing specialty; nor should he consider it an entry position to such a career. In other words, the End User we are addressing already has a well-defined job and any facility he is provided should simply aid him in performing that job.

The End User has an established need to collect and manipulate data that are directly related to his job requirements and experience. In addition, the End User will make use of relationships that exist among items of data. The world of interest to our End User is populated with objects and relationships among those objects. The data he collects describe these objects; the relationships among the data reflect a model of the relationships that exist among the objects.

Finally, we expect that the End User must be willing to understand that his view of data is defined and that there is a finite set of operations that he may invoke. That is, he must become knowledeable about the nature of his new tool; including the limits of its usefulness and the manner in which it may be used.

Some training (we believe a very modest amount) will be necessary for users to learn the pragmatics and syntax of the facility. Any training necessary to learn terminology and data relationships within perceived objects is attributable to the occupation, not to the EUF.

A basic assumption of our forms approach is that the End User is intelligent and able to do his job with or without such an EUF. We also assume that the End User is no more interested in the internal workings of data processing facilities than he is for any other facilities at his disposal; e.g., the telephone.

We feel that the End User Facility being proposed in this report has substantial utility for the above defined class of End Users. We realize that other classes of End Users will require additional End User Facilities.

2.2 NEED FOR A MEANS OF PERCEIVING DATA

There is a requirement for a well defined and effective means or methodology for users to perceive a database. This interface between the End User and the database can "make or break" an EUF, and as a result, it is considered to be a particularly vital element of any proposed system.

There is a need for an approach that allows different users of a database to have their own perception of the database and its contents. It is not enough to provide just any means for the user to perceive the database (any EUF worthy of the name must of necessity do that much) it is essential to provide a means which is effective and will apply to more than one use of the data.

Additionally, certain psychological problems associated with access to computers by unfamiliar users will be ameliorated by giving the End User a good perception of the structure and contents of the database, particularly if the mechanism used is one with which the End User is at ease.

This requirement that particular emphasis be placed upon the means of perceiving data in a database is given even added weight in the light of the fact that other efforts have generally not emphasized its importance to the development of generalized EUF's. In addition, existing systems that could make claims to being general EUF's have provided user interfaces that are similar to computer programming languages, without an effort to establish that such interfaces are the best or that they are truly effective. Similarly, the interface to the database in existing systems is usually expressed, and viewed by the user, in terms of physical storage and other data processing oriented concepts. It is believed that the recognition of the essential importance of the user perception of data and the emphasis placed upon it is one of the primary accomplishments of the EUFTG work to date.

2.3 THE SELECTED PERCEPTION APPROACH

The Forms Approach

The EUFTG has selected a forms oriented approach to satisfy the need for a perception of data. The essence of this approach is that it will be possible for an End User to think of (and perhaps see) database content as a series of two dimensional forms. (We are not referring to forms as they might be used to specify a process, or request information; we are referring to the representation of database content.) The EUF will provide for the management and manipulation of forms and groupings of forms, called files and folders. A given form will consist of static information, data items and groups of data items from the database. The approach depends upon a Database Administrator (DBA) function to establish for the End User a basic view of data which is called the perception forms. There is a guaranteed mapping between data occurrences in the database and instances of perception forms. This approach is covered in significantly more detail in Section 3, "The Technical Characteristics of the EUF".

Familiar User Concepts

When we look at an office or other place of work, such as would be occupied by prospective End Users, we would first see people engaged in all sorts of tasks in the conduct of a business or profession. We would see bookkeepers, accountants, secretaries, statisticians, etc. These potential End Users all have a common characteristic: they all are occupied a great deal of the time with pieces of paper. They handle the paper, write on it by longhand or typewriter, put it in business machines and take it out again, they read from it and copy information to other forms, they place it in envelopes and route it or carry it to others, they file it and remove it from files, they are continually supplying or referring to information with pieces of paper.

Forms, including preprinted forms, scratch paper, worksheets and letters, are the basic tools in a work environment for recording information for reference or for action. These forms are most often used to record or report some event or condition, but also are used to acknowledge, authorize, estimate, instruct, notify, request, schedule, etc.

The present business use of forms includes familiar concepts of different form types, different storage arrangements and sequences,

and the possibility of an arbitrary number of occurrences of each form type. The use includes the concept of a blank form, instructions for inserting information into a form, and the disposition of new forms.

Why would one record data on a form? Obviously, it is anticipated that someone at some time in the future will have a need for the information and will want to examine the form. That is, the form, and its content, will be identifiable and data items on one form might refer to other related forms. A worksheet is also a form over which the user exercises control over an arbitrary unit of time.

The unit of information contained on a form, or on a set of related forms is, if the forms are designed properly, a unit of information that is job related, especially in regard to all different uses of the same form within an organization. Sets of forms can very often be mapped onto organizational entities, and in some cases, onto specific jobs.

Lastly, a preprinted form is a very familiar way to establish a context for all of the information contained on the form. The form can automatically label and identify data items.

The class of End Users which we have previously defined is presently familiar with the formal and informal usage of forms. We believe that this familiarity can be exploited in an EUF to an extent beyond what is presently available in automated data systems. Therefore, we have concentrated our efforts on a forms approach to the information interface between an End User and the data processing system.

Forms As a Familiar Concept

The development of an EUF based upon the user perceiving data via a set of definable forms is a particularly natural evolutionary step. People are already very familiar with forms of all kinds. In addition, they are at ease with the use of forms as a means to organize and structure data in order to effect communication.

From the earliest time that man had to make some sort of record of his various transactions, he has used forms to record and organize such information. In our every-day life, we are constantly exposed to forms and are constantly required to use them. To write a check, to apply for a job, to record a sale, etc., each requires the completion of a form. Further, each time we complete such a form, we are using the form as a means to record and organize data so that we can effectively communicate.

The use of forms as a means to communicate has also been common in the data processing environment. Here, the communication is often between people and machines, but that does not change the nature of the act. When users wish to input data into computers, they do so via some sort of form. When they wish to extract data from a computer, they specify what they want and they implicitly or explicitly specify that they want the data organized to some format; i.e., on some sort of form.

The notion that End Users of an EUF should interact with a computer and with a form as their perception of the database is quite natural because such users are already so familiar with using forms. The only new thing about the EUFTG adoption of this notion is the recognition that the End User's perception of the database can often be effectively definable as a group of forms. That is, whether the End User is perceiving data for input to a database, as output from a database or as a stored database, each perception can and should be defined as a form, and this familiar perception can be exploited in an automated facility.

Feasibility

In the relatively short period of time in which we have investigated the forms-oriented approach to EUF's, successful efforts have been made to demonstrate the basic feasibility of the approach. Whereas the tests conducted were obviously incomplete, both in the sense that only a gross system description was available and that only a narrow range of applications was considered, it did appear that an extreme flexibility could be obtained with even a limited functionality in the system. The user could choose effective ways of solving his problems based on the manner in which he chose to think about the forms with which he was dealing. It is our belief that for purposes of the present progress report a sufficient degree of evidence of feasibility has been developed to allow us to pursue the approach with a feeling of confidence. It is at the same time evident that additional, and more conclusive tests will have to be made later in the project.

Related Benefits of a Forms Approach

The foremost benefit of the forms approach which we foresee is that it represents a vehicle for the collection, presentation, and other manipulations of information in a context and format that is most familiar and is deemed most useful to an End User. This benefit is described previously in this report.

In present database management systems, much emphasis is given to the duties and responsibilities of a DBA. This emphasis is most clearly directed toward managing a central repository of accurate and up-to-date information and making the information available to users. We agree with this centralized concept of database administration and have extended it to a level where it is a DBA's responsibility to present a view of information that is likely to be independent of the view available to data processing personnel. Because the forms concepts are built upon the facilities of a generalized database management system (DBMS), they will maintain the desirable features of such a system. In particular, the forms concept maintains the benefits of central control and storage facilities for a database, and the means to distribute that information. The forms approach will have distinct advantages for the DBA. His job is to help End Users effectively communicate with the database. The use of forms will ease the communications between the End User and the DBA. In addition, the forms approach provides the opportunity to insure that a focal point exists, and for the DBA to become the focal point of control for all information in the corporation whether or not it is computerized.

We believe that the concepts and objects contained in the forms approach to data management can be easily perceived and manipulated by an End User, without recourse to programmers or others knowledgeable in DBMS usage. Also, we believe that the amount of knowledge necessary to use such a system represents only a minor addition to the knowledge necessary for the user to successfully perform his job.

Because of the simplicity and ease of understanding the contents of a database by using forms, there will be a growth or widening of the number of people in the organization who are able to take advantage of stored data. Data which is today informally compiled in an unstructured way and then used for decisions or action can be managed in a more organized way because of the ease of doing so via the familiar form concept.

Another major benefit of the forms approach is the fact that since the user will be able to access and manipulate corporate data utilizing concepts that are already familiar, the need for an intermediary to translate his request for corporate data is reduced. This provides the information in a timely manner and, more importantly, the request is not subject to misinterpretation by the intermediary who today must have an application program written to satisfy the End User request. That is, the opportunities to induce errors of misunderstanding are reduced.

A final benefit of the forms approach is that the need for voluminous reports on a periodic basis should be reduced since the End User can access the corporate data in a timely manner and can selectively get the specific data in which he is interested in a format similar to a report.

2.4 SUMMARY

Our End User is a person whose task will be eased by access to a tool which adopts an existing and a natural perception of data and permits access to data managed by a DBMS.

Our End User is generally a non-data processing oriented person and his acceptance of the tool is a primary goal. The acceptance depends on a minimal requirement for adaptation and training; hence, the forms approach which deals with data in the user's own media and terminology. (A bank balance statement or a personnel history form are examples both of the user's media and of the appearance that forms may take.)

We have observed that the approach appears feasible; it can interface to a generalized DBMS; it will have no unusual impact on a host system, and it provides the least traumatic impact upon a potential user that we can imagine. We believe that these attributes can offer a high degree of End User acceptance.

SECTION 3

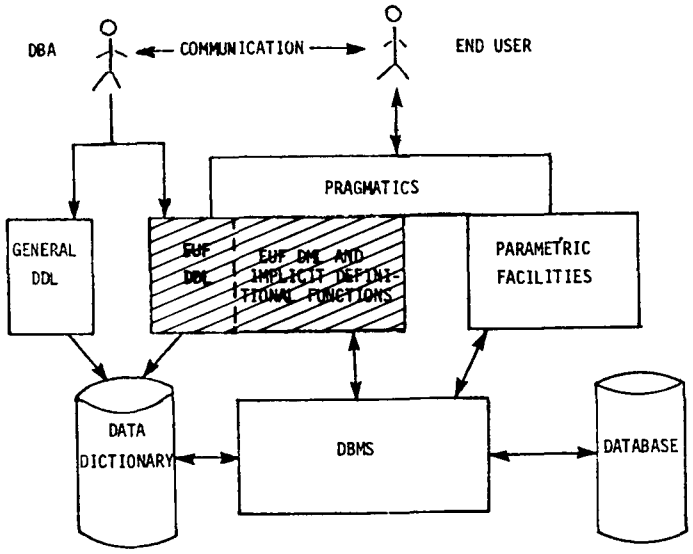
TECHNICAL CHARACTERISTICS OF THE END USER FACILITY

This section contains an architectural overview and some preliminary technical descriptions that are precursors to specification work. The material should be understood as an indication of direction established so far by the EUFTG and should not be evaluated as if it were being presented as EUF Specifications.

3.1 TECHNICAL OVERVIEW

The EUF being defined by the Task Group will exist within a software configuration that is at least a combination of an operating system and a DBMS. The EUF is not capable of operating in a stand-alone environment; various dependencies defined below must be resolved if the Task Group's work is to be implemented as a functioning facility.

The conceptual environment of the EUF is diagrammed in Figure 1 and is explained below. The EUF per se is shown as the shaded portion of the diagram.



Note: Arrows denote relationships, not specific data or control flow.

Figure 1.

Pragmatics

The rectangle labeled pragmatics represents the interface to the human End User. The EUFTG has not attempted to include pragmatics as a part of what is being defined. Therefore, it is necessary to assume that such a facility will exist as part of the environment in which the EUF exists. Pragmatics is responsible for handling the dialog between the user and the EUF. It does error checking, it aids the user in using the facility and provides a bridge to parametric functions (described below).

We assume that the pragmatics converts a users manipulative request to a canonical representation which is sent to the EUF. The user request in some instances is directly convertible to the canonical form; in other cases pragmatics would use dialog facilities to acquire the necessary information.

Pragmatics is responsible for providing human factored output information, be that error or prompting type information or responses to requests for information. Pragmatics is responsible for providing a bridge to device dependent code and tailoring the human interface to available devices and communications equipment.

We recognize that pragmatics is a broad and complex area with non-trivial responsibilities. However, it is not being addressed except in the sense of providing requirements for it. Pragmatics will be implementor defined based on those requirements.

The canonical representation for communication between pragmatics and the EUF for handling input requests and data as well as for the presentation of output information is what determines what capabilities the user has, and it is a function of the Task Group to specify the semantics of this canonical representation. The EUFTG will not "human factor" the canonical representation. Input requests are described later in this chapter under the heading of DML.

Parametric Functions

Parametric functions are prewritten and compiled application programs. The EUFTG is not addressing the problem of creating applications. We recognize the often fuzzy line between End User use of previously written applications and the direct usage of an "interpretive" EUF. It will be normal for an individual to be accustomed to using both types of function, perhaps even in an interleaved fashion to solve a problem. It is a responsibility of the pragmatics described above to provide a consistency to an End User regardless of whether

he is using the EUF or parametric facilities. However, the latter, as indicated earlier, are not being defined by the EUFTG.

Data Dictionary

Data dictionary functions have been described in many industry forums. The EUFTG concepts include the functions generally described. In the assumed EUF architecture, the dictionary would provide management of the object classes described by declarative facilities that are relevant to the EUF. The dictionary provides management of other required system objects (e.g., user profiles) and handles other characteristics of the system such as defaults.

Specific dependencies of the EUF on the data dictionary include name resolution to objects, the ability to present information about database data to the EUF for the user, and the ability of the EUF to update dictionary content.

The data dictionary in the sense of the EUFTG assumed architecture is the logical container/manager of all information declared to the system that is required so that the EUF can function.

Database Administrator

The EUF depends upon the DBA function as shown in Figure 1; there is both human communication between the DBA and End Users, and communication within the system as DBA provided descriptors are used, via the data dictionary, to provide EUF functions. The role of the DBA is discussed in 3.2.

Database Management System

The DBMS is the most obvious and important dependency of the EUF. It is however, the piece of the total environment that is potentially the most complex and confusing to the End User if it is visible. In the EUF architecture, the End User sees information in the forms arranged and specified by the DBA. These forms are called perception forms, and are illustrated in Figure 2. By definition, occurrences of data in perception forms are mappable from and to the database schema. It is critical to recognize that the mapping is not visible to the end user. The EUFTG has not yet defined mapping requirements, however no insurmountable problems have been identified in this area.

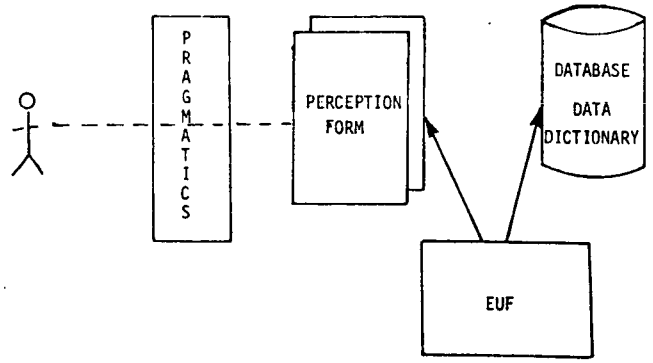


Figure 2. End User perception of database is visible as perception forms defined by DBA and controlled by the EUF via the pragmatics interface.

End User Facility

Given the environment described above and illustrated with Figures 1 and 2, it is now possible to discuss the EUF per se.

The EUF, being defined by the EUFTG has descriptive and manipulative aspects (a DDL and a DML).

The DDL is a facility for use by a DBA to explicitly create form descriptions and define other objects for an End User. The EUF DDL may be assumed to be closely related to the DDL used to describe the database schema; it is envisioned as a stand alone declarative language with the ability to map between perception forms and the DBMS schema.

The DML is the functional execution interface of the EUF. When defined, it will contain specific operators and operands that relate to EUF defined objects to carry out End User requests and provide such implicit definition capability as required. The EUF itself then is primarily executable code that interprets DML operators and interfaces with the DBMS to accomplish the desired results.

3.2 DBA ROLE

The type of responsibilities generally assigned to the DBA function are in the area of custodianship of the database, including control over access to it and maintenance of its integrity. In an environment where an End User facility of the type being discussed in this report exists these responsibilities will extend to the performance of specific functions related to the EUF and its interface to the user.

Since the End User's perception of information and the contents of the database is by means of forms, the DBA function will be responsible for appraising the End User's need for information and assuring that the proper forms are defined to the system and made available to the users. Depending on individual circumstances these forms may be further organized into folders as well as files.

In the performance of the above the DBA function will be the prime user of the DDL specified for the EUF.

The DBA function will be responsible for providing explicitly the mappings between the DBMS schema and perception form definitions.

The DBA function will be responsible for the maintenance of the content of the data dictionary including the user profiles.

In the case where some End Users are given update permission for specified items in the database the DBA function will be responsible for the determination of what update actions via which forms do not compromise the integrity of the database, and the setting up of the proper controls in the forms definitions to assure adherence to these safeguards.

3.3 PERCEPTION FORMS AND OTHER FORMS

A form is an EUF object class that represents an arbitrary but definable unit of information that is deemed the most useful unit for a user or group of users. A more complete definition of forms (and other EUF object classes) will be found in Section 3.4. In the EUF being defined, forms are the way users perceive and manipulate data.

There are three subclasses of forms we have named - Perception Forms, User Forms and Worksheets. Perception form definition is a DBA function via the EUF DDL. User forms are definable via both the EUF DDL and implicitly by ordinary DML operators. Worksheets are only definable implicitly by DML operators.

The following are definitions for the subclasses of forms.

Perception Form - The data structure and relationships (information) as perceived by the End User which is mappable to the database as defined by the DBA. The perception form is one of the means by which the DBA communicates to the End User. Control of database update resides in the definition of perception forms. All updating of the database by the End User is eventually effected using the perception form; i.e., the End User might make his update on another subclass of form but this must be mapped by the EUF through the perception form to cause alteration of the database in order that the integrity and consistency of the database are maintained. It is assumed that perception forms of different types are defined by the DBA, since the definition of a perception form type requires knowledge of the database structure. For each perception form type (e.g., invoice forms) multiple occurrences may be materialized from the database. Each perception form occurrence for a type has the same structure. The geometry of the perception form is associatable with the database structure. The EUF is responsible for maintaining control of the perception form.

User Form - The mechanism for materializing data and relationships (information) for the End User. The user form is mappable to and from the perception form. The user form is not necessarily directly mappable to the database; i.e., the user form maps to the perception form which is mappable to the database. The definition of a user form type does not require knowledge of the database structure - knowledge of the structure of the perception forms to which the user form maps to is required. Thus, user form types are definable by the End User as well as the DBA. For each user form type, multiple occurrences may be materialized using the perception form occurrences. Each user form occurrence of a type has the same structure. The geometry of the user form is not associated with the database structure. The EUF is responsible for maintaining control of the structure of the user form.

A user form may exist in one or more of three states. These different states denote changes in time and usage but not changes in definition. The three states are:

Intermediate State - A user form as maintained by the system - subject to access by the End User for presentation; i.e., output from or input to the system.

Output State - A user form as presented by the system to the End User.

Input State - A user form available for capturing information for the system to maintain.

Worksheet - The mechanism for the End User to extract information from other forms and operate on this information (scratchpad). The worksheet is defined by the user - the definition is not maintained by the EUF. The information on a worksheet is temporary and may not be directly used to alter the database; i.e., the End User must utilize a user form (the definition of which the EUF is aware) or a perception form directly to effect a change to the database. For each worksheet form type, only one occurrence can be generated by the user. The geometry of the worksheet form is totally under the control of the End User; i.e., the EUF is not responsible for - in fact, it is not aware of - maintaining control of the structure of the worksheet form.

> Note: The user is not aware of the transition of information from one state or subclass of form, to another; indeed, the user may not need to be aware that more than one category of form exists. In the implicit forms definition process, the End User is not cognizant of the distinction between a DDL and a DML.

Figure 3 shows the relationship and potential mappings among the different form subclasses and the database.

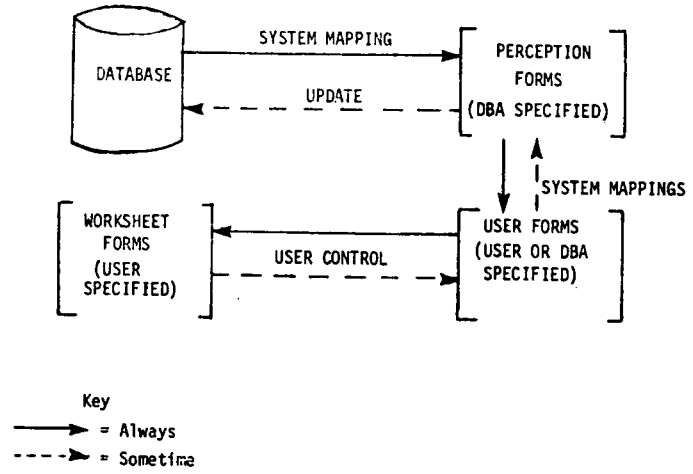


Figure 3

3.4 EUF DDL AND OBJECTS

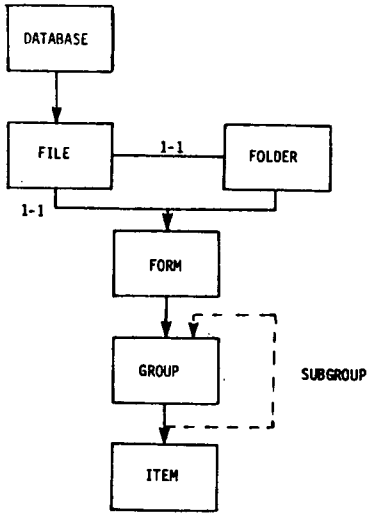
The EUF DDL will include the ability to define classes of EUF objects and their attributes and to describe the mapping of one special subclass of one of the objects (the perception form) to the DBMS schema.

The primary user of the DDL is the database administration function. (End Users may also create certain types of objects implicitly and explicitly using the DML).

The objects describable by the DBA allow an End User to conceptualize data as form types and collections of form type instances. The object classes are:

Database
File
Folder
Form
Group
Item

The objects and the relationships between them are depicted in Figure 4.



- A database is composed of one or more files.
- A file is composed of either a form or a folder, but in either case only one type of form or folder is permitted.
- A folder is composed of one or more types of forms.
- A form is composed of one or more groups and/or items.
- A group is composed of one or more other groups (i.e., subgroups) and/or items.

Figure 4. Object Classes and Their Relationships

All occurrences of these objects exist as named objects to be referenced and manipulated by a user. The terms "form type" and "form name" are synonymous. The view available to the user includes the possibility of more than one instance of each object type. An instance is an occurrence of data value(s) that conforms to an object type definition. For the form object, different subclasses exist as defined in 3.3.

The DDL is the means of naming and identifying the characteristics of each object and grouping like instances under object type names. All of the relationships that exist among objects are likewise defined in the DDL.

The object descriptions here have been developed and are described as they apply to their usage with two particular subclasses of forms - the perception form and user form, defined by the DBA function explicitly with the DDL. Other objects may have different rules established for them and may be defined implicitly via End User manipulative functions.

Database

The database is perceived only as the central repository of company data. For the user it is known to exist, and that its content, organization and accuracy are the responsibility of a DBA.

Files

A file is analogous to the container in which forms are kept. It is the largest data aggregate for which data will in any way be related for one user or for a group of users working on closely related tasks. In a large organization it is similar to maintaining files by functional unit; i.e., accounting files, budgeting files, personnel files, etc. The different files in a data structure most clearly parallel the different elements of the organization.

A file is defined as a named collection of data retrieved or derived from the database. It contains instances of one form type or optionally one folder type. The technical implications of allowing more than one form type or folder type have not yet been thoroughly explored. The objects in a file may be ordered (based on item values in the objects) and selection criteria may be applied at the file level to restrict access or reduce volume for users or groups of users.

The EUFTG has not decided if objects in a file must be unique in the file. We agree that non-uniqueness is desirable in the file, but recognize the technical difficulties encountered in accessing non-unique objects.

Forms

A form represents an arbitrary but definable unit of information that is deemed the most useful unit for a user, or group of users. That unit of information is independent of the organization of the database. Instances of a form are materialized from the database in either of two ways:

(1) as a set of one or more form instances, for one type of form, as if they were arranged in a pile; and

(2) as one of the constituent parts of a folder.

A form definition is the named collection of all groups, items and form geometry that will be used to construct a representation of a form. The definition includes: (1) a list of items and groups; (2) the relationship between groups and items on a form (including the definition of a form instance); (3) algorithms to define derived items; (4) ordering within a form; (4) geometrical rules; (6) selection criteria to limit the existance of an instance of an item or group on a form; and (7) an indication of whether or not a database update is permitted from the form.

Subclasses of forms were defined in 3.3.

Folders

A folder is an arrangement (of instances) of one or more form types grouped together by identifying, or other criteria. For example, in a personnel file it may be desirable to gather together all personnel forms and arrange them in folders, each folder containing all of the forms pertaining to one employee, or, alternately, each folder containing all of the forms pertaining to one department.

The primary rule of folder construction is that an algorithm must exist to uniquely identify the folder that shall contain each instance of a form. Naturally, the algorithm might be different for each form type in the folder.

A folder definition consists of a list of form names that will comprise the folder, implicit or explicit identification of folder tab contents, and reference to folder composition rules.

The folder tab is a unique identifier that applies to the content of an entire folder. A tab value will serve to identify and sequence folders. The value of a tab may be assigned from any item value contained in any form in the folder or it may be assigned from a defined range of item values. For example, a personnel folder might be tabbed by employee name or number so that one folder would exist for each unique employee. Alternately, the personnel folders might be tabbed by a range of dates, such as, anticipated retirement date.

The algorithm(s) that determine the composition of folders must be included as part of the folder definition. The algorithms may include uniqueness, sameness, algebraic tests, sequence tests, etc.

A folder defined in terms of perception forms is a perception folder that follows rules known to the system and are considered part of the users view. Once such perception folders are defined by a DBA, they can be altered by (manual) insertion or deletion, only if the alteration meets the definitional criteria for the folder.

Groups

A group is a collection of related items appearing on a form. The group establishes a context and meaning for each item. For example, an invoice form might include a group of items for quantity, description, unit price, and price. Each of these items are meaningful only in the context established for the group and whenever they occur together. A group might be expected to occur only once on a form or it may repeat up to a limit established for the form.

Groups are defined as a named listing of related items participating in the group. A group definition may include other groups. Item names are unique within a group.

Items

An item is the lowest level of named data known to the end user. It may exist in the database or otherwise be derived.

All of the items to be used directly or indirectly in a form must be defined. For perception items the DBA names the item and its links to the database (often a DBMS schema item name).

Implicitly or explicitly the definition could include, for example: (1) alternate item names; (2) size; (3) type of data (alphabetic, numeric, array, etc.); (4) validation criteria (for updatable items); (5) editing criteria (two way editing for updatable items); and (6) units.

3.5 EUF DML AND NAMING CONVENTIONS

The EUFTG has so far emphasized work on EUF concepts and the EUF DDL, and not on the EUF DML. It seemed natural to us that the topics previously reported upon in this document be well understood before any detailed attempt to work on specific DML semantics would be profitable. However, the directional statements below can be made about the DML.

DML

It is believed that it will be possible for implementors to present the DML via pragmatics as an extensible set of dialects, each driven by a stored user profile. The DML in a canonical form is the mechanism by which a user conveys his wishes to the EUF. The EUFTG is not interested in the exact syntax or semantics of the externalized DML as presented by the pragmatics.

Naming Convention

End Users will be dealing with all five classes (item, group, form, folder, file) of data objects. They will need naming/selection capabilities that mingle object types and content-based occurrence selection within and across object classes and within and across types and within and across occurrences. This requirement seems very complex, but we believe it represents the key to making the EUFTG language appear natural to the end user.

In a database, an arbitrary number of files can exist. Within each file, an arbitrary number of folders can exist or an arbitrary number of forms can exist. For each form, an arbitrary number of groups can exist, and so on.

Our tentative concept is that the combination of an object type and a value existing one "level" down is an identifier of a particular data object instance. For example, MR. SMITH'S INVOICE from JANUARY is an operand which identifies an occurrence (MR. SMITH), and a form type (INVOICE) within a file type (JANUARY).

There is a heavy dependency on a data dictionary capability to resolve these kinds of operands.

A general format for an operand is as follows:

Operand-part1 OF Operand-part2 OF Operand-part3 ---

(a) OF could be replaced by noise words like FROM or WITHIN

(b) Nesting rules must follow the object class hierarchy

(c) Levels of nesting can be skipped or omitted.

(d) A value can appear at any point in the hierarchy; the use of such a value is like an embedded identifier in a record. Additionally, each operand part can be qualified by an ordinal number - FIRST, SECOND, THIRD ..., LAST, PRIOR, and NEXT.

Within a form type, there is a geometric layout. Areas of the form can be named. An "empty" form occurrence of a type will show the names of the areas on the form. Optionally, space permitting, the names of the areas may remain on the form.

DML Operators

During the course of EUFTG discussions many potential operators have been discussed with no real attempt having been made to select a set. The operations described in the appendix suggest the types of operators the EUFTG intends to define.

SECTION 4

OUTLINE OF FUTURE WORK

This report is the first progress report of the EUFTG work. Without doubt, the reader will realize that the report describes and discusses how the EUFTG arrived at the approach that is to be the basis for a proposed EUF and it only briefly discusses this EUF. Because of this, it might be said that the report leaves unsaid more than it says.

It is true that there is more work remaining than that which has been accomplished so far. Therefore, this section of the report is intended to identify some of the areas in which our future efforts are expected to concentrate.

In addition to publicizing our future work plans it is our hope that this section of the report will serve to minimize or avoid unnecessary duplication of work since there are other groups at work in areas that are closely related to what the EUFTG is doing. We hope we will be advised when others have already achieved results that we may be able to use.

Another purpose of this section of the report is to "spark-the-interest" of those among its readers who have the interest and time to assist the EUFTG by seeking active membership.

There are many technical areas relating to the specification of a forms based EUF that will require work by the EUFTG. Some of the major areas are:

- Following the development and definition of "perception" objects of the EUF, we must begin to identify the kinds of constructs that are considered to be user objects. For instance, we believe it will be necessary to define such user objects as user forms, user files, user folders, etc.

- Further study and specification of:

  Semantics of the EUF DDL and DML,
  Syntax of the EUF DDL, and
  Illustrative syntax of the EUF DML

- Establish what it means to update, what can be updated and who may perform updates. In addition, the procedural aspects of performing updates must be addressed.

- Establish as comprehensive a set of requirements as we can for the pragmatics. That is, we plan to identify those aspects of the pragmatics that are dependent upon the particular nature and design of the EUF and to establish the requirements such aspects must satisfy in order for there to be effective communication between the End User and the EUF. It is not the plan of the EUFTG to actually specify these pragmatics since they are deemed to be primarily external to the EUF itself (reference Figure 1).

- Define the class of permissible mappings that will be necessary for relating the various objects of the EUF within themselves and to the DBMS schema.

- Determine requirements for the functions and contents of the data dictionary, including user profiles.

- Evaluate the notions of "escape" and "extensibility". The questions that must be answered are related to how much procedure a user can execute after a form is available (i.e., retrieved from the database) and how such procedures can be executed. For instance, will a user be able to obtain a form from the database and then "escape" from the EUF to perform non-EUF procedures on it? Will he be able to add "verbs" to the EUF as a means to operate on forms he has obtained via the basic EUF?

- Consider the security and privacy requirements in a forms based EUF environment.

- Consider DBMS requirements, if any, in a forms based EUF environment relating to integrity, recoverability, and locking.

These are the areas of work that the EUFTG plans to emphasize. We believe they are reasonably well formulated so that we can look forward to significant progress. We welcome the interest and help of all who believe they can make a contribution to this work.

APPENDIX

FORMS APPROACH BANKING EXAMPLE

The following material has been developed to illustrate the perception and manipulation of data by an End User. A banking example has been developed by the Task Group in discussing a user's perception of data.

The data structure and content of the banking database have been simplified compared to a real world banking environment. We have assumed a unit banking environment and have made a reduction in the number of data items in the database.

The banking example consists of a description of the data environment (database and perception forms) in which an End User would operate, and a list of possible ad-hoc queries that are pertinent to various banking End Users. Associated with each query is the identification of the information contents in terms of perception forms, which the Task Group has determined to be sufficient to solve the problem.

DATA ENVIRONMENT

Figure A-1 is a data structure diagram of the bank database, identifying the units of data available and illustrating the data relationships. Two relationships may not be self-explanatory. The account relationship record defines a legal association for one account among two or more persons. One person is designated the primary owner of the account; others are secondary owners. The customer relationship record defines other associations among two or more bank customers. The association (e.g., family or business relationship) is between customers and not accounts.

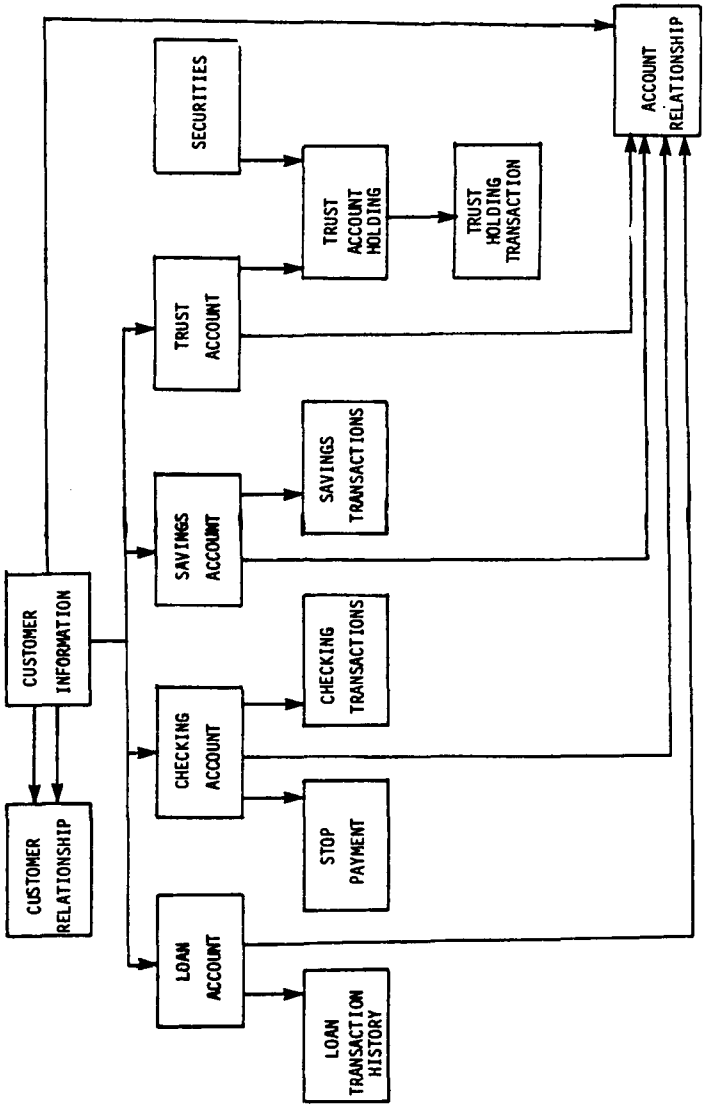
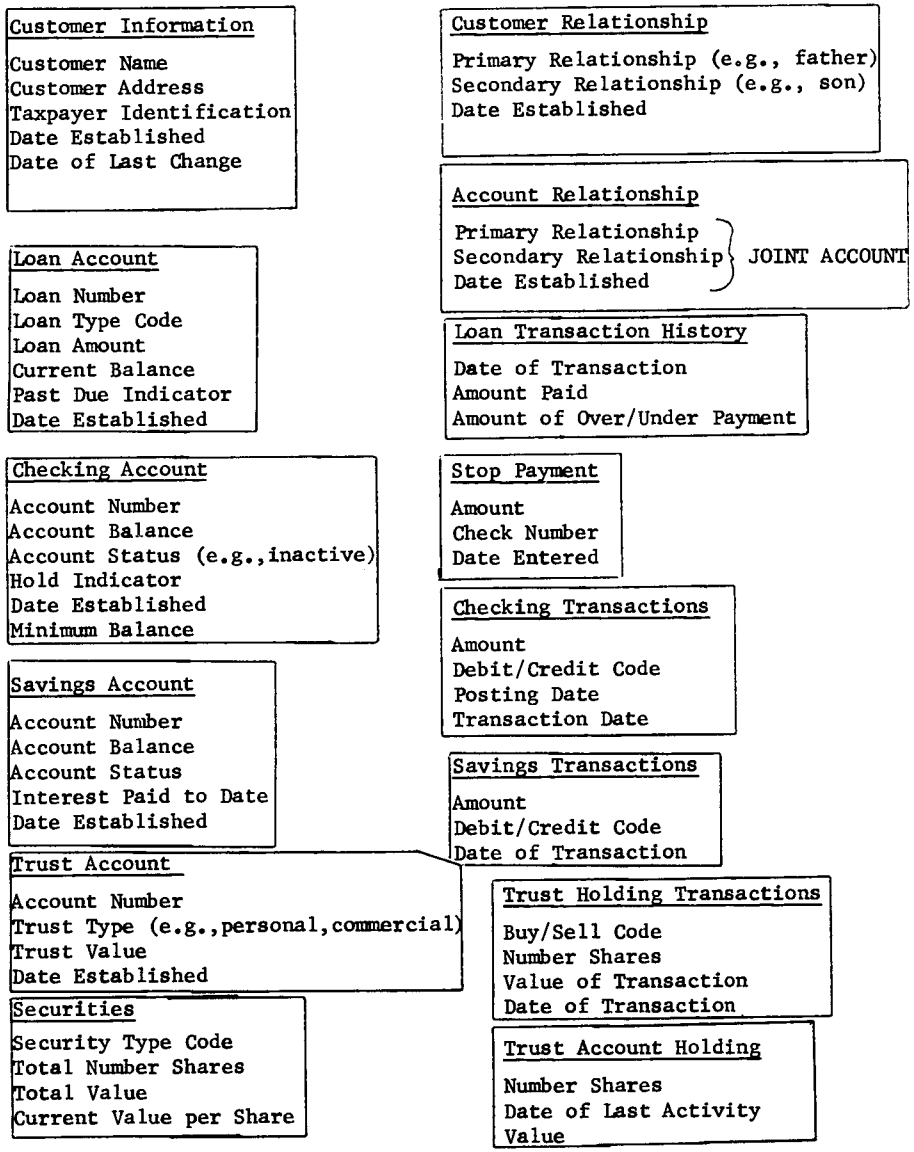


Figure A-1. Bank Data Structure Diagram

The contents of the database records are as follows:



Instances of seven different perception forms are illustrated in the following figures. It is presumed that the pragmatics will permit the entire form, as pictured, to be available to the End User, and that the pragmatics will permit continuation forms. In each figure, static information which appears on each instance of a form is distinguished from data made available from the database. These seven perception forms available to the End User are illustrated in Figure A-2 through A-8. These represent the user's view of the database and not specific output to the queries discussed here. The first five forms are very similar to account statements periodically made available to customers by most banks. The last two forms identify relationships among accounts and customers and are for use by bank personnel.

QUERIES

Query 1: For a given checking account, display all stop payments showing the amount.

This query requires reference to the checking account statement form.

Query 2: For a given trust account, display all securities held by that account, the current value of each holding and the last month's transactions (showing buy/sell, number of shares, value and date of transaction) for each holding.

The query is answered by reference to the trust account holding form and the associated trust activity forms.

Query 3: For a given checking account, savings account or loan account number, display the transaction history for the account showing amount, date and debit/credit for all transactions for the last two weeks.

This query can be answered directly by reference to the checking, savings or loan account perception form.

Query 4: Obtain a list of all checking accounts where, during any month of the last year, the number of transactions has been greater than N and the average monthly balance has been less than X.

This query may be answered by reference to checking account statement forms.

Query 5: For a given security, display all trust accounts that are holding the security, showing for each account the number of shares held, the value of the holding and the date of last activity.

This query can be answered by reference to trust holding and trust activity forms.

Query 6: For a given customer, display his address and social security number. Display, for each account held by that customer with the bank, the type and current balance.

This query requires reference to savings, loan and checking account forms and trust holding forms.

Query 7: For a given account number, display the name and address of the primary owner, the current balance and status of the given account, and the account number and current balance of all other accounts of which the same customer is primary owner.

This query requires reference to savings, loan, checking account and trust holding forms.

Query 8: Determine whether there are sufficient funds in any account having a given customer as primary owner different from the account on which a given check was written.

This query requires reference to checking or savings account forms.

THE BANK

LOAN STATEMENT TYPE: *Automobile*

LOAN NO. *23414*

TOTAL INTEREST *$252.00*

BALANCE *$3800.00*

STATEMENT DATE *7/15/75*

Mr. John Doe
25 Main St.
Anywhere

PAID THRU DATE	TRANSACTION DATE	PAYMENT AMOUNT	DISTRIBUTION OF PAYMENTS			BALANCE
			PRINCIPAL	INTEREST	LATE CHARGE	
1/1/75	*1/5/75*	*242.00*	*200.00*	*42.00*		*4800.00*
2/1/75	*2/1/75*	*242.00*	*200.00*	*42.00*		*4600.00*
3/1/75	*3/6/75*	*242.00*	*200.00*	*42.00*		*4400.00*
4/1/75	*4/1/75*	*242.00*	*200.00*	*42.00*		*4200.00*
5/1/75	*4/30/75*	*242.00*	*200.00*	*42.00*		*4000.00*
6/1/75	*6/2/75*	*242.00*	*200.00*	*42.00*		*3800.00*
7/1/75	*PAYMENT IS PAST DUE*					

Figure A-2. Instance of a Loan Statement Form

SAVINGS STATEMENT ACCOUNT *S-1452*

THE BANK

ACT. ESTABLISHED *3/18/62*

PERIOD *1/1/75 to 7/1/75*

STATUS: *Active*

Mr. John Doe
25 Main St.
Anywhere

DATE	WITHDRAWAL	DEPOSIT	INTEREST	BALANCE
1/1/75				*1629.40*
1/16/75			*17.30*	
1/16/75			*17.49*	
1/16/75	*INT to DEC 31,1975*		*17.68*	*1681.87*
3/24/75			*17.86*	*1699.73*
3/24/75	*500.00*			*1199.73*

Figure A-3. Instance of a Savings Statement Form

THE BANK

CHECKING ACCOUNT

STATEMENT DATE 6/15/75

ACCOUNT C-8451

DATE ESTABLISHED: 4/18/63

TYPE: Active

HOLD STATUS: No

Mr. John Doe
25 Main St.
Anywhere

DATE	CREDIT	DEBIT	NUMBER ITEMS 5	BALANCE TO 6/15/75
5/5/75	Balance			647.25
5/22/75		27.50 41.98		577.77
6/11/75	247.00			824.77
6/12/75		418.12		406.56
6/12/75	STOP PAYMENT	39.84		

Figure A-4. Instance of an Account Statement Form

THE BANK

TRUST HOLDING

DATE 6/15/75

ACCOUNT T-6724
Personal

Mr. John Doe
25 Main St.
Anywhere

DATE	NO. SHARES	SECURITY	VALUE
3/27/64	140	GM	6860.00
3/19/67	300	DEC	34800.00
3/18/74	500	GE	25500.00
4/2/75	100	ATT	5000.00

Figure A-5. Instance of a Trust Holding Form

THE BANK

TRUST ACTIVITY

DATE 1/1/75 to 3/31/75

ACCOUNT T-6724

Mr. John Doe
25 Main St.
Anywhere

DATE	DEBIT/ CREDIT	NUMBER SHARES	SECURITY	VALUE
1/8/75	D	300	ITT	14100.00
3/18/75	C	500	GE	25500.00
3/24/75	D	500	VEP	6000.00

Figure A-6. Instance of a Trust Activity Form

ACCOUNT RELATIONSHIP

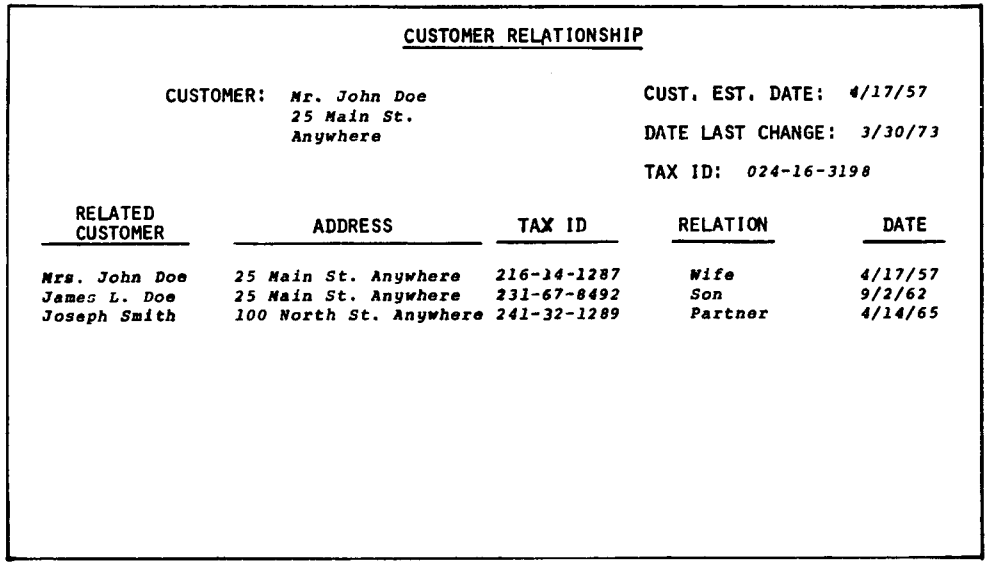
NAME: Mr. John Doe
25 Main St.
Anywhere

TAX ID 024-16-3198

PRINCIPAL OWNER OF:

ACCOUNT	C-8451	ESTABLISHED 3/27/74		RELATION	DATE
SECONDARY OWNER	Mrs. John Doe 25 Main St. Anywhere	216-14-1287		Wife	4/17/57
ACCOUNT	C2384	ESTABLISHED 4/23/64		RELATION	DATE
SECONDARY OWNER	Joseph Smith 100 North St. Anywhere	241-32-1289		Partner	4/14/65

Figure A-7. Instance of an Account Relationship Form



CUSTOMER RELATIONSHIP

CUSTOMER: *Mr. John Doe*
25 Main St.
Anywhere

CUST. EST. DATE: *4/17/57*

DATE LAST CHANGE: *3/30/73*

TAX ID: *024-16-3198*

RELATED CUSTOMER	ADDRESS	TAX ID	RELATION	DATE
Mrs. John Doe	*25 Main St. Anywhere*	*216-14-1287*	*Wife*	*4/17/57*
James L. Doe	*25 Main St. Anywhere*	*231-67-8492*	*Son*	*9/2/62*
Joseph Smith	*100 North St. Anywhere*	*241-32-1289*	*Partner*	*4/14/65*

Figure A-8. Instance of a Customer Relationship Form