Abstract
In the past, clinical trials data were stored in an assortment of separate datasets. Each new clinical program required new datasets and new reporting and analysis programs. In an attempt to reduce the amount of programmer support required for clinical trials, a relational database was developed using Digital Equipment Corporation's Rdb®.

The goal was to provide a stable database that would capture a minimum of 80% of all clinical data collected. All rules governing the management and usage of the data were built into the database instead of the applications programs. We hoped to reduce errors in programming and speed data analysis as a by-product of this effort.

The tools used in the development of this project include V6.06 of the SAS System including SAS/Access®, SQL, Rdb/VMS, and VAX Rally®. This paper presents the results of this effort. The questions that will be answered include: What are the implications of data normalization? Are there real benefits for the users of the data? How well do these tools work together? What have we learned from this project?

Introduction
In this paper I will discuss a particular problem that occurred in our efforts to support clinical trials research. I will give a history of the problem, discuss our proposed solution as well as our implementation plan.

Problem
The nature of Clinical Trials research requires that a variety of information concerning the safety and efficacy of a new investigational drug be collected. This information varies greatly from project to project, and often between different applications of the same drug on the same project. Our first approach to managing this data was to develop datasets specific to each study or project. Each dataset had one record per patient with repeated measures being implemented as multiple variables. The benefit was that only a single dataset had to be opened during data entry and SAS/FSP® screens could be developed that closely mimicked the case report form appearance.

From a data processing standpoint, there were serious implications associated with this method. Any additional instances of the repeated measures required that additional variables be added to the dataset. This in turn required that the data entry screens be modified to reflect these new fields. Then all of the analysis and reporting programs using that dataset had to be modified accordingly.

This problem was compounded every time a new project was begun. None of the previous work was applicable to new studies. As more products came under active investigation, more programming staff were required to support the efforts of the Biostatistics group.

Worse still were the side effects of this approach. Any program that used this data required large data steps to reformat the information into a usable arrangement. Specific assumptions about the data were built into the applications programs and this introduced the potential for inconsistencies in the output.

Proposed Solution
We decided that a different approach had to be found. On reviewing the data it seemed that a database could be designed to capture and store most of the data in a stable format. It was recommended that a new database be implemented using relational technology.
Unfortunately, management was unwilling to commit to such a radical departure from current practice. As an initial step we decided to build an experimental prototype to demonstrate that the proposed tools and methods were a viable alternative.

Experimental Prototype
A prototype application was constructed using DEC's Rdb as the database engine. We choose Rdb because we felt that it was optimized for VMS and as part of Digital's strategic offering we would be insured long term growth and support. Part of our plan was to avoid having large applications programs to maintain where it was not necessary. For this reason we choose VAX/Rally as our development tool to build the data entry interface. This could have been done using FSP, but would have required much more coding.

This prototype was built to fit one specific set of data. The purpose of the experiment was to prove that the tools would all work together and not hamper the efforts of the Biostatisticians. The results were good. The data entry went smoothly. Use of the data stored in Rdb from SAS was nearly transparent.

First Functional Prototype
Based on our initial success, we decided to proceed. Our next step was to develop a database which would have all of the functionality of the proposed application, but only capture one aspect of the data. The intention was to prove the concept of a centralized database without committing too many resources. The application was developed to capture clinical lab data (blood analysis) in a manner that would be flexible enough to work for all conceivable studies. At the same time, the basic control structures essential to the management of the data were put in place.

The most important lesson learned in this phase was the difficulty of introducing change. Some users will be resistant to using something new. What we learned was the importance of understanding everyone's concerns as well as their needs. It is important that they understand why change is important. Even more importantly, the people who will use the system should be included in the planning. They should feel that their input is valued and that this is their system. Any change this large will affect the way people work, for better or worse. It is our responsibility to ensure the former.

First Complete Release
In January of 1991 the first full production release of our new database went on-line. We called it CLINDAT, for Clinical Database. The database captures roughly 80-100% of all clinical data collected, depending on the composition of the study.

The major hurdles encountered in completing this release were training and documentation. Included as part of this release was an extensive documentation manual for the users of the system. This describes the procedure for registering a new study with the database, and describes using each of the data entry screens as well as all of the other features of the application. Since we have a limited number of people to support, we choose one-on-one training, scheduled independently with each user of the system. In our case this worked out very well since each individual has different needs and skills.

In order to more fully test the system we began loading historical data from the past five years. This also gives us a stable foundation upon which to base future analysis in order to make safety and efficacy claims about the products under investigation. It also allows us to predict database performance and compatibility over the next several years.

At this time, CLINDAT has been in production use for a year and a half and we have seen some very real benefits. The size of our applications programs for analysis and reporting have reduced drastically. In one example, a particularly complicated report required a SAS program that was over 1200 lines of program
code. After being rewritten to work in the same fashion but using data from CLINDAT the size was reduced to 900 lines. After being optimized to work with CLINDAT the size was reduced even further to a scant 250 lines. Most of our reporting programs are now in the range of 50 to 100 lines including comments, much less than our former average or 300-500 lines.

Besides just developing shorter programs, we have been able to develop standard programs that may be used for any project that collects similar data. This absolutely cuts down on the programming support required for Clinical Trials, whether it be maintenance of existing programs or developing new reporting and analysis procedures.

In addition to the benefits that we had hoped to achieve at the outset, we discovered several others. Since we had established a repository of control information about all of our clinical studies, we were able to develop several “spin-off” applications.

One example is our clinical project tracking system. This was developed in less than one day and allows us to track all of the information relative to the organization and conduct of a clinical project. Another example is an application that maintains an on-line collection of clinical listings. All of our SAS output that will eventually be assembled into an NDA is available, ready to be incorporated into our desktop publishing system.

Design

The design efforts for this project encompassed two distinct areas: the physical database and the application functionality.

Application Functionality

This step involved determining what types of functions would have to be supported within the application that would control the database. Due to the nature of the system being developed, this was fairly straightforward.

Logical Data Model

From the outset we expected that the design of the database would be the most important facet of the entire project. It was essential that when completed the data structure would be stable and flexible.

When speaking of stability we mean that minimal changes to the database will be required over time. Our experiences in the past with datasets designed specifically for one project were exactly the opposite. As long as the nature of the information remained constant, no changes were required. But if any single aspect of the information varied or if any additional data were collected, everything would have to change. First, new fields would have to be added to support the additional information. Next all of the programs that work with the dataset would have to be modified and re-validated. This includes the data entry screens as well as any programs written to report or analyze the data.

What was desired instead was a more flexible approach. We wanted a solution that would adaptable to our changing needs. In order to accomplish this, we used a process called “Logical data modeling”. This is a methodology that is used to determine the composition of various tables and their attributes. There are many different methods of approaching the logical design of a relational database.

What is important to remember is that, whatever method you choose, it is just a tool to help you think through the problem. Most methods will help you avoid common mistakes. I have found that is is important to step back and look at the complete design frequently. Does what you are working on currently fit with other parts of the design? Will this approach allow you the most flexibility or will certain enhancements be difficult in the future?

The major steps in the logical modeling process involve listing all of the data elements, without any attempt to classify the objects. This is akin to brainstorming. In this list will appear objects that will later be classified as tables, fields, relationships and keys. Through the following
steps in the process these objects are identified and definition of the database is continually refined.

Application Development
Once the database model was completed it was time to begin development of the actual software. The first step was to write the SQL code that would create the physical database including all of the tables, indices, and triggers. The data entry screens and related software were developed using VAX Rally.

If we had any hesitancy at the outset about using RALLY instead of SAS/FSP®, these were quickly laid to rest. Very complex data entry screens were developed in short order. Many of our screens allow the user to enter data into four or more relational tables at the same time, while preserving a simple user interface. Very little programming, in the traditional sense, was involved in developing CLINDAT.

Using CLINDAT
When data is available for a new study one of our statisticians must first enter several pieces of information. This is what controls all of the data entry screens as well as the SAS reporting and analysis programs. Typically this information describes the data that was collected: types of lab tests, normal ranges, limits of extreme changes, test intervals, types of medical history information expected, and more. Additionally, information is entered that specifies how the clinical reports should be formatted. Once complete, data entry can begin.

Some of the features that make data entry a little easier and help assure us of more accurate results are: pick lists for many fields, field validation, consistency checks, transaction controls, and a field level audit trail for all changes and deletions. Many of these features are simple to implement, but time constraints often prevent their inclusion on short term projects. One of the benefits of a standardized database is that the value of these additions increases to the point where they may be incorporated.

Conclusions
What did we learn from this project? First, in trying to change the way we go about the business of managing clinical trails data we had to approach the problem in an open minded fashion. We looked at storing the data in many different layouts, considering the pros and cons associated with each approach. We had to be willing to utilize the most appropriate tools to implement our solution.

There are some differences in working with data from a relational database, too. We were able to eliminate quite a lot of SAS program code that was responsible for preparing the data for use – transposing, sorting, selecting. With relational technology we had more responsibility to write efficient programs, particularly the data extraction steps. We also had the ability to have control over the database performance, which was not possible using datasets.

In total, this project has worked out extremely well for us. We spend less time re-writing reporting programs. Once our users were comfortable using CLINDAT, they were able to work with data from any study without having to learn new any new mechanics. Most importantly, they are able to spend more time being productive with less concern about the peculiarities of the database.

Trademarks
SAS, SAS/Access, SAS/AF, SAS/FSP are registered trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

VAX Rally, Rdb/VMS, VAX and VMS are registered trademarks of Digital Equipment Corporation.