THE REALITY OF SYSTEMS DEVELOPMENT: A CASE STUDY AT THE SLOAN SCHOOL OF MANAGEMENT

by

Leslie Lafer McCafferty

B.A., Economics
Bryn Mawr College
(1984)

Submitted to the M.I.T. Sloan School of Management
in Partial Fulfillment of
the Requirements for the Degree of
Master of Science in Management

at the

Massachusetts Institute of Technology

January 1990

Signature of Author ....[illegible]........
M.I.T. Sloan School of Management
December, 1989

Certified by ....[illegible]........
Stuart E. Madnick
Professor of Management Science
Thesis Supervisor

Accepted by ....
Jeffrey A. Barks
Associate Dean, Master's and Bachelor's Programs

THE REALITY OF SYSTEMS DEVELOPMENT: A CASE STUDY AT THE SLOAN SCHOOL OF MANAGEMENT

by

Leslie Lafer McCafferty

Submitted to the M.I.T. Sloan School of Management
in Partial Fulfillment of
the Requirements for the Degree of
Master of Science in Management

ABSTRACT

The development of computer systems to replace manual processes has become inevitable in today's society given the increasing capabilities and decreasing expense of hardware and software.

The developers of systems designed to manipulate data must be highly cognizant of the ramifications their conceptual systems designs have on the actual implementation of a workable computer system.

This study describes the development of computer systems for use by the Sloan School of Management, with particular focus on the most recent systems development attempts. It discusses the history behind systems development at Sloan and why the current attempts were more limited in scope than prior efforts. It reveals that conceptual systems design is not totally independent of hardware and software considerations.

Thesis Supervisor: Dr. Stuart E. Madnick

Title: Professor of Management Science

CONTENTS

THE REALITY OF SYSTEMS DEVELOPMENT: A CASE STUDY AT THE SLOAN SCHOOL OF MANAGEMENT

CHAPTER ONE
INTRODUCTION

This thesis is a description of a systems development project that was begun during the summer of 1989. The goal of the systems development project was to design, and if feasible given time, program two functioning databases, one for Sloan recruiting information and the other for Sloan alumni information. These databases will provide a threefold benefit to Sloan:

1) They will assist students with the job search process.
2) They will increase the efficiency of Sloan's Career Development Office (CDO) operations.
3) They will provide the Composite Information Systems Tool Kit research project (CIS/TK), being sponsored by the Information Technologies department, with current data to be used for demonstration purposes.

This thesis presents an in depth discussion of the database development projects and a disclosure of the reality of systems development process. First, the history of systems development at Sloan will be outlined via a survey of documentation on this subject (Chapter 2). Next, the fundamental database design approach for the two project databases will be discussed, followed by an in depth discussion of each individual design and implementation effort (Chapters 3 - 5). A discussion of some of the technical issues raised during implementation will follow (Chapter 6). Lastly, based on an analysis of both the history of systems development and the particular experience from the project database development, conclusions regarding the systems development process at Sloan will be drawn (Chapter 7).

CHAPTER TWO
HISTORY OF SYSTEMS DEVELOPMENT AT SLOAN

Computerization of operations at the Sloan School of Management has been discussed by several theses and working papers dating back more than 15 years. These prior works were conducted from both theoretical and practical perspectives and some have resulted in successfully implemented systems of one degree or another. But for the most part, these efforts have not been unified or coordinated and have not resulted in a completed, maintainable, operating Sloan Information System, covering the major administrative functions at Sloan.

In order to understand Sloan systems development history, some background on the Sloan organization is required. While there have been task forces established in the past for directing systems development efforts, there is no centralized Information Systems organization within Sloan. Rather, there are a few key people managing systems usage at Sloan, but no established systems development staff. Nor are there substantial resources earmarked for systems development. Many of the systems development efforts at Sloan have therefore involved students, for both design and programming of the systems. Given an organization without a mandate to develop administrative systems, efforts to create such systems have faced substantial stumbling blocks from the start.

In addition to the organizational constraints, my analysis revealed several other reasons for incomplete systems development at Sloan, the first being a lack of continuity between projects. The earliest project that I could locate was "A Management Information and Control System For a School: Its Application to the Sloan School of Management" by Jean Camille Lavigne and Denis Henri Porges, Thesis June 1971. But there is no way to be certain that this is in fact the first project on the subject. While these older projects are not really relevant to today's needs (due to significant organizational changes and technical advances), the fact that they are difficult to identify and locate is an indication that throughout the years there has probably been a lot of duplication of prior efforts in automating Sloan. These prior works can and should be used as reference points and as a basis for future actions.

The particular paper written by Lavigne and Porges was a study in design approaches. Throughout the process of designing a system for controlling Sloan's use of resources, these students used the then most current frameworks for performing information systems analyses and discussed which approaches worked better than others. From a practical standpoint this project highlighted both the ideal information system for Sloan and the most practical version, given existing constraints. The systems analysis itself concentrated more on the control of information (budgeting and accounting) than on the managerial use of information.

The next paper I identified was actually a follow-up on Lavigne's and Porges' work. "The Implementation of a Management Information and Control System for the Sloan School of Management" is a thesis written by Samuel T. Redwine, Jr. in 1972. The purpose of this thesis was to design, program and implement an information and control system at Sloan - SLINFO. Redwine also outlined some factors about the environment at Sloan which have an influence on systems development (for example, issues concerning use of students to program the system).

The system designed by Redwine was actually implemented in 1972. The thesis does not contain information on the implemented version, nor is there much systems development documentation. Emily Magazzu, the Sloan database administrator for SLINFO, provided the following information:

Originally SLINFO was a batch processing system in a "VS1" environment which required card input, with storage on tape. SLINFO produced enrollment reports, faculty load reports, faculty utilization reports, cost reports and mean salary reports. It changed from card input to interactive input in 1980. This system was operative until Spring of 1987. According to Ms. Maguzzu, by 1987 this system was no longer usable because : 1) the VS1 environment was being eliminated and 2) the programs written for SLINFO could not handle the growing volume of data, and because there were no programmers "available" to Sloan (meaning students with the necessary background), no new programs were written. Hence SLINFO was no longer used after 1987.

In December 1974, William L. Baggeroer and John M. Fox wrote "Student Information System, Sloan School of Management, Preliminary Analysis and Design" as a term project for the management information systems course15.565, Principles of Information Technology II. As with the prior two works, this project tried to address most, if not all, of the administrative functions at Sloan. The focus however, was more on software and hardware selection issues than on the analysis and design of the system itself. According to the authors, this was a preliminary analysis upon which further work should be done.

This paper does mention the SLINFO system produced by Redwine and being used by the Dean's Office. According to Fox and Baggeroer, since that system was working adequately, their project would concentrate on the Master's Program Office (MPO) and the Career Development Office (CDO). They concluded that the MPO did not require automation, but the career placement and alumni tracking functions did.

No information for the period 1974 through 1986 is available. SLINFO was being used, but no comprehensive documentation regarding the system was prepared. Student programmers performed a good amount of the programming for SLINFO, but came and went throughout the lifetime of SLINFO.

The next few papers represent the more current projects aimed at making Sloan administration more efficient. "Systems Requirements Study of the Masters Program Office of the Sloan School of Management", a class project for Course 15.569, Advanced Decision Support Systems, December 1986, was coordinated by Professor Michael E. Treacy. This report is the most current version of a total design for Sloan, contained within one document. The class was split into different student groups, each which was responsible for analyzing a distinct Sloan administrative function:

- Admissions
- Student Services (e.g., grades, courses)
- Placement Office (i.e. CDO)
- Alumni Services
- Integrated System Design (tying all of the functional systems together)

While reading these prior papers it became apparent that there is some confusion about the hierarchy of Sloan administrative functions. In Baggeroer's and Fox's work (1974), the Master's Program Office was responsible for admissions and current student information, and the Career Development Office (then known as the Office of Graduate Student Placement), was responsible for career placement and alumni information. The MPO and the CDO were considered separate organizational entities. The

Treacy Paper implies that the Master's Program Office is the umbrella under which the Admissions, Placement, Student and Alumni services are housed. While this view may or may not have been true in 1986, today the CDO is not a part of the Master's Program Office.

Based on discussions with various individuals involved in the Treacy project, it seems the recommendations of that project were not implemented because of resource problems rather than because of any weakness in the analyses. The project team recommended a hardware, software and manpower package that would, as a rough estimate, cost $185,000. From what I have learned, while there was support in theory for what the Treacy project was trying to accomplish, there was no support in terms of a commitment of resources (computer equipment or programmer/consultant time).

The situation gets even more confusing because in January of 1986 it appears that development of SIS, the Sloan Information System, was begun. The documentation I located on this particular project is fragmented, consisting of whatever was retained by the SIS committee members. The committee was made up of a variety of individuals from Sloan, including faculty and members of Sloan administration. Based on discussions with Ms. Magazzu, it appears that the Treacy project was to be a new starting point for Sloan automation, and the demise of SLINFO. Hence, the Sloan Information System (SIS) project was born. Because there is no good documentation on SIS, it is not clear how much of Treacy's work was used to implement SIS, or if in fact there was a duplication of effort.

In any case, SIS was another attempt to connect all of the Sloan administrative functions via one all encompassing system. The goal of this development was an actual implemented system, and not just a design (as was the Treacy project). Actual programming and implementation did not occur until the summer of 1987. Because of the broad scope of the project several aspects of the system eventually had to be dropped in order for anything to be completed. The resulting system produced enrollment and faculty load reports.

SIS was in operation until quite recently. According to Ms. Maguzzu, two major stumbling blocks for continued use of SIS were the lack of a "front end" for the system (a user interface issue) and the "Ethernet" networking issue. The user interface problem concerned the particular language that was used to create SIS - SQL/DS. It is not easy for a user to learn and there was no support for developing a customized menu-driven user interface. And because the Ethernet network was being used, no prepackaged "front-ends" like DBEDIT were supported. Because Ms. Magazzu is leaving Sloan after completion of the 1989-1990 school year, and because no one has "volunteered" to take over the administration if SIS, it appears that this system will be scrapped.

At the next point in history a more academic and research oriented focus begins to enter into the discussion of a Sloan Information System. "A Composite Information System for the Sloan Placement Office" was written by Laurence Stanley Kooper for his thesis in May 1988. This thesis is the forerunner to my thesis and was done in concert with the Placement Assistant System (PAS) project discussed below. Kooper's thesis is basically a discussion of the PAS project and the rationale behind the decisions being made for that project. It concluded that a "small, committed team who are [sic] familiar with the application can quickly implement an acceptable prototype system". (Kooper, pg. 35) Unfortunately, as will be discussed in further detail below, this conclusion is misleading.

The "Placement Assistant System (PAS)" was a class project submitted by Alan Banks, Laurence Kooper, Andrea Flamburis for course 15.579 (1988) and involved the development of a prototype Placement Assistant System to assist with the CDO's job placement functions. It was hoped that this project would instigate further automation of Sloan career placement and administrative functions. And, as noted above, Laurence Kooper documented key issues in the development process for his thesis. Since PAS was the basis from which I was to develop the recruiting and alumni databases, it will be discussed in more detail than the other prior projects.

From all the evidence it seems that the PAS project was problematic from its inception. It began as an assignment for a course, 15.579. The fact that the project was not initiated from the user area (i.e. the area that would be ultimately responsible for using/maintaining the resulting system - the CDO) was the first and perhaps the most fundamental problem with the project. The motivation for adequately performing a class project is quite different than that of a user area trying to automate its functions. In order to successfully complete the course in one semester, a working prototype had to be developed and completed. Contrast this with the goals of the CDO in terms of automation - to have a functioning system that would meet its information needs and make operations more efficient. This is far from a minor task and would involve significant involvement by CDO staff to ensure that their goals were met. But since this project was not initiated by the CDO, it was not on its agenda at that time to devote the manpower required. As a result, CDO involvement was minimized.

From the PAS project documents and discussions with CDO personnel it appears that the PAS project team did the majority of development by themselves, and did not verity that their assumptions about the tasks they were automating were in line with CDO assumptions. As a result, the final system did not perform in the way the CDO wanted it to, and the system is not being used. While my discussion of this project is not intended to denigrate the work that went into the PAS project, it is intended to highlight a lack of understanding of the CDO perspective on the part of the PAS team. It is evident from the three documents that resulted from PAS (the PAS report, Kooper's thesis and Lisa Tener's paper discussed below) that the intent of the PAS team was to develop what they believed to be a good system, and to hand it over to the CDO for use. However, as my project work testifies (and as is discussed later in this paper) it is impossible to develop an acceptable system in a vacuum, nor should anyone try to do so, because the final product will not be what the end user wants.

In addition to the organizational issues, I believe that the failure of the PAS team system was also due to the team's choice of task. They had chosen amongst the wide range of CDO tasks, the most difficult piece to develop first - the priority card system at Sloan. This task involves second year students bidding for the companies they want to interview with. While it made some sense to select a manually intensive task as the starting point for systems development, the task of computerizing this function was not simple at all. It did not just require a transfer of manually produced reports to computer produced reports. It involved evaluating the bids to determine those students that would interview with each company. It appears that implementing the most complicated function first made it more difficult to test the usefulness/userfriendliness aspects of the system. In other words, had a more simple function been selected first, it would have been easier to attack issues like user screen formats and help functions that would be the users' interface with the system. But by concentrating on the more difficult tasks first, more time had to be spent on programming the application itself and not on how the resulting application would be perceived and evaluated by the users.

Further, a paper written by Lisa Tener as a class project for course15.579 (1988), called "Organizational Issues Involved in the Development of the Placement Assistant System at the Sloan School of Management", also made up part of the PAS project. This paper, by far the most insightful, outlines the organizational dynamics that were going on and also openly identifies areas where improvements could have been made in both the process and the final product. Unfortunately, this paper was difficult to locate, highlighting the difficulty in even attempting to establish some sort of continuity between systems development projects at Sloan. This paper would have been helpful to me when I began designing the alumni database (described later) early in June.

The systems development project to be described in this paper is being supported by a research project described in "Logical Connectivity: Applications, Requirements, and an Architecture" (Working Paper #2061-88, August 1988) by Stuart E. Madnick and Y. Richard Wang. This research project involves the development of a Composite Information Systems Tool Kit (CIS/TK) which will physically and logically link together heterogeneous databases. The paper refers to several Sloan databases that CIS/TK is designed to link, which in fact only exist in a very limited format (e.g., Recruiting, Alumni, and Student databases). The goal of my project was to design fuller versions of the Recruiting and Alumni databases referenced by this paper, that could actually be put in place and used by Sloan, as well as be linked by the Tool Kit. The CIS/TK research project relies on the fact that databases are disparate (not within one computer) and heterogeneous (not using the same programming language or conventions), which is usually the case, and thus while it may not represent the ideal systems development situation for Sloan, this project may eventually result in at least some operable systems for use by Sloan.

The paper "A Microcomputer Network Design for the Sloan School Admissions Office" by Richard R. Barth, is the result of a project for 15.562, in May 1989. The paper outlines the various functions performed by the Master's Program Office and and how they can be better supported by computer applications. While some of the information contained in this report duplicates prior works (because there was no way to identify easily what prior work had been performed), this paper contains a great deal of useful information on MPO activities and will hopefully be used as the basis for future work on automating the MPO.

The "DBMS Software Review", a class project for 15.565 (Spring 1989), Introduction to Information Technology II, represents a review of several database management software packages and how they would serve to meet the needs of Sloan. It was performed on the request of the Sloan Information Systems and Computer Education Committee, in order to identify a relational database management system (RDBMS) that could meet Sloan's needs, given that there is no support for a centralized system as discussed earlier. These general "needs" concern whether a software package supports:

* Complex Procedures/Programming
* Database Maintenance and Growth
* Multiple Computer Platforms
* Educational Uses

The package selected would have to allow for decentralized, user-centered application development and be utilized in an environment without experienced programmers or a database administrator. As a result of this review, the software package PROGRESS was determined to be the package that came closest to meeting the needs and requirements of Sloan.

The conclusion we can reach based on the history of systems development at Sloan is that many of the past projects were trying to address too many functions and too diverse a set of goals. The organizational arrangement at Sloan is not conducive to coordinated efforts because each functional area has its own agenda: the faculty has different goals (primarily research oriented) from the administrative areas (which are more practical). In addition, the responsibility for maintenance has been a stumbling block because in having customized programs developed for use by Sloan, maintenance becomes much more difficult, particularly when programmers come and go. Since there have not been and probably will not be, resources dedicated strictly to the purpose of automating all of Sloan's functional areas, the more reasonable approach from now on appears to be incremental systems development. The systems development project described in this thesis is based on an incremental approach, that is, to develop separate working systems for the various administrative functions, leaving integration and refinement of these systems until a future time. While this may not be the theoretically best approach, it is the most effective practical approach, particularly given Sloan's history. However, despite the limited scope of this project, I faced several obstacles in development, indicating that an incremental approach does not in itself guarantee success.

CHAPTER THREE
DATABASE DESIGN APPROACH

In today's technological environment, there are many choices open to systems developers in terms of the type of system they want to develop - using files versus a database - and within the realm of databases, which type they want to use - network, hierarchical or relational. While this paper is not intended to be a discussion of the merits of each of these types nor a rationale for which is the best, it is worthwhile to briefly discuss each to highlight why the relational approach was used to develop the databases for this project.

A database management system differs from a file based system in that the applications used to process data are separated from the data itself. With a file based system,

> "..each new computer application is typically designed with its own set of data files. Much of the data used in these new files may already be present in existing files for other applications. However, to meet the needs of the new application, the existing files would have to be restructured. This would require that existing programs that use these same files be revised or completely rewritten. For this reason, it is often far simpler (and also less risky) to design new files for each application."(Hoffer & McFadden 1988, pg.9)

As a result of no data sharing, there is much data redundancy as well as a loss of data integrity (because each application makes changes to its own version of the data).

A database system on the other hand,

> "..represents a different concept in information resource management..A database is a shared collection of interrelated data, designed to meet the information needs of multiple users...The data are stored so they are independent of the programs that use them."(Hoffer & McFadden 1988, pg.15)

Centralizing the storage of data in one location allows for a sharing of data that eliminates redundancy and provides for a better degree of data integrity.

Within the database realm there are three types of databases - hierarchical, network and relational. In order to clarify the difference between the three, it is beneficial to define some terminology to describe different elements of data that are stored in a database:

ENTITY An entity is any person, event or object that is of interest to the organization that is automating its operations. For example, one important entity for job placement activities is companies that come recruiting on campus.

ATTRIBUTES Each entity is made up of characteristics called attributes which describe in detail that entity. The attributes of a company would be address, industry, etc.

RELATIONSHIPS Relationships are associations between/among entities which highlight a fundamental interaction between those entities. For example, given that "employee" is an entity, there is a very specific relationship between a company and an employee - an employee works for a company. Relationships may also have attributes.

The difference between the three database types noted above is based on the degree to which relationships can be defined between the key entities within the database. With the hierarchical and network models, relationships must be explicitly defined in the database, usually by specifying an entity's role in a "parent-child" relationship. With the relational model, relationships are implicit, being based on a common attribute between entities and not on any specified role played between entities. With the relational model, redundancy and integrity problems can be eliminated, at least theoretically. In reality, the extent of reduction of these problems is determined by the features of the database management system.

The relational model was selected as the design platform for the Recruiting and Alumni databases. In order to effectively design these databases, a threefold approach was taken with each database:

1) The first step involved designing a "Conceptual Schema" for each database. This schema was based on a thorough analysis of the alumni and recruiting functions, including a review of the input and output documents associated with these functions. Each schema was outlined in an "Entity-Relationship" diagram, to provide a clearer picture of the relationship between entities.

2) After completion of the conceptual design, the next phase in the process was translating the design into a phsycal design, i.e. the physical database. This entails working with a Relational Database Management System (RDBMS) package to program the database schema as close to the conceptual model as is possible, given the capabilities of the RDBMS.

3) The last phase of each project involved programming the applications to manipulate the data in the database and creating an interface for users to access those applications.

This was the general approach taken with each database; a more thorough discussion of each database follows.

CHAPTER FOUR
RECRUITING DATABASE

DATABASE MANAGEMENT SYSTEM SELECTION

For the recruiting piece of the project, a "semi" dual database development scheme was in place, meaning joint conceptual design but independent implementation (described below). The CDO had independently researched their needs and capabilities and selected a relational database management package that suited those needs (RBASE). Amongst all of their career development functions they determined that the Recruiting function needed to be automated first. The CIS/TK team also needed a recruiting database, but a RDBMS was desired that could work in both the mini-computer and personal computer (PC) environments, which RBASE (being PC-based) cannot currently do. It was decided by the CIS/TK research faculty that a separate version of the database would be programmed for CIS/TK's use, and with the CDO's permission, data would periodically be downloaded from the CDO's operating system to the CIS/TK version so the Tool Kit would have access to current data.

The DBMS review work done in 15.565 was used as the primary basis for determining which RDBMS to select for the CIS/TK version of the recruiting database. This project narrowed down the choice of software package to PROGRESS and INFORMIX, the two that came the closest to meeting Sloan's needs highlighted earlier. PROGRESS was ultimately identified by the majority of student groups as the best match because of its more "user-friendly" interface, although the RDBMS was not designed for end-user development. Informed by the analysis of 15.565 as well as by prior indications that INFORMIX was difficult to use (PAS and Larry Kooper's thesis), I selected PROGRESS. PROGRESS appeared to be more user friendly than INFORMIX and therefore I believed it would be more widely accepted by the Sloan community for future database work, if the databases I designed were successfully implemented.

However, during the course of database development, other agendas entered into the decision of a RDBMS, namely the desire to use the same RDBMS in MIS courses at Sloan. This meant that the RDBMS selected had to support SQL (structured query language) which is a standardized approach for querying information in a database. PROGRESS only had SQL available in the PC version at that time, and not in the minicomputer version which would be used by classes and CIS/TK. Therefore the decision was made by the CIS/TK faculty to switch to INFORMIX even though a significant amount of programming had been done in PROGRESS. While this resulted in a delay in completion of the programming of the databases, it did give me a chance to experience a different RDBMS environment. Consequently, it strengthened my conclusion that databases cannot be designed totally independently from the RDBMS that will be used, because the particular RDBMS has a major impact on the final implementation of the conceptual design. While the programming of the recruiting and alumni databases has continued with INFORMIX, it is still unclear what the ultimate software package to be used for future Sloan Information System development will be, particularly since the selection of the software was made independently of the ultimate end users.

GENERAL APPROACH

The approach to the Recruiting Database development consisted of joint conceptual design of the database with a CIS/TK team member (me) and CDO team members (Linda Stantial, Emily Barrett and Darcy Hunter). Each team, using the conceptual design would then program the database in their own RDBMS. In short, as noted earlier, this approach involved concurrent conceptual design but independent implementation. While the development of two independent versions of the Recruiting database was not the most efficient use of available resources, it did serve to meet the goals of the independent groups, namely the CDO and the CIS/TK research project. As highlighted earlier, the differing goals of the administration and faculty can limit the efficiency and effectiveness of systems development at Sloan.

Once the conceptual design was completed, it had to be implemented in the RDBMS. Because of the independent implementation, the CIS/TK version had to be programmed almost in a vacuum, with the specifics of the user-interface, applications, input forms, etc., based on my knowledge of CDO functions and my own judgement. A lot of information was gathered during the design phase but it is not until a design is actually implemented that specific issues such as the format of input forms or the exact functions an application should perform are raised. The CDO is very busy during the summer preparing for the fall/spring recruiting and therefore couldn't be expected to address issues raised by implementation of a database version they weren't planning on using. To basically double their amount of involvement (to cover not only their own version but also the CIS/TK version) was not feasible. Therefore, I had to make several assumptions that otherwise would not have been required if only one software package had been used.

While the CDO did the majority of the programming of their own database in RBASE, I was able to provide some support due to my experience. By assisting the CDO in the development of their version, I was able to obtain a good idea of what they wanted their applications to do/look like generally, and I used this information in the programming of CIS/TK's version. While I was programming my version however, the CDO was making some minor changes in the database design to suit their needs. These changes, for example adding an additional attribute field to a database entity, had ramifications for my version of the database. Because I was not aware of the changes until after I had programmed some applications, I had to reprogram every file that was affected by the change. The moral here is there are so many unanticipated issues that are raised when a database is actually implemented, that one must be flexible enough to make the necessary changes to the database. However, in the case of this project, independent implementation meant that there was a delay in communicating changes which could have been avoided had the development of CIS/TK's version occurred sequentially rather than concurrently.

The original database design was limited to the recruiting piece to ensure that some working system would be finished, rather than trying to design an entire system for all CDO functions, and becoming overwhelmed. Up to this point the CDO staff had had only limited experience with database design and thus wanted to keep things manageable. The PAS project team documentation was used as a starting point for reviewing CDO functions, to take advantage of the research on CDO functions that had already been done and to limit the amount of time that the CDO had to spend on designing the new system, given their past time constraints.

The database design that resulted from this project was very similar to the design that came out of the original PAS project. But because the rationale behind the PAS design entities, relationships and attributes had not been completely documented, the new design required more thought and decision making (probably duplicating what had gone on before) than should have been necessary. For example, in the original PAS design, it was not documented why a unique position number was necessary to keep track of positions companies were recruiting for. And therefore, during preliminary design of the new database, there did not seem to be a need for a unique position number. However, during successive design phases, the usefulness of a unique position number became apparent (and is discussed in more detail later). Had the original rationale for a unique position number been documented, perhaps much time could have been saved in the conceptual design phase of this project. Thus one conclusion this project has highlighted forcefully is the need to document systems development. While it is common wisdom that systems should be documented, it rarely happens because it is given low priority and few incentives. Usually the proponents of documentation are more concerned with large systems development where there is a lot of programmer turnover; and hence documentation is necessary to transfer information. But the same is also true for smaller systems, particularly since smaller systems do not have some of the automatic documentation features that larger systems development environments now have (e.g., automatic documentation of programming changes).

THE RECRUITING FUNCTION ITSELF

The recruiting function at Sloan occupies a significant amount of the CDO staff time. By June and July of each summer, companies recruiting for a variety of positions are already calling the CDO to make reservations for presentations and interviews. The CDO takes reservations beginning in the middle of the summer for presentations to occur in the fall and actual recruitment to occur during January of the following year. The CDO must schedule the interviews and presentations with the companies, reserve rooms, coordinate any catering that is requested, etc. Unfortunately because most of Sloan's administration is currently not computerized, much of what the CDO does is still manual. For example, reserving rooms for interviews or presentations must be done "manually" in the sense that the CDO cannot go on-line to check what rooms are available when, and place a reservation for those rooms.

When the reservations for recruiting and presentations have been finalized, the CDO prepares instruction sheets regarding recruiting procedures and reports outlining the presentations and recruiting that will occur. These reports are circulated to students. Once the students have been given a chance to determine those companies they are interested in interviewing with, the CDO coordinates sign up efforts for interviews by:

- Gathering cover letters and resumes for submission to company recruiting contacts
- Documenting which students have been selected for interviews
- Managing the sign up process for individual interview schedules

First and second year recruiting efforts are handled a little differently, but to the extent that these processes would be computerized during this phase of the project, the functions are essentially the same. The priority card system was not within the scope of this project because of the project's incremental approach. While the recruiting process also includes recordkeeping of actual offers made and acceptances by students, these functions were not covered by this first phase of development because of the time frame. It was more important for the CDO to have a working database as of the start of the school year with the functions relating to presentations and interviews. The functions relating to the tabulation of recruiting results (which does not begin until March/April of the Spring semester) could be postponed until later in the school year.

There are four main components of recruiting , outlined below, which encompass the majority of information that will have to be maintained in the Recruiting Database.

CDO KEY RECRUITING ENTITIES:

Company	Companies that come to recruit on campus for various positions.
Contacts	Individuals at companies that may be important for both the CDO and students to keep in contact with. Currently the contact information is kept in a word processing document (with limited sorting capabilities) and in roladexes by year of acquaintance.
Position	Each company has various positions that they recruit for. Each position is considered a unique entity (e.g., even though many companies recruit for consultants, a McKinsey consultant position is considered as a separate entity from a Bain consultant).
Schedule	Students are interviewed for positions by signing up for interview "schedules". Each position may have more than one schedule. Each schedule is made up of the names of students selected/signed up to be interviewed and their respective interview time slot.

CDO RECRUITING INPUT DOCUMENTS:

The three primary documents used by the CDO to coordinate company recruiting activities are as follows (copies of which can be found in Appendix A):

The Recruiting Reservation Form

This form is completed each time a company calls to make a reservation for the positions they plan to interview for. It is used as the basis for documenting the positions being interviewed for, the dates the company intends to interview, who the recruiting coordinator at the company is (the individual responsible for coordinating the company's recruiting efforts) and the number of interview schedules to be held for each position.

The Presentation Reservation Form

Many companies that interview at Sloan arrange a presentation during the Fall term to familiarize the students with the company and its goals. So, for those companies planning to give a presentation, a presentation reservation is usually made at the same time the recruiting reservation is made. This reservation form contains information on the date of the presentation, the food and/or audio/visual requirements of the presentation, who the presentation coordinator is, and the names of any speakers.

The Job Description Form

This form, completed by the recruiting coordinator of each company, contains information about each position a company is recruiting for. It is sent to each company recruiting on campus and returned to the CDO by a prespecified date, so that students have more information about the jobs they will be interviewing for. This form also documents what industry the company is in and categorizes the job function of the position into predesignated functional areas (e.g., accounting, consulting, finance).

CDO RECRUITING OUTPUT DOCUMENTS:

The following recruiting reports are produced periodically by the CDO, to be used by students and the CDO as needed. The databases that are designed as a result of this project must be able to produce these reports.

Listing of company presentations, both by company name and by date: These listings are used by students to plan what company presentations they want to attend.

Listing of companies recruiting, by date, both for permanent and summer positions: These documents are used by students to determine who is coming on campus to interview and whether or not they need to write a cover letter to the company contact.

Listing of companies recruiting, by company name, both for permanent and summer positions: These listings indicate to students the total number of interview schedules a company has reserved time for.

Listing of Positions Offered by recruiting companies: The title of this report is a bit misleading. It is an alphabetical list of companies and the titles of the positions they are recruiting for at Sloan, not a list of actual positions offered to Sloan students based on interviews held.

Listing of companies, by industry: This is a summary document which lists companies according to the primary industry they are in.

Listing of companies, by job function: Similar to the report noted above, this report summarizes companies recruiting on campus by the functions of the jobs they are recruiting for (consulting, economics, etc.).

Listing of company contacts and listing of recruiting contacts: Both of these lists contain address and phone information of company contacts, the second listing being limited to the contacts responsible for coordinating recruiting efforts.

Interview Schedule: This document is prepared for each interviewer and contains the names of students he/she will be interviewing and the times for each interview.

BENEFITS OF DATABASE DEVELOPMENT

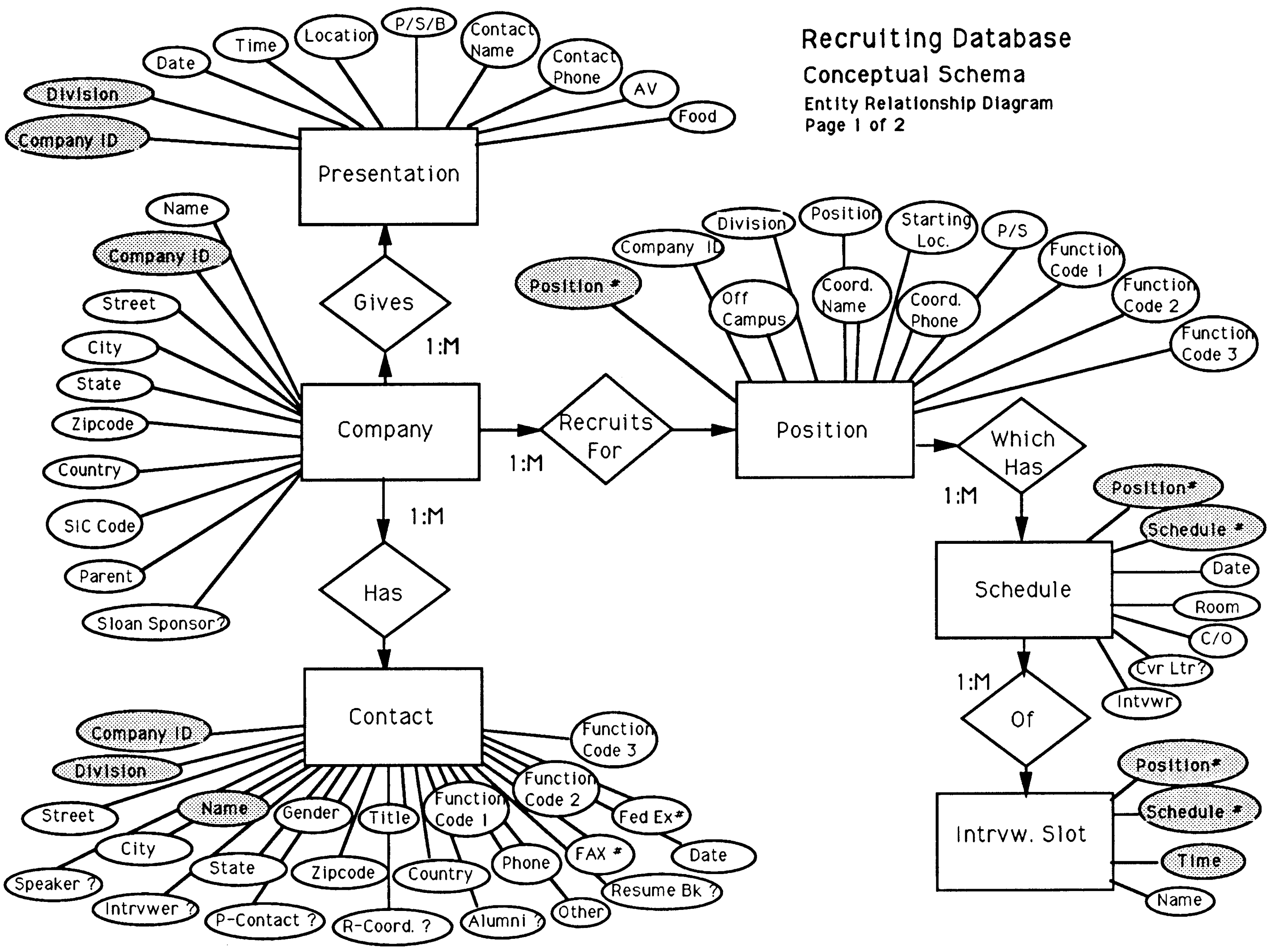
How will a Recruiting database help the CDO? At this point the database is going to help the CDO in terms of the ease of producing the myriad of reports they are now handling via word processing tools. For example, consider the list of companies recruiting at Sloan, sorted by company name. Currently, this information is typed into a word processing document. If a company is added to the recruiting schedule, the company information can be inserted into the list fairly easily, because of the word processing software capabilities. But once the information is entered into the wordprocessing application by name, the entire list has to be retyped if a sort by date is desired (with the CDO staff keeping track of the chronological order for data entry purposes). Computerization of the company information via a relational database should at least make data entry less redundant and production of output much easier. As the CDO becomes more familiar with their software package, they can attempt to computerize their more difficult functions (like the priority card system).

RECRUITING DATABASE ENTITY-RELATIONSHIP DIAGRAM

Recruiting Database

Conceptual Schema

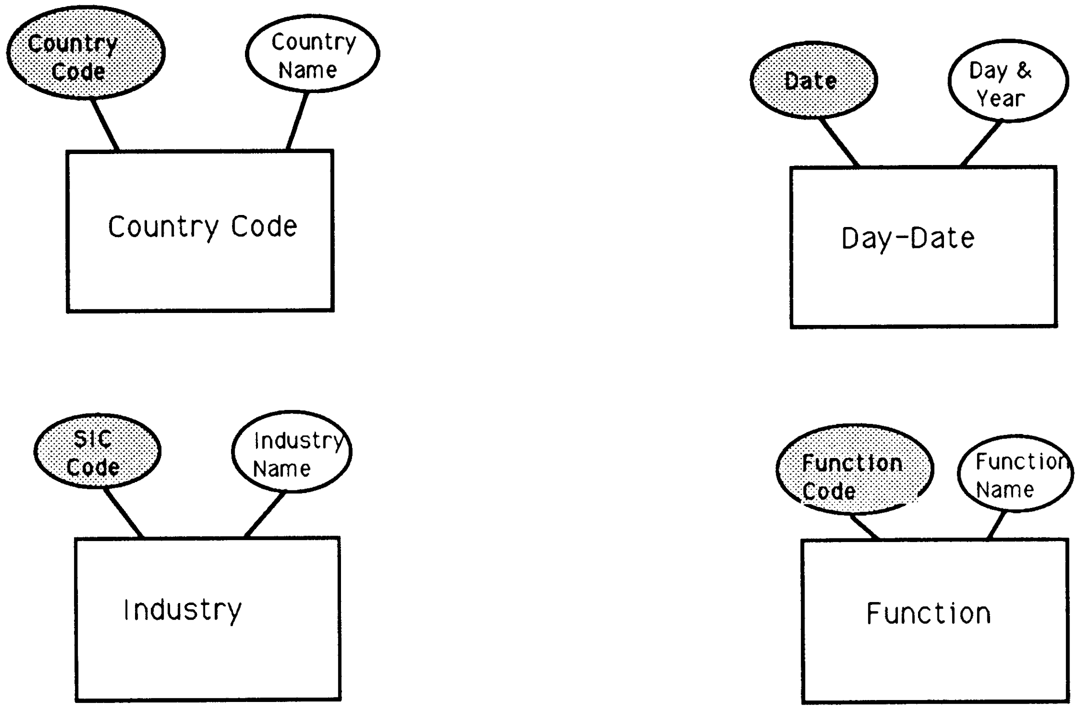
Entity Relationship Diagram
Page 1 of 2



Recruiting Database

Conceptual Schema

Entity Relationship Diagram
Page 2 of 2



DISCUSSION OF CONCEPTUAL DESIGN

The above diagram follows standard entity-relationship diagramming conventions:

- a square/rectangle indicating an entity
- a diamond indicating a relationship
- a circle/oval indicating an attribute

The diagram depicts the conceptual design for the Recruiting Database. When the conceptual schema noted above is actually programmed in a database, the terminology changes. An entity is the equivalent of a database table or file and an attribute is the equivalent of a field or column within a table/file. The attributes that have been shaded in the diagram have been chosen as primary "indexes", an attribute or a combination of attributes that, within each table, is unique. The index serves two purposes: 1) By being specified as unique within a table in the database it will ensure that no duplicate entries can be entered into that table; and 2) For each index, the RDBMS will use that index to sort the records upon retrieval of data from a query. "Concatenated" indexes, i.e. indexes made up of a combination of attributes, are used when there is no single attribute that can be considered unique within a table. For example, it is only the combination of position number and schedule number that is unique in the schedule table.

For each index that is defined, the RDBMS must store an abbreviated version of the indexed records in a new file. Each RDBMS also allows the user to define non-unique indexes, for sorting purposes. There is a tradeoff between defining an attribute/attributes as an index to make sorting more efficient, and having to store the resulting index file. Therefore, indexes should be reserved for the most frequent ways the user needs data sorted (e.g. alphabetically by company date, by date, etc.). The last items of interest in the diagram are the relationship specifications - e.g., 1:M. These specifications indicate the nature of the relationship between two entities. For example, there is a 1:M (one to many) relationship between company and position. One company may be recruiting for many positions. Relationships between entities within the database are established via the use of common attributes. For example, the relationship between a company and a presentation is established by the common use of the company ID.

The following is a discussion of each of the major entities pictured in the Entity-Relationship diagram and how the attributes were selected for inclusion in the database.

Company

For the purposes of the Recruiting database that unit of the company which is coming to interview on campus and in some cases put on a presentation is of interest (i.e. the interviewing entity). This may be a whole company or may be simply a division within a company. It is at this level that we believed recruiting information should be accessed. Thus we needed to cope with the issue of multiple levels of companies. The case of Prudential-Bache and Prudential Investment Corporation is an example where the parent entity is Prudential, and these two companies are subsidiaries. Originally we felt that these two companies should be related back to Prudential within the database. In contrast we have a company like Goldman Sachs (which itself is the "parent"), for which both the Investment Banking division and the Sales and Trading division interview on campus. If two different divisions at Prudential-Bache were interviewing on campus, what should be done? Should each division be linked back to both Prudential-Bache and Prudential?

Through a series of discussions it was finally decided that the information relating directly to recruiting positions and presentations would have to be identified at the divisional level. All divisions would be linked back to the more general company level. So, for example, the company table would contain Goldman Sachs. The presentation table would contain Goldman Sachs, Sales & Trading. As far as subsidiaries like Prudential-Bache were concerned, originally we had included a"parent" field in the company table. This was eventually removed however, because

1) many companies do not have a parent and this would be a waste of space, and
2) the CDO does not have a critical need for this information to be in the database.

We decided to leave it up to the students to determine whether or not the company of interest is a subsidiary of another (e.g. Prudential-Bache is a subsidiary of Prudential), most usually during the phase when they gather information about a company for interviewing purposes.

Up until this point in time, the CDO had been using predefined "industry codes" to categorize the industries in which companies functionally operate. These codes are not comparable to Standard Industrial Classification (SIC) codes, yet are the general standard being used by the career planning services of other MBA programs. After some discussion on this matter, the CDO decided that they were not constrained by what other MBA programs were doing and therefore chose to use SIC codes instead. To make it easier for companies to use the job description forms however, the CDO decided to leave the actual code numbers off the forms (see Appendix A), and let companies check off their industry by industry name.

It was determined that new companies would generally be added to the database from the recruiting reservation form, because these companies first make themselves known to the CDO when they make a reservation. However, industry information, as noted above, is received from the job description form and not the recruiting reservation form. In order to improve the efficiency of entering company information, the CDO agreed to add SIC code to the recruiting reservation form and would ask the recruiting coordinator, at the time he/she is making the reservation, to provide SIC code information. The SIC code information on the job description form would still be used by students when performing their company research but would no longer be the primary source of industry information for the CDO's Recruiting database. Thus, we were able to eliminate the job description form for entry of any primary company information. Once the job description form finally comes in (which is usually much later than the original reservation for the recruiting schedule) only the job function information is entered into the position table of the database.

The company table is where the company ID is established. Company ID's in this table must be unique. The reason for a company ID is mainly for ease of input and efficiency in sorting. The CDO indicated that for companies recruiting on campus, the company names are themselves unique. However, company information has to also be in the position, presentation and contact tables, because it provides the link between each of these entities. We believed an ID code would make data entry easier and would reduce the chance for inconsistency in the spelling of a company name. It was decided that an ID would be used that was an abbreviated version of the company name. The use of numeric codes was discussed (within which each number could represent the various company levels - parent/company/division) but this would have been more confusing than helpful for the CDO.

Rather than getting into numeric codes, the alpha code serves the purpose of reducing manual input while uniquely identifying companies.

The original design for the Company table included address information so that the main address information for a company would be available. However, during actual programming of the database it became apparent that the CDO and students would never have a need for a company address independent of a company contact. Therefore, the address attributes were eliminated from the company table.

The SIC code is in the company table, rather than with any other entity, in order to keep SIC code maintenance to a minimum. Parent information, as mentioned earlier, was eliminated from the Company table.

While many discussions were held on the company issue, I did not learn all the details until after the design was completed (e.g., that only companies that plan to recruit on campus give presentations). Although these details did not have an impact on the final design of the recruiting database, they potentially could have. The key point is that an independent systems developer, even if she understands the functions she is automating, is still bound to have made some incorrect assumptions about the functions being automated.

Contact

Information on contacts had some problems similar to company information. Should contact information be stored with each presentation record and/or each position record that a contact is responsible for coordinating or more generally by company/organization? There are a large number of contacts that are not responsible for coordinating a presentation or an interview and these contacts would properly belong in a separate table.

Ideally, and as was finally implemented, it made sense to have one table containing all the information on contacts. This would eliminate redundancy of information, if for example, one contact was to be coordinator for two presentations and two recruiting efforts. The contact information (e.g., address) would only have to be entered once into the database and would be linked to the presentation and position information. However, herein lies the problem: the best method to link the contact with the presentation and position information would be to use unique numbers (e.g. ID numbers) in the position and presentation tables, to distinctly identify each position and presentation. These unique numbers would be recorded in the contact's record to indicate for each contact the events she/he is responsible for. As conceptually designed, the position table already had a unique position number, but if a particular contact was responsible for multiple positions, including each position number in that contact's record in the contact table would be cumbersome. Presentation did not have a unique number nor was it deemed worthwhile to give it one. (There is more discussion on unique identifiers shortly.) Using contact name by itself in the position and presentation tables would not be a good idea because names are not unique.

The resulting design was the most functional: having the contact name and company ID in the presentation and position tables and using the combination of these fields to link the information in these tables to the contact's full record in the contact table. The CDO accepted the fact that there may be a limited amount of redundancy in this method (having contact name in each table), recognizing the tradeoff between good design and matters of practicality. In actual fact having the contact's name within the position and presentation tables makes identifying the coordinator of a position or presentation much easier, because doing a structured query language (SQL) query on one table is easier than doing one when trying to join two tables. Thus, the actual disparity of the tradeoff between design and practice was minimized. If the software package is capable, the redundancy of information in terms of storage would not necessarily mean a duplication of input. If the input applications could be programmed adequately, the contact information would only have to be typed in once but would in fact be entered into every table where it would be appropriate. So, for example, when the CDO would input a recruiting reservation, the contact information would be entered in both the contact table and the position table. There is a more detailed description of the input applications later in this paper.

One option open to the Recruiting Database was to use unique identifiers (either numbers or letters) for each table in the database, and not just for the Company, Position, Function, Country and SIC tables. That would mean also having a unique identifier for the Presentation, Contact, Schedule and Interview Slot tables. Unique ID's to identify records in each table may be, from a theoretical design standpoint, the best way to ensure there is no mistaking one piece of data from another and the easiest way to link records in the tables. However, in reality the CDO function does not involve information that would effectively use unique identifying numbers (like a purchase order or inventory system would), nor would the database grow to such an extent to require unique identifiers for these tables. Position Number and Company ID were reasonable comprises for the CDO.

While most of the information in the contact table pertains to the address and position of the contact, there are some additional descriptive fields that were added to make the contact record more meaningful. A simple "y" for yes or "n" for no in the fields Speaker?, Interviewer?, Presentation Contact?, Recruiting Coordinator? and Alumni? indicate what relationship this contact has with Sloan. In addition, the field for Resume Book? indicates whether or not this contact has purchased a resume book that the CDO compiles of student resumes. The Date information is used to determine how old the contact record is.

Presentation

As mentioned earlier, contact information was put into this table in order to link the presentation record with the full contact record in the contact table. Telephone number was also added because of the convenience of having this information with the respective presentation, again in terms of querying the database. However, during the actual programming of the database it was determined that RBASE was not very flexible in terms of its data entry abilities. Therefore, telephone information was removed from the presentation table in the CDO's version of the recruiting database to avoid the inefficiency of having to manually enter the telephone number in both this and the contact table. In the CIS/TK version of the database the telephone information was retained in this table because I was able to program the data entry facilities to enter the telephone information into both the contact and presentation tables from one data entry input.

The audio/visual and food fields will allow the CDO to enter information on what types of equipment the presenting company requires and what catering service will be providing food.

Position

It was at the level of position that information such as position type (permanent/summer) and job function would be kept. So even while there may be more than one interview schedule for a given position, for all schedules for that position, the identifying information such as title and job function would apply in most cases. For those instances when there would be a mix of positions within one interview schedule, it was decided by the CDO that a unique position number would be created and given a title such as "Multiple" to indicate that the position/schedules were mixed.

As discussed earlier, it was determined that the best way to keep track of the positions that companies were recruiting for was to use a unique position number (as the earlier PAS team had done). The CDO treats each position as a separate entity, yet there was no easy way to make each position unique (titles are often the same; we would have to use a concatenated key made up of company ID, division, and title to uniquely identify each record). So a unique position number was selected instead.

The interesting thing about this field is that we wanted the number to be generated by the RDBMS. What one would think would be an easy task, was in fact fairly complicated depending on which RDBMS was selected. For PROGRESS, a new table had to be created that kept track of elements that are system controlled. Then within the application programmed to add new positions to the database, special code was needed to keep track of the position number count. I had to define a nextposition# variable and a lastposition# variable, in addition to the already existing position#. The lastposition# would keep track of the last generated position number. When a new position was added to the database, the lastposition# would be incremented by one and this value became the position# in the position table.

In contrast, INFORMIX had a data type called serial which automatically generated unique index numbers based on the starting number the user designates. RBASE required a calculation similar to that of PROGRESS. For copies of the PROGRESS and RBASE programs, see Appendix B.

Some discussion took place on capturing information on starting location, starting salary, etc., as originally designed by the PAS team. For two reasons this information was omitted:

1) This information is not used by the CDO (only information on actual offers made is used)
2) This information is subject to negotiation.

For example, a company from New York may be interviewing for positions all over the world. Where an individual may end up depends on her skills, interests and availability. The same is true of salary. Many factors enter into the determination of salary, particularly experience. Therefore, any starting location or starting salary information entered in the position record at this point would be spurious.

Schedule

As mentioned earlier, each position may have several interview schedules. For each schedule, the company determines whether the interview will be closed or open (C/O). An open schedule implies that if the student is interested in interviewing, all she has to do is sign up for the interview. If the interview schedule is closed, it means that the company selects whom they want to interview. They notify the CDO which keeps a log of those who have been selected by the company. Students look at this log and if they have been selected, proceed to sign up for an interview. (For second year students the process is complicated by the priority card system.) Companies also indicate whether or not they require a cover letter along with the student's resume. As it turned out, during the course of using the database, the CDO encountered some companies that wanted schedules made up of half closed/half open interviews. To address these instances the CDO decided they would just put a "b" for both in the C/O field, rather than the usual "c" or "o".

There is usually more than one schedule of interviews for a given position, because each schedule has a maximum of 15 interview slots. Each schedule is handled by an interviewer from the company. The schedule number in this table must be input by the CDO (rather than be self generated) because for each position number the schedule numbers range from one to usually no more than five (and as mentioned earlier, having totally unique schedule numbers was not deemed worthwhile). Therefore, while schedule numbers by themselves are not unique, the combination of position number and schedule number together is unique.

Interview Slot

Interview Slot is the actual list of student names and the times they will be interviewed for a given position and schedule number. Once a company has identified those students it wants to interview for closed schedules, or students decide on their own for open ones, these students sign up for a particular schedule and a time slot to be interviewed. Time slots are usually in 30, 45 and 60 minute increments, depending on the company. The CDO then types this sign-up sheet in a "neat" form for the interviewer. The Interview Slot table is intended to replace the manual typing up of the sign up sheet. Via the combination of the position, schedule and interview slot table information, the "neat" version of the day's interview schedule can theoretically be produced by the database. All the CDO should have to do is print out from the database a blank sign up sheet and after it is filled out, enter the student names and times into the interview slot table and print out the completed form. While this is what was intended, it is not apparent whether this application will work as planned because generation of the form has not yet been attempted. If it involves using standardized report writers, as RBASE does, it may be difficult to get the forms to print out exactly as desired. There is a more detailed discussion on the topic of application development later.

Support Tables

The Industry, Function and Country tables are supporting tables which not only reduce the amount of space the database takes up in the computer but makes maintenance of the database easier. For example, the Industry table keeps track of Standard Industrial Classification (SIC) Codes and the industry names to which they apply. If this table were not in the database, then the company table would have to have a field for industry name rather than SIC code. Not only would this require a large amount of space (about 45 characters would have to be reserved for the industry name field), but it would make maintenance difficult because if the name of the industry changed (for example from Securities to Investment Banking), the name would have to be changed for each company record in the company table whose industry was Securities. With the Industry table, the name only has to be changed once.

Originally it was thought that to get the date printed out in the way the CDO desires (e.g. Thursday, January 12th), it would require a table (hence the Day-Date table in the diagram) that would relate a date to the day of the week that date falls on. However, after becoming familiar with the software packages, it appeared that these RDBMS's often have preprogrammed functions that do date conversion. Therefore, this table became unnecessary and was eliminated from the database.

PHYSICAL DESIGN

The final physical design as implemented in each RDBMS (PROGRESS, INFORMIX and RBASE) differed from the conceptual schema in a variety of ways, some of which have already been described above. Some additional changes worth noting are as follows:

1) *Rather than having contact name as one field, it made sense to split it out into a separate field for both first and last name, so that sorting and querying could be done on the last name.* However, this means that in order to print mailing labels, a program will have to be written to remove extraneous blanks between the first and last name (because the field sizes are 12 and 15 characters long, respectively, but not all names use up all that space). The issue of extraneous space is also true for the address information, when one would try to combine city, state and zip code. Some RDBMS packages, like RBASE, have mailing label applications already written which minimize the problem.

2) *One line for street information in the contact table would not be enough in all cases to capture a contact's address.* This issue came to light only after the CDO attempted to input contact address information into the database. Both the CDO and I had reviewed the address information at the conceptual level and determined that one line would be enough. However, as evident in my thesis, every possible situation can never be identified in advance of using a system. Other examples of this include the change in the size of the zipcode field that had to be made. Five more spaces were added to the original field size of five, because it turns out that many contacts have an extra four digits in their zipcode, separated from the main zipcode by a dash. In addition, many contacts can only be reached at a certain phone extension.

Thus an additional field for phone extension had to be added in each table where there was a contact phone number.

3) *The issue of data type for each field in the database was very dependent on the RDBMS selected.* For example, in both the interview slot and presentation tables there is a field indicating the time of the interview or presentation. Neither PROGRESS nor INFORMIX have a "time" data type. Originally in PROGRESS I had programmed the time field type to be an integer (because time is in numbers not letters). But in order to get the numbers to print out with a colon between the hour and the minutes, I had to go through a very complicated process of cutting and pasting together the data in the time field. I tried to change the type to character, and then specify the format to include the colon, but I could not get that to work either. When I switched to INFORMIX I specified time as a character, and in INFORMIX it is easier to specify a format with the colon. RBASE on the other hand, the least powerful of the databases, has a data type called "time" which allows the time to be displayed in a variety of ways (with colon, with seconds, with am/pm, etc.) The same problem is true for phone number information - it has to be declared as a character instead of as integers in order to be displayed in an appropriate way. In addition, RBASE allows one to enter the date in one format (e.g. 09/10/89) and display it in another format (Friday, September 10th). To do this in PROGRESS and INFORMIX required programming, cutting and pasting to convert dates rather than a simple format statement.

4) *The RDBMS's each provided a different method for creating a database.* Located in Appendix C are copies of the data dictionaries for each of the RDBMS's, which are created when the database schema is programmed. These dictionaries contained within each database, keep track of the table and field characteristics (i.e. the database structure). Each RDBMS has a different way for programming the database structure.

PROGRESS has the most extensive method of creating the database structure. By this I mean that in addition to specifying the data type and length for each field within a table, PROGRESS also allows one to:

- specify whether or not there is to be a default value for the field
- define a help message to appear at the bottom of the screen each time a particular field is accessed in a data entry form
- define indexes that are made up of more than one field
- define criteria that must be met before a record can be deleted from a table
- define criteria for a field that must be validated before any data can be inserted into that field

While the other two RDBMS's also support these features, they must be programmed separately from when the table and fields are created. While this is not a problem, it does result in certain inefficiencies. For example, in INFORMIX, for each data entry/query form that is programmed, the help messages for each field must be defined within each form (rather than having them predefined once in the data dictionary for use by all forms). It is somewhat inconvenient to have to program validation criteria or concatenated indexes at a different point in time than when the database table and field structure are being programmed, because it makes it more difficult to keep track of what has and has not been programmed.

One concern with the PROGRESS table creation procedure is that trying to define delete and validation criteria at the time the database is created (and not after the programmer and user are comfortable with the structure based on their practice with it) has its own drawbacks. I had a lot of problems programming data entry forms for the PROGRESS database tables because I had written the validation criteria incorrectly. It took a lot of trial and error debugging to figure out that there was nothing wrong with the data entry forms I had programmed, but rather the field validation criteria in the actual structure of the database were incorrect. These errors were not detected until I actually tried to enter data into the database via the data entry forms, because the data dictionary is not designed to check validation criteria for proper syntax or coding.

APPLICATION DEVELOPMENT

In order for users to enter information into the database, or to query the database to locate information, I designed "forms" that would appear on the terminal screen to be used for these purposes. The forms that could be designed were dependent on the capabilities of the underlying RDBMS. As will be discussed in more detail later, the tools provided by the RDBMS's both helped and hindered form design and usage. Once database forms were designed for each table and for each key input document prepared by the CDO (namely the recruiting and presentation reservation input forms), programs were written to allow the user to use the forms for data entry and querying. Then these forms and programs, as well as any programs written to generate reports, would be joined together in a menu system that would be the general interface a user would have with the database.

Form and Report Design

Because creating forms and programs to access information contained within one database table was fairly straightforward, the forms and programs designed for the individual recruiting database tables will not be discussed in detail here. Instead, the design and programming of the Recruiting Reservation form and Presentation Reservation form will be discussed, because each involved a much more complicated programming process.

Recruiting Reservation:

As described earlier, and as is evident from the form itself in Appendix A, this input form contains information that will eventually need to be entered into several tables in the database, namely the company, contact, position and schedule tables. The basic idea behind creating a database form that replicates the information on this reservation form, is to allow the CDO to enter data into the database in the format in which they initially record the reservation from a phone conversation. Eventually, the CDO should be able to bypass having to document the information manually, by entering it directly into the database; but until they become confident with the integrity of their database, they are going to continue to manually document the reservations before performing data entry.

Because the recruiting reservation form is the primary CDO form, it should be the starting point for recording recruiting information in the database. Appendix D has copies of both the PROGRESS and INFORMIX version of the programs designed to perform the recruiting reservation data entry. Each is commented to describe the process that must be gone through in order to enter a reservation into the database. While the programming of these applications was fairly extensive, they will result in a significant reduction of manual input on the part of the CDO because they handle some complicated tasks. For example, the PROGRESS and INFORMIX programs insert contact name and phone information into both the contact and position tables, from one input of the data into the database form by the user. Before inserting any data into the database, the programs check the company and contact tables to determine whether or not the particular company or contact have already been recorded in the database. For the INFORMIX version I made the program automatically insert the current date into the contact date field, indicating the date the contact became known to Sloan. The program also automatically places a "y" in the recruiting coordinator? field of the contact table, thus eliminating the need for it to be manually entered by the CDO.

In order to create the recruiting reservation form in RBASE in contrast, we had to use RBASE's form developer, which produces forms to be used by RBASE's preprogrammed data entry and data modification applications. (As mentioned earlier, I provided some support to the CDO during their programming of the database.) While RBASE allows the creation of forms that will affect multiple tables, it has a very specific procedure for doing so, to allow for the forms to be used by RBASE's preprogrammed applications. This procedure, however, would not allow us to design the form exactly as I was able to in PROGRESS and INFORMIX. It would not allow us to check against both the company and contact tables to see if these entities already existed for the reservation in question. This "limitation" in RBASE form creation comes from the fact that only one "master lookup" can be defined. A "master lookup" is RBASE terminology for an expression which checks for preexisting records. For example, for each company ID that is entered into the form, the master lookup checks whether or not there is already a company with this ID, and if there is, it displays the values of the other fields for this company record into the database form fields on the screen. The user is then able to make any changes to the record. If a record does not already exist, the user will be allowed to enter in the additional information for this company ID. As the master lookup can only be done once within each form, we could only check the company or contact tables for preexisting records, but not both.

While this limitation is probably intended to keep the forms simple enough to be used by the RBASE applications, it highlights the tradeoff between userfriendliness and power. It was simpler for the CDO to avoid programming applications to manipulate their forms by using RBASE's preprogrammed applications instead. But it took a significant amount of trial and error on our part before we would accept the fact that we could not develop the recruiting reservation form the way we wanted to. The result was that the section of the form for inputting contact information for the contact table (e.g. address and phone information) was removed from the recruiting reservation form (and only the fields from the position table for contact name were retained). For a printout of the "compromised" RBASE Recruiting Reservation input form, see Appendix D.

In order to enter the address information for the contact captured manually when the reservation is made, another form was created to be used solely for contact information. The result is a more cumbersome data entry process because the CDO user has to use two separate forms in order to enter the information from one recruiting reservation. The CDO can go back and forth easily between the two forms by creating a special application, but they are still required to enter some duplicate information (namely, company ID and contact name). While RBASE is the more userfriendly RDBMS, its lack of power in terms of capabilities made certain tasks in the database development project more difficult to accomplish. Thus, it appears that some of the more powerful packages, like PROGRESS and INFORMIX, for all their complexity may in fact be worth learning because of the ease of developing specific applications with a minimum of difficulty once the learning curve has been ridden. While increased familiarity with RBASE would make development easier, it would not mitigate the fact that application design has to be compromised to fit its capabilities.

When programming a form, more questions emerge than were originally anticipated during initial design. For example, having to check for preexisting records before inserting data to avoid duplication. Some degree of this is avoided by specifying that certain attributes within a table have to be unique (e.g. company ID in the company table), so if a user tries to add a company that was already in the database, he would get an error message. But as was experienced with RBASE, defining attributes as unique or mandatory caused problems when the CDO tried to use their data entry forms. The problems relate to the master lookup described earlier: when the form was accessed by the user, and, if a field was declared mandatory or unique, the master lookup would be performed before the user entered any data and an error message would be received that no record was found. Rather than change the data entry forms (and thus the way the CDO handles information) while still using the RDBMS rules regarding uniqueness and mandatory input, we decided to keep the forms in the format that was most in line with CDO processing and removed the rules from the CDO database for the time being. With greater familiarity with RBASE, the CDO will most likely learn about the "correct" way for using the rules, but the manuals do not really address this issue and thus experience will again have to be gained through trial and error.

This issue reveals a fundamental question that must be addressed by systems developers and end users: Should automation direct how the task should be handled or should the task direct how the automation should be handled? In the case of the Recruiting database, the answer to this question is a combination of both - as applications are developed we are trying to minimize the changes in procedures that have to occur because of the limitations of the software packages.

An additional problem that came up during the development of the recruiting reservation programs supports the notion that systems cannot be designed without user involvement. After I had completed the design of the recruiting reservation input form I discovered that the form was not exactly what I had assumed. There is a line for division on the form, below which the recruiting coordinator's address information is listed. I assumed that the division was the recruiting coordinator's division (usually human resources) when in fact it was the division recruiting for a position.

This discrepancy makes a difference because when programming an input screen, one must designate the table into which each piece of input information will be inserted. Had this matter not been identified during development, "incorrect" information would have been put into the database, because I originally programmed the recruiting reservation input form to insert the division information into the contact table. Thus, the division of the recruiting area would actually be going into the contact's record rather than in the appropriate record, position. As noted earlier, the assumptions a systems developer makes about the task being automated can be critical to the functionality of the resulting system.

Presentation Reservation:

Similar problems were encountered with RBASE when designing the Presentation Reservation input forms and applications. With the PROGRESS and INFORMIX database versions the user could enter the names of multiple presentation speakers into the contact table. In RBASE this was not possible because a "master lookup" for a preexisting contact record could not be programmed when the primary table used in the Presentation Reservation form was the presentation table. Appendix E has the PROGRESS and INFORMIX presentation reservation application programs and a copy of the compromised RBASE presentation reservation input form.

Reports

The CDO is currently in the process of designing their recruiting reports. Because the fall semester will begin shortly, the CDO has been trying to use RBASE to design their reports and has hit some snags that they are slowly but surely overcoming. For example, creating reports that list company information alphabetically was initially difficult. Before the CDO had become familiar with the RBASE way of doing things, the reports were coming out alphabetically by company ID, not by company name (the ID's of which are not always alphabetically equivalent).

Upon further trial and error the CDO determined that, when creating reports from information in more than one table, the first table that is defined in the report will be the table that is used to sort the information. It was not until we investigated the problem in more detail that we learned about RBASE "views", that solved the sorting problem. These views allow selection of only those fields from the existing database tables that are needed. By using a view, we were able to choose company name from the company table, as well as presentation information from the presentation table, to produce an alphabetical listing of companies (by company name) giving presentations.

Because the CDO has just begun creating their reports, they will face similar trial and error processes in design until they become more familiar with RBASE's capabilities.

RECRUITING DATABASE CURRENT STATUS

The CDO has completed the majority of input screens they need to accomplish their most pressing tasks. They are now in the process of completing their report design. They still have a long way to go before they can develop a menu system for their database, but their system is essentially up and running. They have already produced the schedules of company presentations for students and are working on the recruiting schedules.

They have added a "library" function to the database, indicating whether or not the CDO has a file of information on the company. Originally they were planning on creating a separate database for the library function. But after we discussed issues of duplication (namely company and industry information) and storage space (they were already reaching the limits on their computer), it was determined that this function would be added to the Recruiting database for the time being, by adding additional attributes to the company table. In addition, they have added a new entity called "JOBPOSTS", containing information on "correspondence" opportunities, i.e. job postings that are sent by mail to the CDO.

As just mentioned, because of the hardware the CDO is using, they are already experiencing space problems. In fact, the space problems have been causing some "bugs" in their applications. For example, after having successfully run their presentation schedule reports several times, they noticed that the dates being printed on the reports were incorrect. When they finally deleted unnecessary files from their computer, the problem went away. They are in the process of requisitioning new hardware which will hopefully relieve their space problem.

The CIS/TK version written in INFORMIX is in an intermediate stage. The majority of input forms and data entry programs to manipulate those forms have been written. However, with changes being made to the CDO's version, changes in INFORMIX version data dictionary, input forms and data entry programs are necessary. The final step in the process will be to write the report programs and then link the applications with a menu system. Even without the menu system completed, we have begun the process of dumping a copy of the CDO's data to a floppy and then loading the data into the INFORMIX version. Once the loading is complete, this database will be accessed by the Composite Information

Systems Tool Kit. I plan to write a program in RBASE that when run, will automatically perform the dumping function. This dumping will be performed periodically to ensure that CIS/TK has current data.

CHAPTER FIVE
SLOAN ALUMNI DATABASE

ALUMNI HISTORY

Because the documentation available on this subject is sparse, much of the history of Alumni recordkeeping functions outlined below is based on my discussions with Ms. Magazzu, database administrator for many of Sloan's database projects. According to Ms. Magazzu, the following phases describe the Alumni "system" at M.I.T.:

1950's	File of Index Cards
1960's - 1970's	Punched Cards
1976 - 1977	Inforex System (which created card images)
1978 - present	ADABAS DBMS

The ADABAS system is called the Alumni/Development/Donor Systems (ADDS) and is maintained by the centralized M.I.T. Alumni Office.

Sloan had its own alumni records on punched cards in 1972 which were computerized in 1973 to produce the first Sloan Alumni directory. Prior to this, directories had to be manually typed. As with SLINFO described earlier, programs were written for the VS1 environment to take card input until 1982 when data entry was made interactive and the information was stored on tape. This system was expanded to produce lists, labels and various reports. In 1979, Ms. Maguzzu initiated some major additions to allow sorting of information by SIC code and company name.

In 1982 the Alumni Association information was computerized (it had begun in 1978), and Sloan alumni functions were merged with all of M.I.T.. From that point in time it took almost two years to get reports produced by SIC code and company name, for use by Sloan. Most recently Ms. Maguzzu has had direct access to ADDS and therefore produces many of the reports on Sloan alumni that the Sloan administration needs.

GENERAL APPROACH

Ms. Maguzzu is leaving Sloan in Spring of 1990. This project is intended to design a relational database that will allow the Sloan administration, students and CIS/TK direct access to Sloan alumni information. If the design and programming of an alumni database for exclusive use by Sloan can be completed before Ms. Maguzzu leaves, then she can set up a routine whereby Sloan periodically receives a dump of information on Sloan alumni from ADDS that can be fed into this Sloan alumni database and used as needed by Sloan. If the programming cannot be completed, then in the future all requests for Sloan alumni information will have to be circulated to the M.I.T. Alumni Office.

The major impetus today for developing a Sloan Alumni database is coming from CIS/TK's need for a full blown alumni system in place of its very limited existing one. This is not to say that the MPO and CDO do not want a system, but they are not the one's supporting the development at this time. Therefore, the RDBMS selected for this database will be the same as for the CIS/TK Recruiting database - INFORMIX.

In an effort to identify the potential uses for Alumni information by Sloan, I held discussions with Ms. Maguzzu, who was a significant help in the design of the database outlined below, as well as reviewed the major documents she produces for Sloan, namely the various Alumni directories, sorted on different dimensions. Based on these discussions and documents I was able to design an alumni database, although until there is end-user review of the design, its validity remains limited. It is obvious that students and the administrative staffs can benefit from having access to alumni information, but it must be in a format which they can use to meet their needs.

Because ADDS contains a significant amount of sensitive alumni donation information, Ms. Maguzzu explained up front that the Sloan database could not contain all information that is currently in ADDS. In addition, because this project is not being initiated by the administrative office at Sloan (which may or may not at some point want add/update/delete capabilities and thus a direct link to ADDS), the applications that will be developed for the Sloan database will be "view" only. In other words, the users will be able to search for alumni information in varying ways, and print out that information, but they will not be allowed to make any additions or changes to the data.

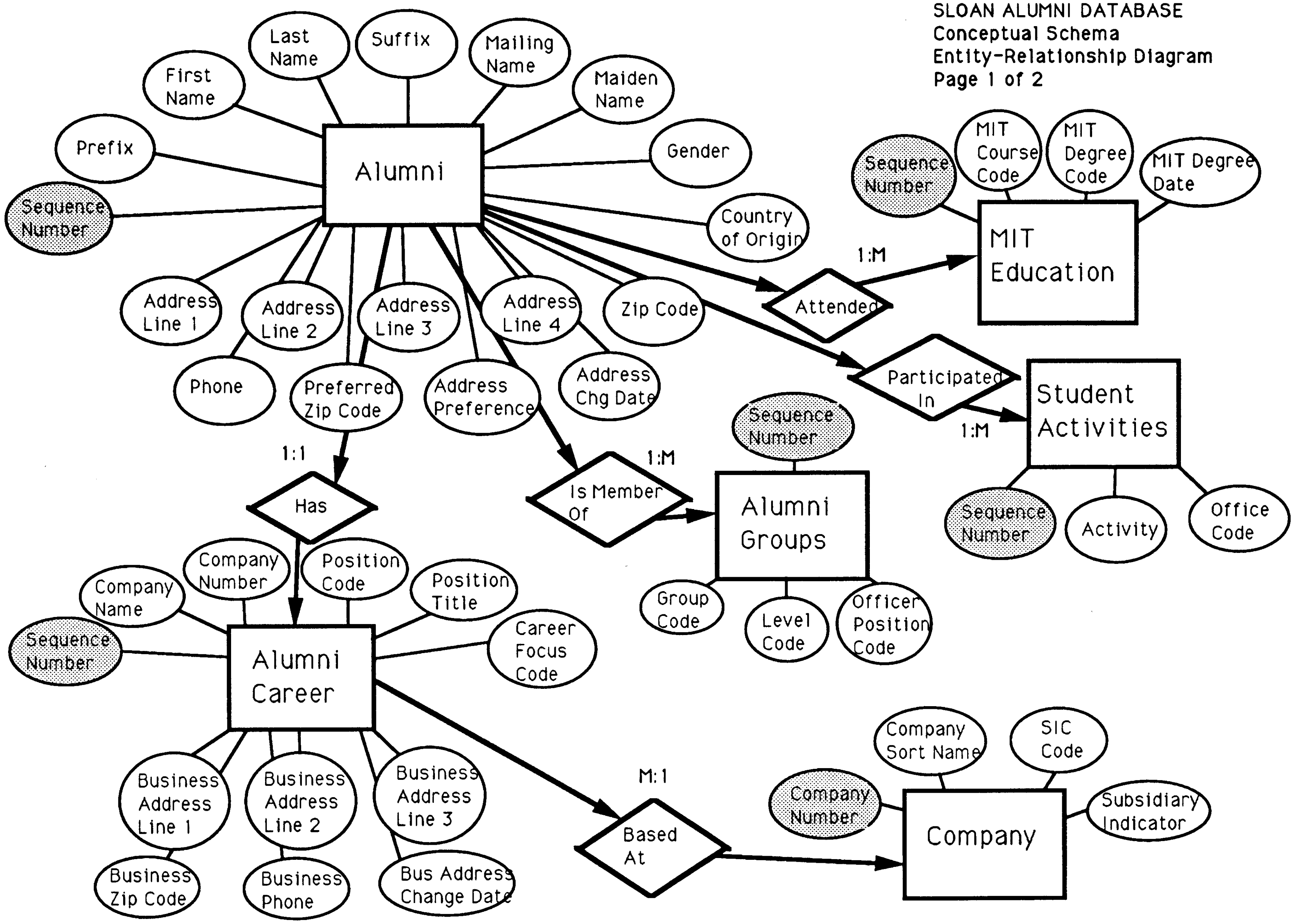
BACKGROUND REVIEW

The conceptual design process began with my review of the ADDS files and tables to determine what alumni information already exists and to decide what we would want and would be allowed to have access to. Then I designed relational tables to handle the information. There were a total of 57 ADDS files and approximately 138 "tables" with supporting information pertaining to alumni information. Appendix F is a copy of a document I created describing these files and tables and whether or not I thought they were suitable for the Sloan Alumni database. On the first review I was able to narrow down the relevant information to 5 files and 11 tables that could be utilized by the Sloan Alumni database. Then based on further discussion with Ms. Maguzzu, this was narrowed down even further to select information within each of those files and tables.

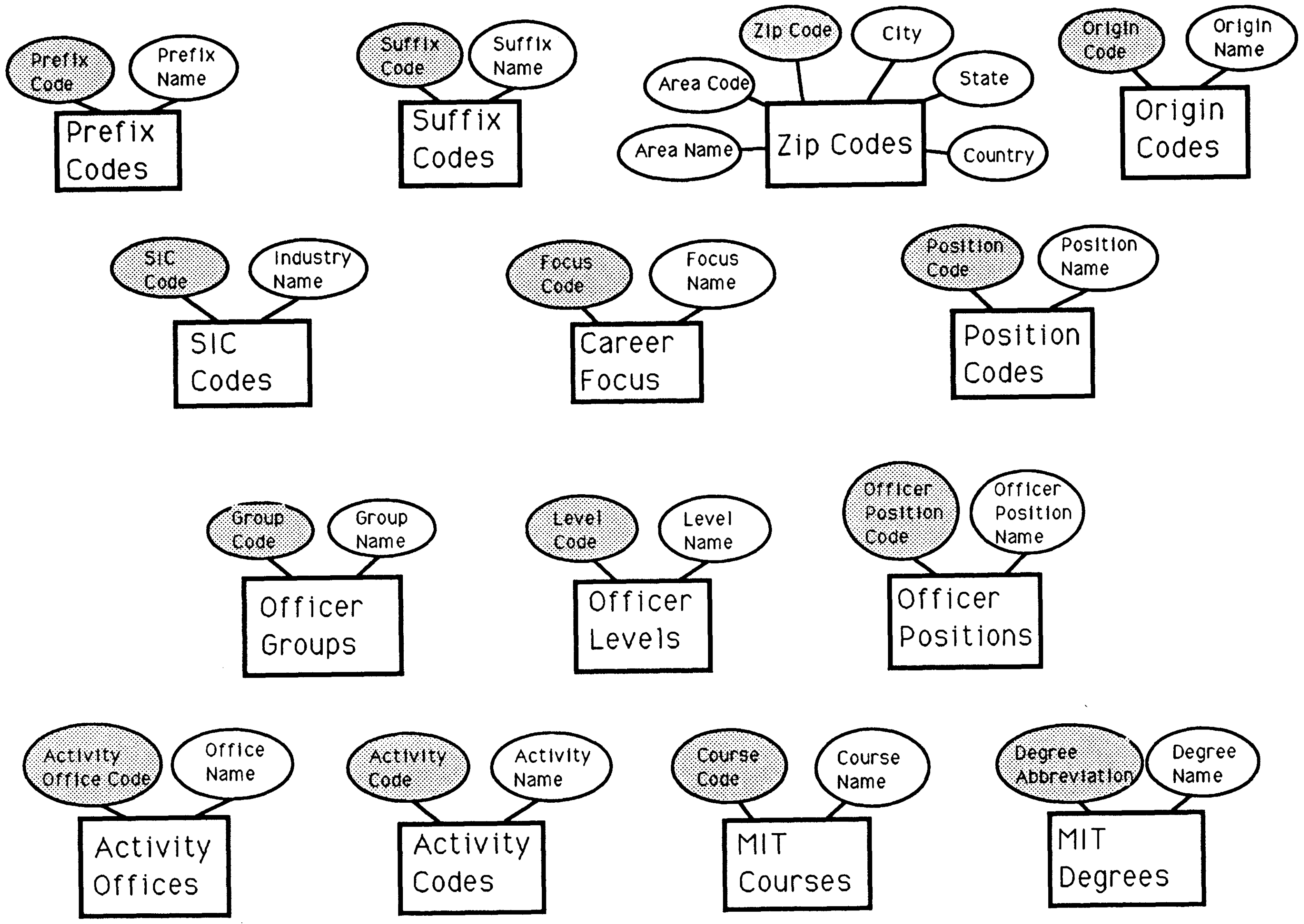
The majority of the information that is relevant to the Sloan Alumni database comes from the ADDS master file, which contains most of the background information on each alumni. This file alone consists of approximately 230 fields of information on each alumni. Thus the majority of my review involved selecting information from this file that I thought would be of interest to Sloan students and administration. To provide a flavor of the decision making process that I went through, appendix G outlines my thought process in selecting fields from this file. The resulting conceptual design is outlined below in an entity-relationship diagram.

SLOAN ALUMNI DATABASE ENTITY-RELATIONSHIP DIAGRAM

SLOAN ALUMNI DATABASE
Conceptual Schema
Entity-Relationship Diagram
Page 1 of 2



SLOAN ALUMNI DATABASE
Conceptual Schema
Entity-Relationship Diagram
Page 2 of 2



The above entity-relationship diagram uses the same diagramming conventions noted earlier, only in this case the shaded attributes are not all unique indexes. The sequence number, as described below in more detail, identifies each individual alumnus and is used to link the information between many of the tables. So while sequence number will be unique in the alumni table, it may not be unique in the M.I.T. Education table, because one alumnus may have graduated with several M.I.T. degrees. Therefore, in the M.I.T. Education table, this alumnus' sequence number will appear several times. So while the sequence number in this table is an index for sorting, it is not unique.

CONCEPTUAL DESIGN

Alumni

The alumni table consists of primary alumni information, including background information and home address. Using the knowledge I had gained from completing the Recruiting Database design, I selected from the ADDS Master File the separate fields for first and last name, rather than the field called "legal name" in the master file, which has both first and last name together in one field. In the alumni table, a code for both prefix and suffix will be used; these codes are in an expanded version in the Prefix Code and Suffix Code tables. For example, if the prefix were "Atty", the expanded version would be Attorney. These codes are more extensive than just Mr. and Mrs. and therefore provide additional descriptive information about the alumni.

The history and current usage of ADDS is very interesting in itself, and there are many fields in the ADDS database that appear redundant and unnecessary to an outside observer. But given the piecemeal way the database was developed over the years, due to changes in the M.I.T. environment and the environment at large, these inefficiencies are somewhat understandable. Some of what may appear as a duplication may not be because of the special purpose a "duplicate" field serves for one or another M.I.T. entity. For example, in the alumni entity pictured above, there is a field for mailing name. This field contains the name of the alumni in a format suitable for appearing on mailing labels or envelopes. While it is basically a concatenation of the Prefix, First Name, Last Name and Suffix fields, it removes any extraneous blank spaces that would occur if these fields were printed individually in a row.

By copying this field from ADDS and inserting the information into the CIS/TK version, the need to write a program to concatenate the four fields into one and delete the blanks each time mailing labels are needed is eliminated (as may have to be done in the Recruiting Database). Because the volume of mailings at M.I.T. is enormous, it is evident that having this field already in the database makes processing the labels that much easier for the users. Access to this field in the Sloan Alumni database will probably be restricted to key individuals in the administration because the easier it is to print out mailing lists and labels, the greater the chance that this ability will be misused. A mailing list is a valued commodity in these days of mass, direct marketing.

The attributes gender and country of origin were selected because they provide insight into the alumnus' background and may help students and the administration in identifying subgroups within the entire Sloan alumni in which special interests are held. The Country of Origin table contains the country name spelt out. Maiden name will help in those instances when trying to locate an alumnae who has married since attending Sloan.

The sequence number is a unique ten digit number assigned to each alumnus as he/she graduates from M.I.T.. It consists of the year of the alumnus' first degree (the first four digits), with the last six digits being assigned somewhat sequentially. Ms. Maguzzu and I discussed adding a hyphen to the number after the first four digits, so that the number will be more meaningful to the user and more easy to remember and detect errors. She can add the hyphen in the edit mask she creates, which is the program she has to write in order to download the alumni information for the Sloan Alumni database from ADDS. This added hyphen should not be a problem at this point because for the time being, the Sloan Alumni database will not have any add or update capabilities. If eventually adds/updates are done at Sloan, this may be an issue because we would want things consistent with the information in ADDS.

The reason for four separate lines of address information is a direct result of the format of this information in ADDS. In the ADDS master file, the home address information for the alumni is actually in one field, which is four lines long. Each line contains different information, usually: 1) street; 2) city, state, zip code; and 3) and 4) country information. To handle this information in the Sloan Alumni database, I split it into a field for each address line. And even though line 2 of the address contains the zip code, because it is not segregated from other information, this zip code cannot be used for sorting purposes. So an additional zip code field had to be included in the Alumni table. The address change date identifies the last time the home address was changed and thus gives an indication of whether or not the address information is still accurate.

Phone number is the alumnus' home telephone number, as based on my design only home address information is contained in the Alumni table. By separating home address and business address into two different tables, querying for one or the other type of address will be made easier.

The remaining fields relate to the alumni's address preference, in terms of being contacted by individuals affiliated with M.I.T. The preference mainly has to do with solicitations by M.I.T. fundraisers, but can be extended to the realm of student contact. The information to be stored in the Preference field will be either an "h" for home or a "b" for business. And the preferred zip code will be the zip code for the preferred address, either home or business. By having the preferred zip code in a distinct field, what might at first appear as a duplication, will in fact more easily allow users to query information based on the preferred address of the alumni (by sorting on the preferred zip code). Even though the preferred zip code field has enough information to identify the preferred address, it is still worthwhile to have the address preference attribute because it is more descriptive of which address is preferred (the zip code is just a number). So if a user wanted to merely scan the database, based on the "h" or "b" in this field the user will know which address is preferred without having to compare zip codes. And also, there are some alumni records that only have one address, and the address preference attribute will identify which address definitely exists in the database for an alumni.

Because the volume of Sloan alumni records is fairly large, the following records may have to be restricted from the download we receive:

- Records of deceased Sloan alumni
- Records of retired Sloan alumni
- Records of undergraduate Sloan alumni

The final decision on restricting these subgroups from the download should best be made after discussions are held with the ultimate end users, who will primarily be students and Sloan administration.

Alumni Career

This table contains information relating to the alumnus' career and company. It contains the business address in three lines, because the master file field for business address is three lines long. Again, business zip code while in the address line information, is also separated out as its own field so that it can be used for sorting. The remaining address fields are similar to those for the home address, described above.

Both the name of the company the alumnus works for, as well as the company number, are in this table. The company number is a link to the Company table. The reason for having the company name in the Career table, and not just in the Company table, is because of the varying levels of company information. When a new company is added to ADDS, it is recorded in the ADDS "companies" file and it is assigned a company number. But sometimes an alumnus may spell the name of the company differently than it is recorded in the "companies" file. For example, if the alumnus gives I.B.M. as the name of the company he/she works for, chances are very good that this company is the same as International Business Machines, to which a company number has already been assigned. So, the company name is recorded in the alumnus' master file record exactly as he provided it, but the company number that is used in that record will be that for International Business Machines. So, in the CIS/TK database, the career table will have both the company name as the alumnus provided it and the standardized company number.

The same holds for the alumnus' position information. Both a position title, as well as a position code referring to information in another table, are needed in the Career table. The title appears in this table exactly as it is provided by the alumnus. The position code, which is expanded in the Position Code table, relates to a position name which standardizes the alumnus' title for querying purposes. For example, if the alumnus' title was President and CEO, letters should be sent to her with that title. However, if one wanted to get the records of all alumni who are both CEO's and Presidents (in that order), the query that only used position title would not locate any alumni with titles where the words were reversed. Therefore, it is important to have both the verbatim title and a standardized version of the title.

The last attribute in this table is the career focus attribute. The information that will be stored in this field in the database is supported by the Career Focus table. The career focus codes basically categorize the focus of the alumnus' career. For example, while the title of the alumna might be "President", this title is not very informative. If however, it is combined with a career focus code that relates to "Entrepreneur", it is more clear that this alumna is probably the president of his/her own company.

Company

As mentioned above in the discussion of the Career table, the Company table specifies the standardized version of the company name that the alumnus works for. In this table however, the company name is recorded in an order suitable for sorting purposes. So that a company name like Eli Lilly would in fact be stored as Lilly, Eli, because this is the correct way to alphabetically sort this information.

The company number is a ten digit number, with the first seven digits representing the parent company, and the remaining digits indicating that the company is a subsidiary of the parent. There is no additional numbering scheme for special cases like jointly owned companies or subsidiaries of subsidiaries. These cases are fit into the parent/subsidiary numbering scheme. For ease of viewing, as with the alumni sequence number, a hyphen will be added by Ms. Maguzzu after the first seven digits. While the company number itself indicates whether or not the company is a subsidiary, there is also a one character subsidiary indicator which provides an additional means to access subsidiary information. Since it is only one character in length and it already exists in the ADDS master file, I included it in the Sloan Alumni database design in anticipation of future uses.

It is in this table that the standard industrial classification (SIC) code for the company will be recorded, with values supported by the SIC Code table.

M.I.T. Education

As noted earlier, this table will contain information on each M.I.T. degree earned by Sloan alumni. The sequence number will identify the particular alumnus. The M.I.T. Course indicates the code for the course, whether it be a "15" for management or "6" for Electrical Engineering and Computer Science, etc. The M.I.T. degree code highlights the degree received as a result of that course, e.g., M.S. for Master of Science. This table will include information on alumni who had attended courses but did not graduate, because there are unique codes for non-degree candidates. Lastly, the date the degree was received, or the date of last attendance, will be included. The information in this table will be supported by the M.I.T. Course Codes and M.I.T. Degree Abbreviations tables.

While ADDS contains information on degrees held by alumni from other institutions, this information is not reliable (because it is not verified or updated), and therefore it will not be included in the Sloan Alumni database. In addition, even though many alumni have several degrees from M.I.T., we will be limiting our download to the most recent five M.I.T. degrees held by each alumni.

Up until now, information pertaining to the area of concentration while at Sloan has not been recorded anywhere. Ms. Maguzzu is checking out the possibility of having this information retained, and if so, then it would probably be recorded in this table of the Sloan Alumni database.

Student Activities

This table will contain information on the student activities the alumni was involved with while at M.I.T. While currently this information mainly covers sports and religious affiliations, Ms. Maguzzu is trying to have it contain information about Sloan clubs. So, for instance, a user could determine whether or not an alumnus was a member of the Venture Capital Club at Sloan, and if so, whether or not he/she held an office. If Sloan club information is not eventually provided, then deletion of this table from the Sloan Alumni database may be warranted, because the other information is not crucial and pertains mainly to undergraduate affiliations.

The codes in this table are supported by the Activity Code and Activity Office tables.

Alumni Groups

This is the last major table in the Sloan Alumni database and outlines which alumni groups the alumni is active in. The level and group information indicate the alumni interest groups which the alumnus is a member of. For example, it may indicate that the alumnus is a member of M.I.T.'s Enterprise Forum, a group that gives advice to small businesses. This table will also indicate whether or not the alumnus holds an official position in that group. The Officer Group, Officer Level and Officer Position tables support the codes recorded in this table.

Zip Codes

While this table is really a supporting table, it is of interest because it will provide the link between an alumnus' address and the city, state and country in which that address is located. Because the city , state and country of an address is together in one line for both home and business address, it can not be used to sort alumni information. If a user wants to sort by state, and not by zip code, then he/she must use this Zip Code table to get at state information. The area information is particularly interesting. From what I am told, these area codes were nationally established and used by M.I.T. before zip codes were established. Even when zip codes came to be, the area codes were still utilized because many of the administrative areas at Sloan had already set up regional groups based on those area codes and did not want to rearrange those organizations simply to suit the new zip code usage. While the zip codes do provide a lot of information, the area codes can provide information relating to larger regions. Region information would be useful to students looking for alumni information. For example, if they wanted alumni information for the New England region or the European region, rather than having the user determine which zip codes comprise those regions, the pre-established area codes can be used and linked internally to the zip code information.

SLOAN ALUMNI DATABASE CURRENT STATUS

At this point in time, Ms. Maguzzu is programming the edit mask that will provide us with the Sloan alumni information. The remainder of the database development, physical design and application programming, will occur subsequent to the completion of this thesis and thus will not be described herein. Suffice it to say that based on my experience with the Recruiting Database, when actual programming of the database begins, new issues will be raised regarding the functionality of the database design and applications which will have to be addressed on an ongoing basis.

CHAPTER SIX
TECHNICAL ISSUES

This project involved the use of three very different software packages, each with a different level of user friendliness. RBASE is the most user friendly, from the standpoint that it seeks to minimize the amount of development that a user has to do independently from its preprogrammed application programs and its CASE (Computer Aided Software Engineering) tools. (I use CASE in its broadest sense, to include not only tools that help with programming but also tools that eliminate the need for programming, as RBASE predeveloped application programs do.) PROGRESS was developed for a more sophisticated database developer, most likely a programmer, but it has CASE tools and a non-procedural fourth generation language (4GL) which allows the user to describe what she wants to accomplish without having to specify each step to achieve that end. INFORMIX's 4GL on the other hand, has both non-procedural and procedural elements, and thus is the most complicated of the three from a programming standpoint. While INFORMIX has some CASE tools, they are not very extensive.

As was discussed earlier, the recruiting Reservation form involved inserting data into multiple tables. The programming of these forms in PROGRESS and INFORMIX was not that difficult once the methodology for form/application design was understood (although for a person without any programming background it would have been difficult). In RBASE however, there is not a fully fledged programming environment in which to program forms. While one can write some programs to do small tasks, such as generate unique position numbers or convert date types, RBASE is not intended to require a lot of programming and thus much is left to the CASE tools to handle.

One of the valuable CASE features of PROGRESS was its ability to generate program code after I designed a form or report using its form and report generating tools. So I could design a basic image of what I wanted the form or report to look like, and from that image, PROGRESS's FAST TRACK application would produce the actual program code. I could then change the code as I desired to change the appearance or functionality of the form or report. While INFORMIX has a similar feature, it was much more limited in its scope. For a non-programmer as I was at the beginning of this project, this PROGRESS feature was a mixed blessing. I was at first relieved to know that the amount of programming I would have to do myself would be limited. So I spent most of my time using PROGRESS's tools, learning their idiosyncrasies and how to get around their limitations. But this was by no means a small effort, and I had to invest a lot of time in learning how to successfully use the CASE tools. The following is a further discussion of some of the problems that can develop when a database developer relies too heavily on CASE tools, based on my history with PROGRESS's FAST TRACK, RBASE and INFORMIX. Since many of RBASE's "limitations" have been discussed previously, and because INFORMIX's CASE tools are so limited, the following discussion will concentrate heavily on my experience with PROGRESS.

PROGRESS's FAST TRACK allows one to 1) create forms for which it will generate program code that can be used within other programs and 2) using those forms, it will generate a "Query By Form" (QBF) program that allows end users to query for information within a table using the fields in the form. The PROGRESS manual cautions users that making any changes to the underlying database structure (like changing the size of a field or the format of a field), will require recompilation of any program code affected by that change. However, the manuals do not explicitly note that for each form that is affected, recompilation is not enough. One actually has to go back into the FAST TRACK form generator, cut the field in question from the form, reinsert it (from the updated data dictionary) and then have FAST TRACK generate the code. Then FAST TRACK has to be invoked to generate a new QBF for that form. One cannot just recompile the existing code for the form and QBF and assume that during compilation, it will access the updated data dictionary for the new field characteristics.

Another "hidden" fact in PROGRESS was that the usefulness of the help messages I programmed in the data dictionary for each field was negated by the FAST TRACK tools. It was not until I had created several forms using FAST TRACK, and then generated the QBF program code to use the forms, that I found out that the length of the help messages I had programmed in the data dictionary were too long for the message size that can be handled by the FAST TRACK form generator. So, when I ran the QBF and proceeded to enter data into the various fields, the messages that were appearing at the bottom of the screen were getting truncated. The length of the message, or more particularly the fact that the length differed from that in the data dictionary, was not mentioned anywhere in the documentation nor does it logically make any sense. This is actually what I would call "anti-CASE tool" logic because it requires extra wasteful steps to shorten the help messages within the form. If the table creation in PROGRESS goes so far as to let you enter help messages for each field, it would seem reasonable to assume that those help messages could be used with the FAST TRACK form generator without any necessary modification.

To add to this list of foibles for PROGRESS's FAST TRACK is the fact that when creating forms that show the fields listed across the screen, rather than one on top of the other, not all of FAST TRACK's features work. In particular, the feature that allows specification of a title. Even if a title is specified for the form and FAST TRACK generates the code, the title does not print out on the screen when it is used for data entry. In order to have the title print out, one has to physically type the title on the screen within FAST TRACK's form generator, and then have the code generated.

As mentioned earlier in the discussion of the Recruiting database, the accuracy of the data dictionary specifications I input for each field were tested by PROGRESS when I used the CASE tools to design forms. This was one good thing about having a CASE tool, because without it it would have taken me much longer to debug my data entry forms. But even given this benefit, I still had to spend a lot of time on the forms that I may not have had to spend had I programmed them from scratch. For example, consider a field like country code in the contact table. While I did not specify this field as a mandatory field (i.e. it can be left blank by the user when entering a contact record), because I had specified a validation criteria (namely that any country code input into the contact table must be a code that exists in the country table), it automatically became a mandatory field. In other words, if I tried to enter a contact record without inputting a country code in this field, I would get an error message from PROGRESS. To get around this problem I had to specify a default value, in this case U.S., to be filled into this field automatically but which could be changed by the user. While this may not seem to be serious, it was not mentioned anywhere in the manuals, and as discussed earlier, because I had to specify the default value in the data dictionary, I then had to redo each form and QBF affected by this change. Thus for every field for which I had specified validation criteria yet did not want to be mandatory, I had to correct the form and QBF.

The manuals have no references covering how the code generated by FAST TRACK can be modified. As a matter of fact, the code generated for each QBF is made up of subprocedures, some of which are proprietary to PROGRESS and hence cannot be modified. This makes it impossible to modify the existing QBF to do special processing, such as looping in an update program to allow iteration within a specific group of fields. For example, I had to program a separate application for adding Recruiting interview slots to the database in order to allow the user to enter multiple students names and times for a given interview schedule without having to reenter the position and schedule number. Even RBASE's form generator has the capability to use "tiers", which allow for iteration within a specific portion of a form.

Another thing that seemed inefficient to me was that even though the manuals continually stress the benefit of being able to use the code generated by FAST TRACK in other applications you may develop, the code is not in a configuration usable by other programs. In order to make the code usable, the word "FORM" has to be added to the top of the code and a period added to the end of the code. I did not know this until I tried to use the form code in another program and it did not work. Then I had to debug the error. It is not clear why these words are not automatically entered into the code when it is generated.

Another simple issue concerns "auto-return". Both PROGRESS and INFORMIX allow a field to be specified with the auto-return capability, which means after information has been entered, the cursor will automatically go to the next field without having the user have to hit the return key. But the manuals do not indicate that this facility should be used for fields with standard input, for example, where a four digit code is always entered. Fields that involve variable length input, will not work with auto-return unless the data entered fills up the entire field length. For data that does not fill up the entire field length, the user does have to hit the return key. If this feature can only be used for fields with standardized length input, it probably should not be used at all because it will be confusing for the user. If the user has to remember each field that he has to hit the return key for, the data entry process becomes that much more difficult. And as happened to me during testing, if I did not remember that a field had auto-return and I hit the return key anyway, the cursor would actually advance two fields and would confuse data entry.

The above noted technical issues were not anticipated or evaluated during the software selection process. Attempting to identify these types of issues at that stage of the systems development process would probably have been impossible. In outlining these issues here, the desire is to stress just how difficult it is to truly know the strengths and weaknesses of a software package before the implementation stage of a systems development project.

CHAPTER SEVEN
CONCLUSIONS

As a result of the development of the Recruiting and Alumni databases, a general "methodology" for approaching systems development at Sloan "evolved". This methodology is really a series of steps, not necessarily ones that can be performed in a given sequence, but nevertheless actions that can be taken to increase the likelihood of successful system implementation.

Allocate a significant portion of resources to the implementation of a conceptual design, rather than towards the conceptual design phase. At most the conceptual schema for a database can occur independently of the software selection and even the validity of that schema is questionable in light of the software capabilities. Issues as simple as how "time" is recorded in a database or generating unique identification numbers, which are central to the entire concept of a working database one may have designed, can become stumbling blocks if designed independently of the software capabilities. Therefore, time spent perfecting a conceptual design is wasted when in many cases, as my projects reveal, changes must be made to that design no matter how perfect it is due to software ideosyncracies.

Given resource restrictions, selection of software should (within limits) conform to hardware specifications. When it comes down to it I believe in many cases the hardware decision is not a decision at all because so many organizations already have personal computers which they are not going to scrap simply because they want to establish a database. Given the hardware that exists, software selection becomes that much "easier" in the sense that the software available to one particular platform are more limited then if the decision is being made independently of hardware. Of course, if the hardware is inadequate (particularly in terms of memory), then this issue should be addressed in tandem with software selection.

<u>A decision must be made up front over who the implementers will be and who the users will be, and whether or not they are one and the same.</u> This is a crucial decision because it will determine which software feature (userfriendliness or power) will be the overriding factor for selection of an RDBMS. Userfriendliness takes the form of a menu driven system and the existence of CASE tools. Power means the ability to use the package to perform complex tasks. The dichotomy between userfriendliness and power is not clear cut because of the way CASE tools work; trying to work within the confines of a CASE tool (for example RBASE's form developer) can often be more difficult than programming an application would.

Using CASE tools is not as easy as software marketers would have users believe. Because there is a significant amount of underlying processes which the user never sees (which are within the CASE tools themselves), it can at times be very difficult to debug a problem, as was highlighted by my experiences with both PROGRESS and RBASE:

1) If the user is not going to look at the code generated by the CASE tools, because they are using the RDBMS as an end user programming facility, chances are they will take an unnecessary amount of time to figure out the idiosyncratic way to use the CASE tool so that they can get around rather than solve their problem (e.g. recruiting reservation form in RBASE).

2) If they do intend to use the generated code, they will have to deal with some very sophisticated programs (in terms of the coding) and will have to go back and learn how to use the SQL or 4GL so as to be able to debug the generated code (which will cause delays in the completion of the database and its applications). This latter is an example of PROGRESS because while it would generate the code for a form that would access more than one table, it would not generate a QBF if the form involved more than one table. So I had to do the programming for the recruiting reservation form myself, having up until this point relied on the FAST TRACK tools to generate

the code for the data entry/querying applications I wanted.

Perhaps in the best of circumstances the user will have initially started by learning how to program using the SQL or 4GL before using the CASE tools, but this is not very likely because it would defeat the purpose of having a CASE tool to begin with. This issue may in fact be a function of the specific CASE tool in question, and not for CASE tools in general, however, given that during one summer I had the chance to use three different packages to develop the same database, I can honestly say that there are some general concerns that must be recognized when using CASE tools.

Become familiar with the operating system that the database will be supported by early in the development process. The RDBMS's themselves should warn users that they need to become much more familiar with the operating system of their computer in order to most effectively use that RDBMS. Unfortunately, it is an incorrect assumption on the part of RDBMS developers that their clients will be familiar with their hardware and operating system. While many offices start out using their computers for fairly basic operations like word processing or spread sheets, this limited knowledge will not suffice when it comes to using the RDBMS. For example, because the CDO is using a personal computer that does not have a lot of memory, it is crucial that they become familiar with using the DOS operating system to perform file maintenance, deleting old files that are not used any more and are just taking up space. For INFORMIX, it is crucial that users become familiar with the editor available to them in their computer environment, because much of the programming will be done with that editor. For example, it was extremely useful to me to be able to use the editor to copy the application programs I had written, because I could modify each copied code to do different things, without having to reprogram all the code from scratch.

As noted earlier, it is not easy to translate an original design concept into an implementable version that can be handled by the software package. The same holds true for the translation of a design into a version that the operating system can handle. Consider the need to name the database, tables and fields. This is not a simple task. There are at least two major questions that have to be asked: what can the operating system handle and what can the RDBMS handle? Even if the RDBMS allows long names, the operating system may not. For example, DOS requires that the name be a maximum of 8 digits long. If the RDBMS allows longer names, these will in fact be truncated by DOS to 8 digits and this may cause problems for file maintenance (because DOS only recognizes the first 8 unique digits). For a product like RBASE, which is only available on a PC platform, this is not so much of a problem because the RDBMS itself requires name lengths that the operating system can handle. But for packages that are transportable across platforms, like PROGRESS and INFORMIX, it can be a problem if one does not conform to the least capable system's naming conventions. Yet doing this means not taking full advantage of the RDBMS's capabilities.

Even taking advantage of our RDBMS (for example with INFORMIX which allows table and field names up to 18 characters long) does not mean it is producing files with names 18 characters long. When INFORMIX actually stores these tables in an operating system directory, the name of the file containing the table information is truncated to 7 characters long and a unique number (starting from 100) is added to the 7 characters. So if the first 7 characters of table names are not distinct, the only way a user can tell which file contains the table information he/she is concerned with is by knowing what the unique number for that table is. For ease of maintaining the system being developed, the users must be able to, from the name of the file, identify what type of information is contained within. If a user has to refer to a "chart of accounts" for each file to identify what information is in it (because the name is not indicative of the information) then maintenance becomes that much more difficult.

While this is may not seem like a reason for concern, it did cause another interesting problem when making changes to the table and field information. INFORMIX, during updates, rather than saving the changed information into the same file/table, would save the information into a new table with a new unique number, and the old table would be deleted. While this does not cause any problems with the database, it does cause problems when creating a backup using the operating system utilities. Each time I backed up after I had made changes, the backup would have both the old and the new tables in it, because the old tables are not deleted from the backup. I had to manually delete all the old files from the backup. This problem may not exist if INFORMIX's backup utilities are used.

Determine, in advance of design, the characteristics of the user population. Is there significant turnover in the area where the system will be used? Are the users sophisticated? (i.e., do they regularly use computers?). Will they be frequent users of the system or will usage be on an ad hoc basis? The answers to these and similar questions will enable one to build the correct amount of userfriendliness into the database applications. In designing the INFORMIX and PROGRESS versions of the Recruiting database, the userfriendliness of the applications did become an issue. I designed both versions (PROGRESS and INFORMIX) to be somewhat userfriendly, programming help messages for the data entry screens to assist the user with data entry. But there was still a lot of implied knowledge necessary to use the application. For example, the messages did not tell the user whether they had to use the Return key or the Escape key to get things processed. While some of this "implied" knowledge could be programmed into a help menu/function, it did not seem worthwhile to spend the time to do the programming when with one session at a computer, this type of information would become common knowledge to the user. In the case of the CDO, the same users would be using the database frequently, therefore this type of information would be unnecessary. In the case of the Alumni database, with a diverse set of ad hoc users, extensive help messages is probably warranted.

In addition, if time were spent to program an application to "spoon feed" the user in a data entry process, the spoon feeding would eventually become cumbersome, because a regular user would not want to waste time being spoon fed each time he/she used the application. It was difficult to identify the line between being userfriendly and being overfriendly during this project because the functions being automated were relatively simple to begin with. But it is evident that there is a tradeoff to be made and the decision on where the line should be drawn must rest with the end user.

<u>Systems developers, if separate from the users, should continually communicate with the end users.</u> Discussion can not only take place during the design phase. As my project indicates, a system developed by a consultant will not be valid unless there has been continual discussion and verification with the end user. And the usefulness of the final system will depend on the quality of that communication. Even if the quality of the communication is good, there will still be questions that arise during the programming of a system that could not have been anticipated during initial design stages, questions which will require user input. The consultant must be flexible enough to design a system that meets the users' needs, even if the design is not optimal from a theoretical standpoint.

The overall conclusion that this thesis can draw from the experience of the systems development project is that systems development is not static. A conceptual design that one has of a system may be far removed from the actual system that is developed for a number of reasons:

- Hardware
- Software
- Systems Developer
- Interested Parties

Given the enormity of the task and the myriad of issues that are raised, an incremental approach to systems development is probably the only way working systems will ever be completed and used at Sloan.

APPENDIX A
CDO RECRUITING INPUT DOCUMENTS

MIT SLOAN SCHOOL OF MANAGEMENT
RECRUITING RESERVATION FORM

Company:____

Division:____

Contact:____

Title:____

Address:____

____

____

Telephone:______________________________________

Alternate Contact:______________________________

Telephone:______________________________________

Presentation Date:______________________________

****

Interview Day/Date:_____________________________

	P/S	O/C	CL?	Min.	Position(s)
1.					
2.					
3.					
4.					

Job description received:

****

Interview Day/Date:_____________________________

	P/S	O/C	CL?	Min.	Position(s)
1.					
2.					
3.					
4.					

Job description received:

MIT SLOAN SCHOOL OF MANAGEMENT
PRESENTATION RESERVATION FORM

Company:____

Division:____

Contact Name:____

Address:____

____

____

Telephone:____

Day/Date:____

Time: 12-1 / 5:30-7

Place: On-campus / Off-campus

Speakers Names & Titles:

____

____

____

Audio-visual equipment needed:

____

____

Food/Catering arrangements:

Recruiting for: Summer / Permanent / Both

Reservations taken by:________________

Date:__________________________

MIT Sloan School of Management

Job Information Form

(Please type. Complete a separate form for each position you wish to fill.)

Career Development Office
Room E52-111
50 Memorial Drive
Cambridge, MA 02139
(617) 253-6149

Job Title: ____________________

Recruiting Date(s) for this position: ____________________

Check one: ___ Permanent position
___ Summer position

Organization: ____________________

Division: ____________________

Parent Company, if relevant: ____________________

Recruiting Contact: ____________________ Title: ____________________

Street: ____________________ City: ____________________

State: __________ Zip Code: __________ Telephone: ____________________

# of Open Schedules for this Position: __________ Interview Length (30, 45, 60 min.) __________

# of Closed Schedules for this Position: __________ Interview Length (30, 45, 60 min.) __________

Primary Industry Code (check one)

Manufacturing
- ___ Agricultural Production
- ___ Oil &Gas Extraction
- ___ Food & Kindred Products
- ___ Tobacco Products
- ___ Apparel & Other Textile Products
- ___ Lumber &Wood Products
- ___ Paper & Allied Products
- ___ Printing & Publishing
- ___ Chemicals & Allied Products/Pharmaceuticals
- ___ Petroleum & Coal Products
- ___ Rubber & Plastics Products
- ___ Primary Metal Industries
- ___ Industrial Machinery & Equipment/Computers
- ___ Electronic & Other Electric Equipment
- ___ Transportation Equipment
- ___ Instruments & Related Products
- ___ Other Manufacturing (please specify): ____________________

Nonmanufacturing
- ___ Construction
- ___ Rail Transportation
- ___ Air Transportation
- ___ Communications
- ___ Utilities/Energy Services
- ___ Wholesale
- ___ Retail
- Finance, Insurance & Real Estate
  - ___ Commercial Banking
  - ___ Investment Banking/Brokerage
  - ___ Insurance
  - ___ Real Estate
  - ___ Investment Management/Venture Capital
- ___ Hotels & Other Lodging
- ___ Advertising Services
- ___ Computer & Data Processing Services/Software
- ___ Health Services
- ___ Legal Services
- ___ Educational Services
- ___ Social Services
- ___ Accounting & Auditing
- ___ Management Consulting
- ___ Government
- ___ Other Services (please specify): ____________________

Function Codes (check all that apply to position)

___	01	Accounting & Control	___	10	International Management
___	02	Applied Economics	___	11	Law
___	03	Consulting	___	12	Management Development Program
___	04	Corporate Strategy/Planning	___	13	Marketing/Sales
___	05	Finance	___	14	Operations/Production
___	06	General Management	___	15	Operations Research/Statistics
___	07	Health Care Management	___	16	Real Estate
___	08	Human Resources	___	17	System Dynamics
___	09	Information Systems	___	18	Technology Management

(See over for Job Information)

Job Information Form

If you wish, rather than using the space provided below, you may attach a one-page job description to this form.

Brief Profile of Organization: ______

Position Responsibilities: ______

Qualifications/Background (e.g., education, experience, citizenship, personal qualities): ______

Hiring Locations: ______

Number of MBA graduates you plan to hire for this job: ______

Base Salary Range: $ ______ (annual) or $ ______ (monthly)

Year-end bonus? Yes ___ No ___ Stock Options? Yes ___ No ___ Profit Sharing? Yes ___ No ___

Policy on drug screening/psychological testing: ______

Your signature attests that your organization considers candidates for all job opportunities without regard to race, creed, color, national origin, sex, age, handicap, or sexual orientation.

Signature: ______ Date: ______

APPENDIX B
POSITION NUMBER GENERATING PROGRAMS

PROGRESS POSITION NUMBER GENERATING PROGRAMS

```
Define variable nextpos# like lastpos#.


Repeat:
    {posform.f}     /* include file layout for the frame */

    Create position.
    Update companyid division postitle p-s functioncode function2 function3
           offcampus coordfirst coordlast coordphone.
    Find first syscontrol exclusive-lock.
    nextpos# - syscontrol.lastpos# + 1.
    syscontrol.lastpos# = nextpos#.
    position.position# = nextpos#.
    Display position#.

End.
```

```
Find first syscontrol.
Display lastpos# Label "Old position number value" with frame a.
syscontrol.lastpos# = 0.
Display lastpos# label "New position number value" with frame b.
```

RBASE POSITION NUMBER GENERATING PROGRAM

```
*(batch.mu -- multi-table batch autonumbering)
set zero on
set var vnuml integer
compute vmaxl as max pos# from position
enter recres
clear vmaxl vnuml
```

APPENDIX C
DATA DICTIONARIES

PROGRESS

Data Dictionary Report Legend

```
* - Indicates that a field participates in an index
# - Indicates the primary index for a database file
M - Indicates that a field is mandatory
```

```
08/02/89                PROGRESS Data Dictionary Report
Database: recruit
                                        schedule File
                                        =============
                  (Schedule information for each unique position.)

Frozen: no

Delete Validation
    Criterion: {schedule.v}
      Message: Cannot delete schedule if interview slot exists.

    Field        Type  Ext Dec Format                         Init
    ------------ ----- --- --- ------------------------------ ----------
* M position#    int           >>9                            0
* M schedule#    int           9                              0
    intvlength   int           99                             0
*   sdate        date          99/99/99
    room         char          x(7)
    c-o          logic         c/o                            o
    coverletter  logic         yes/no                         no

    Field-Name   Label                          Col-label
    ------------ ------------------------------ -------------------------
    position#    Position Number                Position!Number
    schedule#    Schedule Number                Schedule!Number
    intvlength   Interview Length               Interview!Length
    sdate        Date                           Date
    room         Room                           Room
    c-o          Closed/Open                    Closed/!Open
    coverletter  Cover Letter?                  Cover!Letter?

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
# pos-schedule yes    position#      1 yes       no
                      schedule#      2 yes       no
  sdate        no     sdate          1 yes       no

Field Validation Criteria, Validation Messages
----------------------------------------------------------------------
position#    :  can-find (position of schedule)
                Invalid position number.  Please reenter.
schedule#    :  schedule# > 0 and schedule# < 10
                Valid schedule numbers range from 1 to 9 inclusive.
intvlength   :  lookup(intvlength, "00,30,45,60") <> 0
                Interview length must be either 30, 45 or 60 minutes.

Help Messages
----------------------------------------------------------------------
position#    :  Enter position number for this position.
schedule#    :  Enter unique 1 digit schedule number.
intvlength   :  Enter interview length in minutes, 30, 45 or 60.
sdate        :  Enter date of schedule as mm/dd/yy.
room         :  Enter room name or number.
c-o          :  Enter ""c"" for closed interview, leave ""o"" for open.
coverletter  :  Enter yes if cover letter required.
```

```
08/02/89                  PROGRESS Data Dictionary Report
Database: recruit
                                      sic File
                                      ========
                    (Standard Industrial Code classifications.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                         Init
    ------------ ----- --- --- ------------------------------ --------
  * M siccode    int           >>>9                           0
  M industryname char          x(40)

    Field-Name   Label                          Col-label
    ------------ ------------------------------ ----------------------
    siccode      SIC Code                       SIC!Code
    industryname Industry Name                  Industry Name

  Index Name   Unique Field Name    Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
# siccode      yes    siccode        1 yes       no


Help Messages
----------------------------------------------------------------------
siccode      :  Enter unique 1 to 4 digit Standard Industrial Code.
industryname :  Enter name of industry pertaining to the SIC code.

                                   syscontrol File
                                   ===============
                            (System control information.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                         Init
    ------------ ----- --- --- ------------------------------ ---------
    lastpos#     int           >>9                            0

  Index Name   Unique Field Name    Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
* default      no
```

```
08 02 89                PROGRESS Data Dictionary Report
Database: recruit
                                     present File
                                     ============
                           (Company presentation information.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                        Init
    ------------ ----- --- --- ----------------------------- ---------------
* M companyid    char          x(8)
  M division     char          x(20)
*   pdate        date          99/99/99
    ptime        int           9999
    location     char          x(10)
    p-s-b        char          x(1)
    contactfirst char          x(12)
    contactlast  char          x(15)
    contactphone char          (999) 999-9999
    av           char          x(15)
    food         char          x(15)

    Field-Name   Label                          Col-label
    ------------ ------------------------------ ------------------------------
    companyid    Company ID                     Company!ID
    division     Division                       Division
    pdate        Date                           Date
    ptime        Time                           Time
    location     Location                       Location
    p-s-b        Permanent/Summer               P/S/B
    contactfirst Contact First Name             Contact!First Name
    contactlast  Contact Last Name              Contact!Last Name
    contactphone Contact Phone #                Contact!Phone Number
    av           Audio/Visual                   Audio/!Visual
    food         Food                           Food

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
  companyid    no     companyid      1 yes       no
# pdate        no     pdate          1 yes       no

Field Validation Criteria, Validation Messages
------------------------------------------------------------------------------
companyid     :  can-find (company of present)
                 Invalid companyid.  Please reenter an existing ID.

Help Messages
------------------------------------------------------------------------------
companyid     :  Enter the alpha code for company name.
division      :  Enter name of division.  If none, enter ""Corporate"".
pdate         :  Enter presentation date as mm/dd/yy.
ptime         :  Enter time as 4 digit string: 1000 for 10:00am.
location      :  Enter the room/location of the presentation.
p-s-b         :  Enter ""p"" for permanent, ""s"" for summer, ""b"" for both.
contactfirst  :  Enter the first name of the presentation contact.
contactlast   :  Enter last name of presentation contact.
contactphone  :  Enter phone number of presentation contact.
av            :  Enter audio/visual requirements of presentation.
food          :  Enter catering information for presentation.
```

```
08/02/89                    PROGRESS Data Dictionary Report
Database: recruit
                                          position File    (continued)
                                          =============
        (Unique positions being recruited for by interviewing companies.)

Help Messages
-----------------------------------------------------------------------------
division      :  Enter name of division.  If none, enter ""Corporate"".
postitle      :  Enter position title.
p-s           :  Enter ""s"" for summer, leave ""p"" for permanent.
functioncode  :  Enter job function code.
function2     :  Enter code for secondary job function.
function3     :  Enter code for third job function of position.
offcampus     :  Enter yes if interview will be help off campus.
coordfirst    :  Enter the first name of the interview coordinator.
coordlast     :  Enter last name of interview coordinator.
coordphone    :  Enter phone number of interview coordinator.
```

```
08 02/89                 PROGRESS Data Dictionary Report
Database: recruit
                                       position File
                                       =============
        (Unique positions being recruited for by interviewing companies.)

Frozen: no

Delete Validation
    Criterion: not (can-find(schedule of position))
      Message: Cannot delete position if schedule for interviews exists.

    Field        Type  Ext Dec Format                          Init
    ------------ ----- --- --- ------------------------------- ------------
* M position#    int           >>9                             0
* M companyid    char          x(8)
  M division     char          x(20)
    postitle     char          x(35)
    p-s          logic         -p/s                            p
    functioncode int           >9                              0
    function2    int           >9                              0
    function3    int           >9                              0
    offcampus    logic         yes/no                          no
    coordfirst   char          x(12)
    coordlast    char          x(15)
    coordphone   char          (999) 999-9999

    Field-Name   Label                          Col-label
    ------------ ------------------------------ ---------------------------
    position#    Position Number                Position!Number
    companyid    Company ID                     Company!ID
    division     Division                       Division
    postitle     Position Title                 Position!Title
    p-s          Permanent/Summer               P/S
    functioncode Primary Function               Primary!Function
    function2    Function Code 2                Function!Code 2
    function3    Function Code 3                Function!Code 3
    offcampus    Off Campus?                    Off!Campus?
    coordfirst   Coordinator First Name         Coordinator!First Name
    coordlast    Coordinator Last Name          Coordinator!Last Name
    coordphone   Coordinator Phone Number       Coordinator!Phone Number

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
* companyid    no     companyid      1 yes       no
  position#    yes    position#      1 yes       no

Field Validation Criteria, Validation Messages
-----------------------------------------------------------------------------
companyid    :  can-find (company of position)
                Invalid companyid.  Please reenter an existing id.
functioncode :  {posfunc.v}
                Invalid function code.  Please reenter.

Help Messages
-----------------------------------------------------------------------------
position#    :  Position # for this table is automatically generated.
companyid    :  Enter the alpha code for company name.
```

```
08/02/89               PROGRESS Data Dictionary Report
Database: recruit
                                  intvslot File
                                  =============
                (Slots for each individual interview schedule.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                         Init
    ------------ ----- --- --- ------------------------------ ---------
* M position#    int           >>9                            0
* M schedule#    int           9                              0
* M itime        int           9999
    studentfirst char          x(12)
    studentlast  char          x(15)

    Field-Name   Label                          Col-label
    ------------ ------------------------------ -----------------------
    position#    Position Number                Position!Number
    schedule#    Schedule Number                Schedule!Number
    itime        Time                           Time
    studentfirst Student First Name             Student!First Name
    studentlast  Student Last Name              Student!Last Name

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
  pos-schedule no     position#      1 yes       no
                      schedule#      2 yes       no
# slot         yes    position#      1 yes       no
                      schedule#      2 yes       no
                      itime          3 yes       no

Field Validation Criteria, Validation Messages
-----------------------------------------------------------------------
position#    :  can-find (position of intvslot)
                Must enter existing position number.
schedule#    :  {intvslot.v}
                Must use number for an existing schedule.

Help Messages
-----------------------------------------------------------------------
position#    :  Enter position # for this interview schedule.
schedule#    :  Enter the 1 digit number for this interview schedule.
itime        :  Enter time as 4 digit string: 1000 for 10:00am.
studentfirst :  Enter the first name of the student.
studentlast  :  Enter last name of student.
```

```
08 02 89                 PROGRESS Data Dictionary Report
Database: recruit
                                          country File
                                          ============
                                (Country code and country name table.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                          Init
    ------------ ----- --- --- ------------------------------ -----------
* M countrycode  char          x(2)                            US
  M countryname  char          x(13)

    Field-Name   Label                           Col-label
    ------------ ------------------------------ -------------------------
    countrycode  Country Code                    Country!Code
    countryname  Country Name                    Country!Name

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
# countrycode  yes    countrycode    1 yes       no


Help Messages
---------------------------------------------------------------------------
countrycode  :  Enter unique 2 character alpha country code.
countryname  :  Enter name of country for this country code.
                                        function File
                                        =============
                (Function codes and function names for job functions.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                          Init
    ------------ ----- --- --- ------------------------------ ------------
* M functioncode int           >9                              0
  M functionname char          x(30)

    Field-Name   Label                           Col-label
    ------------ ------------------------------ -------------------------
    functioncode Function Code                   Function!Code
    functionname Function Name                   Function!Name

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
# functioncode yes    functioncode   1 yes       no


Help Messages
---------------------------------------------------------------------------
functioncode :  Enter unique function code.
functionname :  Enter name describing function.
```

```
                              contact File    [continued]
                              ============
                          (Contact information.)

    Field-Name   Label                          Col-label
    ------------ ------------------------------ -------------------------
    present      Presentation Contact?          Presentation!Contact?
    recruit      Recruiting Coordinator?        Recruiting!Coordinator?
    alumni       Alumni?                        Alumni?
    resumebook   Resume Book?                   Resume!Book?
    contactdate  Date                           Date

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
# companyid    no     companyid      1 yes       no
  lastname     no     lastname       1 yes       yes

Field Validation Criteria, Validation Messages
------------------------------------------------------------------------
companyid    : can-find (company of contact)
               Invalid companyid.  Enter existing companyid.
functioncode : {confunc.v}
               Invalid function code. Please reenter.
zipcode      : zipcode > 0
               Zipcode must be a 5 digit number.
countrycode  : can-find (country of contact)
               Invalid country code.  Please reenter.

Help Messages
------------------------------------------------------------------------
companyid     : Enter the alpha code for company name.
division      : Enter division name.  If none, enter ""Corporate"".
firstname     : Enter the first name of the contact.
lastname      : Enter last name of contact.
gender        : Enter f for female, leave as m for male.
contacttitle  : Enter contact's title.
functioncode  : Enter job function code.
function2     : Enter code for secondary job function.
street        : Enter street address.
city          : Enter name of city.
state         : Enter state of contact's address.
zipcode       : Enter 5 digit zipcode.
countrycode   : Enter 2 character alpha country code.
phone         : Enter phone number of contact.
fax#          : Enter fax number of contact.
fedexpress#   : Enter contact's federal express number.
speaker       : Enter yes if contact was speaker at company presentation.
interviewer   : Enter yes if contact conducted interviews.
present       : Enter yes if coordinator of company presentation.
recruit       : Enter yes if coordinator for company recruiting.
alumni        : Enter yes if contact is a Sloan alumni.
resumebook    : Enter yes if purchased Sloan resume book.
contactdate   : Enter date (mm/dd/yy) contact become known.
```

```
08/02/89              PROGRESS Data Dictionary Report
Database: recruit
                                     contact File
                                     ============
                                 (Contact information.)

Frozen: no

Delete Validation
    Criterion:
      Message:

    Field        Type  Ext Dec Format                         Init
    ------------ ----- --- --- ------------------------------ --------
* M companyid    char          x(8)
  M division     char          x(20)
  M firstname    char          x(12)
* M lastname     char          x(15)
    gender       logic         m/f
    contacttitle char          x(35)
    functioncode int           >9                             0
    function2    int           >9                             0
    street       char          x(35)
    city         char          x(15)
    state        char          x(2)
    zipcode      int           99999                          0
    countrycode  char          x(2)                           US
    phone        char          (999) 999-9999
    fax#         char          (999) 999-9999
    fedexpress#  char          9999-9999-9
    speaker      logic         yes/no                         no
    interviewer  logic         yes/no                         no
    present      logic         yes/no                         no
    recruit      logic         yes/no                         no
    alumni       logic         yes/no                         no
    resumebook   logic         yes/no                         no
    contactdate  date          99/99/99

    Field-Name   Label                          Col-label
    ------------ ------------------------------ ------------------------
    companyid    Company ID                     Company!ID
    division     Division                       Division
    firstname    First Name                     First Name
    lastname     Last Name                      Last Name
    gender       Gender                         Gender
    contacttitle Title                          Title
    functioncode Primary Function Code          Primary!Function Code
    function2    Function Code 2                Function!Code 2
    street       Street                         Street
    city         City                           City
    state        State                          ST
    zipcode      Zip Code                       Zip Code
    countrycode  Country Code                   Country!Code
    phone        Phone Number                   Phone Number
    fax#         Fax Number                     Fax Number
    fedexpress#  Federal Express #              Federal!Express #
    speaker      Speaker?                       Speaker?
    interviewer  Interviewer?                   Interviewer?
```

```
08 02 89                PROGRESS Data Dictionary Report
Database: recruit
                                  company File
                                  ============
                           (Primary company information.)

Frozen: no

Delete Validation
    Criterion: {company.v}
      Message: Cannot delete company that is currently recruiting.

    Field        Type  Ext Dec Format                         Init
    ------------ ----- --- --- ------------------------------ ---------
* M companyname  char          x(45)
* M companyid    char          x(8)
    street       char          x(35)
    city         char          x(15)
    state        char          x(2)
    zipcode      int           99999                          0
    countrycode  char          x(2)                           US
    siccode      int           >>>9                           0
    sponsor      logic         yes/no                         no

    Field-Name   Label                          Col-label
    ------------ ------------------------------ ----------------------
    companyname  Company Name                   Company!Name
    companyid    Company ID                     Company!ID
    street       Street                         Street
    city         City                           City
    state        State                          ST
    zipcode      Zip Code                       Zip Code
    countrycode  Country Code                   Country!Code
    siccode      SIC Code                       SIC!Code
    sponsor      Sloan Sponsor?                 Sloan!Sponsor?

  Index Name   Unique Field Name   Seq Ascending abbreviate
  ------------ ------ ------------ --- --------- ----------
# companyid    yes    companyid      1 yes       no
  companyname  no     companyname    1 yes       no

Field Validation Criteria, Validation Messages
----------------------------------------------------------------------
zipcode      :  zipcode > 0
                Zipcode must be a 5 digit number.
countrycode  :  can-find(country of company)
                Country code not valid.  Please reenter.
siccode      :  can-find (sic of company)
                Invalid sic code.  Please reenter.

Help Messages
----------------------------------------------------------------------
companyname  :  Enter full name of company.
companyid    :  Enter a unique alpha code up to 8 characters.
street       :  Enter street address.
city         :  Enter name of city.
state        :  Enter state of company address.
zipcode      :  Enter 5 digit zipcode.
countrycode  :  Enter 2 character alpha country code.
siccode      :  Enter 1 to 4 digit Standard Industrial Code.
sponsor      :  Enter yes if company is a Sloan Sponsor.
```

INFORMIX DATA DICTIONARY

```
INFO - company:  Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

---------------------- recruit ---------------- Press CTRL-W for Help --------

Column name         Type                Nulls

companyname         char(45)            no
companyid           char(8)             no
siccode             smallint            yes
sponsor             char(1)             yes
```

```
INFO - company:  Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

---------------------- recruit ---------------- Press CTRL-W for Help --------

Index name          Owner     Type    Cluster  Columns

ix100_1             lmccaff   dupls   No       companyname

ix100_2             lmccaff   unique  No       companyid
```

```
INFO - company:  Columns  Indexes  Privileges  Status  Table  Exit
Display status information for a table.

---------------------- recruit ---------------- Press CTRL-W for Help --------


Table Name          company
Owner               lmccaff
Row Size            56
Number of Rows      13
Number of Columns   4
Date Created        08/10/1989
Audit Trail File
```

```
INFO - company:  Columns  Indexes  Privileges  Status  Table  Exit
Display user access privileges for a table.

---------------------- recruit ---------------- Press CTRL-W for Help --------

User          Select             Update            Insert  Delete  Index  Alter

public         All                All               Yes     Yes     Yes    No
```

```
RUN - contact:   Next  Restart  Exit
Display the next page of query results.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name          Type                 Nulls

companyid            char(8)              no
division             char(20)             yes
firstname            char(12)             no
lastname             char(15)             no
gender               char(1)              yes
contacttitle         char(35)             yes
primaryfunction      smallint             yes
function2            smallint             yes
street1              char(35)             yes
street2              char(35)             yes
city                 char(15)             yes
state                char(2)              yes
zipcode              char(10)             yes
countrycode          char(2)              yes
```

```
INFO - contact:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name          Type                 Nulls

phone                char(12)             yes
ext                  char(5)              yes
faxnumber            char(12)             yes
fedexpress           char(11)             yes
speaker              char(1)              yes
interviewer          char(1)              yes
presentcoord         char(1)              yes
recruitcoord         char(1)              yes
alumni               char(1)              yes
resumebook           char(1)              yes
contactdate          date                 yes
```

```
INFO - contact:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name           Owner     Type     Cluster  Columns

ix101_1              lmccaff   dupls    No       companyid

ix101_4              lmccaff   dupls    No       lastname

uniqcontact          lmccaff   dupls    No       companyid
                                                 firstname
                                                 lastname
```

```
INFO - position:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name           Type                Nulls

posnumber             serial              no
companyid             char(8)             no
posdivision           char(20)            no
postitle              char(35)            yes
p_s                   char(1)             yes
primjobfunc           smallint            yes
jobfunc2              smallint            yes
jobfunc3              smallint            yes
offcampus             char(1)             yes
jobdes                char(1)             yes
coordfirst            char(12)            yes
coordlast             char(15)            yes
coordphone            char(12)            yes
coordext              char(5)             yes
```

```
INFO - position:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name           Owner     Type    Cluster  Columns

ix110_1              lmccaff   unique  No       posnumber

ix110_2              lmccaff   dupls   No       companyid
```

```
INFO - schedule:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name            Type                Nulls

posnumber              smallint            no
schednumber            smallint            no
intvlength             smallint            yes
sdate                  date                yes
room                   char(7)             yes
c_o                    char(1)             yes
coverletter            char(1)             yes
```

```
INFO - schedule:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name             Owner     Type     Cluster  Columns

ix112_4                lmccaff   dupls    No       sdate

posschedule            lmccaff   unique   No       posnumber
                                                   schednumber
```

```
INFO - present:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name            Type                Nulls

companyid              char(8)             no
presdivision           char(20)            no
pdate                  date                yes
ptime                  char(5)             yes
location               char(10)            yes
p_s_b                  char(1)             yes
contactfirst           char(12)            yes
contactlast            char(15)            yes
contactphone           char(12)            yes
contactext             char(5)
```

```
INFO - present:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name             Owner     Type     Cluster  Columns

ix111_1                lmccaff   dupls    No       companyid

ix111_3                lmccaff   dupls    No       pdate
```

```
INFO - sic:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name          Owner     Type    Cluster  Columns

ix113_1             lmccaff   unique  No       siccode
```

```
INFO - sic:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name          Type                Nulls

siccode              smallint            no
industryname         char(45)            no
```

```
INFO - jobfunc:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name          Owner     Type    Cluster  Columns

ix136_1             lmccaff   unique  No       functioncode
```

```
INFO - jobfunc:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name          Type                Nulls

functioncode         smallint            no
functionname         char(30)            no
```

```
INFO - intvslot:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name            Type                Nulls

posnumber              smallint            no
schednumber            smallint            no
itime                  char(5)             no
studentfirst           char(12)            yes
studentlast            char(15)            yes
```

```
INFO - intvslot:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name          Owner      Type     Cluster  Columns

slot                lmccaff    unique   No       posnumber
                                                 schednumber
                                                 itime

posschedule2        lmccaff    dupls    No       posnumber
                                                 schednumber
```

```
INFO - country:   Columns  Indexes  Privileges  Status  Table  Exit
Display column names and data types for a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Column name            Type                Nulls

countrycode            char(2)             no
countryname            char(13)            no
```

```
INFO - country:   Columns  Indexes  Privileges  Status  Table  Exit
Display information about indexes for the columns in a table.

----------------------- recruit ---------------- Press CTRL-W for Help --------

Index name          Owner      Type     Cluster  Columns

ix102_1             lmccaff    unique   No       countrycode
```

RBASE DATA DICTIONARY

```
LIST TABLES function

   Table: function        No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length          Key   Expression
   1 funccode INTEGER                  yes
   2 funcname TEXT      30 characters

   Current number of rows:       0



LIST TABLES company

   Table: company         No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length          Key   Expression
   1 compname TEXT      45 characters yes
   2 compid   TEXT       8 characters yes
   3 siccode  INTEGER
   4 sponsor? TEXT       1 characters



LIST TABLES intvslot

   Table: intvslot        No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length          Key   Expression
   1 pos#     INTEGER
   2 sched#   INTEGER
   3 itime    TEXT       5 characters yes
   4 studentf TEXT      12 characters
   5 studentl TEXT      15 characters

   Current number of rows:       0

Press any key to continue
```

```
LIST TABLES industry

   Table: industry        No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length           Key   Expression
   1 siccode  INTEGER                   yes
   2 indname  TEXT      40 characters



   Table: contact         No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length           Key   Expression
   1 compid   TEXT       8 characters yes
   2 division TEXT      20 characters
   3 namef    TEXT      12 characters
   4 namel    TEXT      15 characters yes
   5 gender   TEXT       1 characters
   6 title    TEXT      35 characters
   7 primfunc INTEGER
   8 func2    INTEGER
   9 street   TEXT      35 characters
  10 street2  TEXT      35 characters
  11 city     TEXT      15 characters
  12 state    TEXT       2 characters
  13 zipcode  TEXT      10 characters
  14 cntrycd  TEXT       2 characters
  15 phone    TEXT      12 characters
  16 ext.     TEXT       5 characters
  17 fax#     TEXT      12 characters
  18 fedexp#  TEXT      11 characters
  19 speaker? TEXT       1 characters
  20 intvwr?  TEXT       1 characters
  21 present? TEXT       1 characters
  22 recruit? TEXT       1 characters
  23 alumni?  TEXT       1 characters
  24 rsumebk? TEXT       1 characters
  25 date     DATE
  26 computed TEXT      35 characters       'compid'+'namef'+'na
                                            mel'

    Current number of rows:     37

 Press any key to continue
```

```
LIST TABLES country

   Table: country         No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length         Key   Expression
   1 cntrycd  TEXT       2 characters yes
   2 name     TEXT      14 characters


LIST TABLES position

   Table: position        No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name     Type     Length         Key   Expression
   1 pos#     INTEGER                 yes
   2 controll INTEGER                        (1*1)
   3 compid   TEXT       8 characters yes
   4 posdiv   TEXT      20 characters
   5 postitle TEXT      35 characters
   6 p/s      TEXT       1 characters
   7 jobfunc1 INTEGER
   8 jobfunc2 INTEGER
   9 jobfunc3 INTEGER
  10 offcamp? TEXT       1 characters
  11 coordf   TEXT      12 characters
  12 coordl   TEXT      15 characters
  13 jobdes?  TEXT       1 characters

   Current number of rows:     27

Press any key to continue
```

```
LIST TABLES present

   Table: present         No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name      Type      Length          Key    Expression
   1 compid    TEXT       8 characters   yes
   2 presdiv   TEXT      20 characters
   3 pdate     DATE                      yes
   4 ptime     TIME
   5 location  TEXT      10 characters
   6 p/s/b     TEXT       1 characters
   7 contactf  TEXT      12 characters
   8 contactl  TEXT      15 characters
   9 av        TEXT      15 characters
  10 food      TEXT      15 characters


LIST TABLES schedule

   Table: schedule        No lock(s)
   Read Password: No
   Modify Password: No

   Column definitions
   # Name      Type      Length          Key    Expression
   1 pos#      INTEGER
   2 sched#    INTEGER                   yes
   3 intlgnth  INTEGER
   4 sdate     DATE                      yes
   5 room      TEXT       7 characters
   6 c/o       TEXT       1 characters
   7 cvrltr?   TEXT       1 characters

   Current number of rows:     50

Press any key to continue
```

APPENDIX D
RECRUITING RESERVATION PROGRAMS/INPUT FORMS

PROGRESS RECRUITING RESERVATION PROGRAM

```
*************************************************************************************
File Name: recres.p
Form Name: recres.f
Title: Recruiting Reservation Input Procedure
Description:  Procedure for inputting reservations for company recruiting
efforts.
File Names: company, contact, position, schedule.  This procedure updates
information in the aforementioned files.
Database Name: recruit.
*************************************************************************************/

/* The topmost repeat loop allows the user to enter subsequent recruiting
reservations without having to exit the procedure. */

Define variable answer as logical.
Answer = YES.

Repeat:
    {recres.f}       /* Include file containing recruiting reservation form */
    Prompt-for company.companyname company.companyid company.siccode.
    Find company using companyid no-error.

/* This portion of code checks to see if a company already exists with this
companyid.  If it does, the program verifies with the user that the companyname
found is the companyname desired.  If not (i.e. if the user did not enter a
unique ID for this company) the program tells the user to reenter a unique ID.
If it is the company the user wants, the program procedes.  If no company is
found with the input companyid, then a new company record is created in the
company table. */

    If available company then do:
    Message "The company" companyname "has been assigned this ID.".
    Update answer label "Is this the company you want?" with frame a.
    If answer then do:
    Message "Proceed to enter contact information".
    Pause 2 no-message.
    hide no-pause.
    end.
    Else do:
    create company.
    assign input companyname input siccode.
    Message "Enter a unique companyid.".
    prompt-for companyid.
    assign companyid.
    end.
    end.
    If not available company then do:
    Message "A company does not exist with this company ID.".
    Message "A new company record will be created.".
    Create company.
    Assign companyname companyid siccode.
    Pause 2 no-message.
    Hide no-pause.
    end.
    Display companyname company.companyid siccode.
```

```
* This section of code checks whether or not the presentation contact is
already in the contact table.  If he/she is, the program skips to the portion
where presentation specifics are added.  If he/she is not, the program procedes
to the contact to the contact table and allows the user to enter additional
contact information into the contact table. */
            Prompt-for contact.division firstname lastname.
            find contact where contact.companyid = company.companyid and
            contact.division = input contact.division and firstname
            = input firstname and lastname = input lastname no-error.
            if available contact then do:
            Message "This coordinator is already in the contact file.".
            Message "Proceed to enter position/schedule specifics.".
            end.
            If not available contact then do:
                create contact.
                contact.companyid = company.companyid.
                assign contact.division firstname lastname.
                update contacttitle contact.street contact.city contact.state
                contact.zipcode
                contact.phone contact.fedexpress# contact.fax#.
            End.

            Display firstname lastname contacttitle.

/* This portion of code checks to see if there is company address information
in the company table for this companyid.  If no address has been put in the
company table, the program assigns the company with the same address
information as the recruiting coordinator.  If the company record already has
an address, then no changes are made. */

    if company.street = " " then do:
    company.street = contact.street.
    end.
    if company.city = " " then do:
    company.city = contact.city.
    end.
    if company.state = " " then do:
    company.state = contact.state.
    end.
    if company.zipcode = 00000 then do:
    company.zipcode = contact.zipcode.
    end.

    Define variable nextpos# like lastpos#.

/* This repeat loop allows the user to create more than one position for each
company recruiting on campus, without having to exit the procedure.  Each new
position is assigned a unique position number. */

    Repeat:
    {recres.f}
        Display companyname company.companyid contact.division
        firstname lastname contacttitle.
        create position.
        find first syscontrol exclusive-lock.
        nextpos# = syscontrol.lastpos# - 1.
        syscontrol.lastpos# = nextpos#.
        position.position# = nextpos#.
        display position#.
        position.companyid = contact.companyid.
        position.division = contact.division.
        position.coordfirst = contact.firstname.
        position.coordlast = contact.lastname.
        position.coordphone = contact.phone.
        update postitle p-s.

/* This repeat loop allows the user to create more than one schedule for a
particular position (i.e. unique position number) without having to reenter the
position information. */


            Repeat:
            {recres.f}

        Display companyname company.companyid contact.division
        firstname lastname contacttitle position# postitle p-s.

                create schedule.
                schedule.position# = position.position#.
                update schedule.schedule# sdate c-o coverletter intvlength.
            End.
    End.
End.
```

INFORMIX RECRUITING RESERVATION PROGRAM

```
{Program: recresp.4gl
Database: recruit
Description: This program is designed to be the primary data entry screen for
inputting recruiting reservation information.  It reduces the amount of manual
input required by the CDO staff by inserting information that has been entered
only once, into more than one table, where necessary.
Form: recresf.per
Tables: company, contact, position, schedule }

database recruit

{This portion of the program, global variable definition, is where "program
variables" are defined.  These program variables take input from "screen
record variables" that are the actual interface with the user. The p before
each table name, selected for convenience, distinguishes the program variable
from the screen record and database values.  After the user enters input to
the screen (and into the screen record variable), it must be moved into a
program variable and then from the program variable it can then be inserted
into the actual database.}

globals
        define
        pcompany record
                companyid like company.companyid,
                companyname like company.companyname,
                siccode like company.siccode
                end record,

        pcontact record
                firstname like contact.firstname,
                lastname like contact.lastname,
                contacttitle like contact.contacttitle,
                street1 like contact.street1,
                street2 like contact.street2,
                city like contact.city,
                state like contact.state,
                zipcode like contact.zipcode,
                countrycode like contact.countrycode,
                phone like contact.phone,
                ext like contact.ext,
                faxnumber like contact.faxnumber,
                fedexpress like contact.fedexpress
                end record,

        pposition record
                posnumber like position.posnumber,
                postitle like position.postitle,
                posdivision like position.posdivision,
                p_s like position.p_s,
                offcampus like position.offcampus
                end record,

{The following array allows the user to enter up to 6 schedules into the
schedule table, for one position number.  Although the recresf.per form only
has one screen line for data entry of schedule information (because the screen
can only hold 20 lines of a form and informix doesn't allow multiple screen
data entry forms), by setting up a program array of schedules, the user can
enter subsequent schedules on the same line and have all successfully make it
to the database.}

        pscheds array [6] of record
                schednumber like schedule.schednumber,
                sdate like schedule.sdate,
                c_o like schedule.c_o,
                coverletter like schedule.coverletter,
                intvlength like schedule.intvlength
                end record,

        answer char(1)

end globals

{The main program establishes the looping that allows the user to enter
reservations for many companies, without having to leave the program.}

Main

        open form recresf from "recresf"
        display form recresf
        let answer = "y"
        while answer = "y"

                call enterrecres()

                        prompt "Do  you want to enter another reservation (y/n)
" for answer

        end while

        Message "Program Ended."
        Sleep 2
        Clear screen

End Main

{The enterrecres function (short for enter recruiting reservation) is actually
the piece of the recresp.4gl program that does most of the work.  It is in thi
function that the four tables are updated for new reservation information.}

Function enterrecres()
        define counter, schedcount smallint
        clear form

{This portion of the enterrecres function tests whether or not the recruiting
company already exists in the company table.  It first prompts the user for the
companyid. If the company is already in the company table, the program asks the
user if the company that was found is the one she wants. If it is, it displays
the company name and sic code and proceeds to the portion where the user enters
contact information.  If it is not the company she wants, she is then prompted
to enter a unique cpompany id, as well as the company name and sic code. Or if
the company is not in the company table yet, the program proceeds to prompt
the user for the company name and sic code.}

        message "Enter the company's ID"
        input by name pcompany.companyid
        select companyname, siccode into pcompany.companyname, pcompany.siccode
  from company  where companyid = pcompany.companyid
        if status = 0 then
        message pcompany.companyname clipped," has already been assigned this c
mpany id. "
        sleep 3
        message ""
        Prompt "Is the company you want (y/n)?" for answer
                if answer = "n" then

                display "To create a record for this company, enter company "
                at 2, 1
                display "name,a unique company id, and company sic code."
                at 3, 1

{Because companyid in the company table must be unique, the select statement
above serves to prevent any duplicate companyid's, by returning the value of
a company that has the same id as that entered by the user originally.  But
once the program has passed that portion of code, and the user is trying to
enter a new company into the company table, she will get a fatal error if she
tries to enter a non-unique companyid.  In an effort to prevent the fatal
error from kicking the user out of the program, the function comperr (short
for company error) was created and is called if at this point in the program
the user again enters a non-unique companyid.}

                whenever error call comperr

                        input pcompany.companyname, pcompany.companyid,
                        pcompany.siccode from companyname, companyid, siccode

                        insert into company (companyname, companyid, siccode)
                        values (pcompany.companyname, pcompany.companyid,
                        pcompany.siccode)

                        display "" at 2,1
                        display "" at 3,1
                        message "Company Added."
                        sleep 2
                        message ""
                else display pcompany.companyname, pcompany.companyid,
                        pcompany.siccode to companyname, companyid, siccode

                end if
        end if

        if status = notfound then
        input by name pcompany.companyname, pcompany.siccode
        insert into company (companyname, companyid, siccode) values
        (pcompany.companyname, pcompany.companyid, pcompany.siccode)
                        message "Company Added."
                        sleep 2
                        message ""
        end if

{This portion of the enterrecres function accepts contact information input
from the user and checks to see if the contact is already in the file. If
she is, then the program skips to the position and schedule data entry portion
of the recruiting reservation form.  If no contact exists with this companyid
and first and last name, the program prompts for additional contact information
entry.  This portion also records contact information (name, phone) in the
position table automatically. And lastly it automatically updates the
recruitcoord field in contact to be "y", indicating that this contact is a
recruiting coordinator. If this contact is new to the contact table, the
current date is inserted as the value for contactdate.}

        input by name pcontact.firstname thru pcontact.lastname
        select contacttitle, street1, street2, city, state, zipcode,
countrycode, phone, ext, faxnumber, fedexpress into pcontact.contacttitle,
pcontact.street1, pcontact.street2, pcontact.city, pcontact.state,
pcontact.zipcode, pcontact.countrycode, pcontact.phone, pcontact.ext,
pcontact.faxnumber, pcontact.fedexpress from contact where companyid =
pcompany.companyid and firstname = pcontact.firstname and lastname =
pcontact.lastname
        if status = 0 then
        update contact set recruitcoord ="y" where companyid =
                pcompany.companyid and firstname = pcontact.firstname and
                lastname = pcontact.lastname
        message "This contact is already in contact file."
        sleep 3
        message "Proceed to enter position/schedule specifics."
        sleep 3
        message ""
        end if

        if status = notfound then
        input by name pcontact.contacttitle thru pcontact.fedexpress
        insert into contact (companyid, firstname, lastname,contacttitle,
        street1, street2, city, state, zipcode, countrycode, phone, ext,
        faxnumber, fedexpress, recruitcoord, contactdate) values
        (pcompany.companyid,
        pcontact.firstname, pcontact.lastname, pcontact.contacttitle,
        pcontact.street1, pcontact.street2, pcontact.city, pcontact.state,
        pcontact.zipcode, pcontact.countrycode, pcontact.phone, pcontact.ext,
        pcontact.faxnumber, pcontact.fedexpress, "y", TODAY)
        message "Contact Added."
        message "Proceed to enter position/schedule specifics."
        sleep 3
        message ""
        end if

{This portion of the enterrecres function adds new positions to the position
table, with the system generating the unique position number.  The SQLCA
function ensures that the unique number generating function has worked
properly.}

        input by name pposition.postitle thru pposition.offcampus
        let pposition.posnumber = 0
        insert into position (companyid, posnumber, postitle, posdivision, p_s,
                offcampus, coordfirst, coordlast, coordphone, coordext) values
                 (pcompany.companyid, pposition.posnumber, pposition.postitle,
                  pposition.posdivision, pposition.p_s, pposition.offcampus,
                pcontact.firstname, pcontact.lastname, pcontact.phone,
                pcontact.ext)

        let pposition.posnumber = SQLCA.SQLERRD[2]

        display pposition.posnumber to posnumber
        message "Position Added."
        Sleep 2
        message ""
        message "Proceed to enter successive schedules; press return after each
"
        sleep 3
        message " Press escape when complete."
        sleep 3
        message ""

{This portion of enterrecres allows for multiple schedules to be input for
one position.}

        input array pscheds from scheds.*
        let schedcount = arr_count()
        for counter = 1 to schedcount
                        insert into schedule (posnumber, schednumber, sdate,
                                c_o, coverletter, intvlength)
                values (pposition.posnumber, pscheds[counter].schednumber,
                        pscheds[counter].sdate, pscheds[counter].c_o,
                        pscheds[counter].coverletter,
                        pscheds[counter].intvlength)
        end for
        message "Schedules Added."
        sleep 2
        message ""
end function

{This error function prevents users from entering a non-unique companyid a
second time and thereby being forced out of the program because of a fatal
error.  The error -239 is the informix error for an attempt to insert a
record into a field which requires a unique value.}

        Function comperr()
                whenever error continue
                if status = -239 then
                        Display "" at 2,1
                        Display "" at 3,1
                        Display "A company already exists with this ID." at 3,1
                        Prompt "Enter a unique ID:" for pcompany.companyid
                        insert into company (companyname, companyid, siccode)
                        values (pcompany.companyname, pcompany.companyid,
                        pcompany.siccode)
                        Display pcompany.companyid to companyid
                end if
        end Function
```

RBASE RECRUITING RESERVATION INPUT FORM

```
Press [ESC] for the menu
èëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë£
¤                    Recruiting Reservation Input Form                          ¤
¤                                                                               ¤
¤Company ID:                                                                    ¤
¤Company Name:                                                                  ¤
¤SIC Code:                                                                      ¤
¤Recruiting Division:                                                           ¤
¤Coordinator First Name:                                                        ¤
¤Coordinator Last Name:                                   Position #:           ¤
¤Permanent/Summer:                                                              ¤
¤Position title:                                          Offcampus?:           ¤
¤                                                                               ¤
Scheduleë#,ëDate,ëInterviewëLength,ëC/O,ëCoverëLetter?ëëëëëëëëëëëëëëëëëëëëëëëëë¿
¤  Schedule#:      Date:          Interview Length:      C/O:    CL?:           ¤
¤  Schedule#:      Date:          Interview Length:      C/O:    CL?:           ¤
¤  Schedule#:      Date:          Interview Length:      C/O:    CL?:           ¤
¤  Schedule#:      Date:          Interview Length:      C/O:    CL?:           ¤
¤  Schedule#:      Date:          Interview Length:      C/O:    CL?:           ¤
àëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëëë¥


[ESC] Done    [F2] Clear field    [Shift-F2] Clear to end    [Shift-F10] More
Form: recres     Table: company    Field: compid     Page: 1
```

APPENDIX E
PRESENTATION RESERVATION PROGRAMS/INPUT FORMS

```
      PROGRESS PRESENTATION RESERVATION PROGRAM
 ********************************************************************************
File Name: presres.p
Form Name: presres.f
Title: Presentation Reservation Input Form Procedure
Description: Procedure for inputting reservations for company presentations.
File Name: company contact present.  This procedure updates information in
each of the aforementioned files.
Database Name: recruit
********************************************************************************/

/* The topmost repeat loop allows the user to enter subsequent presentation
reservations without having to exit the procedure. */

Repeat:
    {presres.f}
    prompt-for company.companyid.
    find company using companyid.

            create present.
            present.companyid = company.companyid.
            update division contactfirst contactlast.

/* This section of code checks whether or not the presentation contact is
already in the contact table.  If he/she is, the program skips to the portion
where presentation specifics are added.  If he/she is not, the program procedes
to the contact to the contact table and allows the user to enter additional
contact information into the contact table. */

            find contact where
            contact.division = present.division and  contact.firstname =
            present.contactfirst and contact.lastname = present.contactlast.
            If not available contact then do:
                create contact.
                contact.companyid = present.companyid.
                contact.division = present.division.
                contact.firstname = present.contactfirst.
                contact.lastname = present.contactlast.
                update contact.street contact.city contact.state contact.zipcode
                contact.phone.
            End.
                present.contactphone = contact.phone.

                    update present.pdate present.ptime present.location
                    present.av present.food present.p-s-b.

/* This bottommost repeat loop allows the user to enter information for more
than one speaker for a particular company presentation. */

                         repeat:
                         {presres.f}
                             create contact.
                             contact.companyid = company.companyid.
                             update firstname lastname contacttitle
                             contact.division.
                         End.

End.
```

INFORMIX PRESENTATION RESERVATION PROGRAM

```
{Program: presresp.4gl
Database: recruit
Description: This program is designed to be the primary data entry screen for
inputting presentation reservation information. It reduces the amount of manual
input required by the CDO staff by inserting information that has been entered
only once, into more than one table, where necessary.
Form: presresf.per
Tables: contact, present }

database recruit

{This portion of the program, global variable definition, is where "program
variables" are defined.  These program variables take input from "screen
record variables" that are the actual interface with the user. The p before
each table name, selected for convenience, distinguishes the program variable
from the screen record and database values.  After the user enters input to
the screen (and into the screen record variable), it must be moved into a
program variable and then from the program variable it can then be inserted
into the actual database.}

globals
        define

        pcontact record
                companyid like contact.companyid,
                firstname like contact.firstname,
                lastname like contact.lastname,
                contacttitle like contact.contacttitle,
                street1 like contact.street1,
                street2 like contact.street2,
                city like contact.city,
                state like contact.state,
                zipcode like contact.zipcode,
                countrycode like contact.countrycode,
                phone like contact.phone,
                ext like contact.ext
                end record,

        ppresent record
                presdivision like present.presdivision,
                pdate like present.pdate,
                ptime like present.ptime,
                location like present.location,
                p_s_b like present.p_s_b,
                av like present.av,
                food like present.food
                end record,

{The presresf.per form does not have data entry fields for speaker information
because the screen can only hold 20 lines of a form and informix doesn't allow
```

```
multiple screen data entry forms.  By defining a program record for speakers,
and using it to allow users to enter speaker information, we can get around the
multiple screen form limitation.}

        pspeakers record
                speakerfirst like contact.firstname,
                speakerlast like contact.lastname,
                speakertitle like contact.contacttitle,
                speakerdiv like contact.division
                end record,

        answer char(1)

end globals

{The main program establishes the looping that allows the user to enter
reservations for many companies, without having to leave the program.}

Main

        open form presresf from "presresf"
        display form presresf
        let answer = "y"
        while answer = "y"

                call enterpresres()

                        prompt "Do  you want to enter another reservation (y/n)?
" for answer

        end while

        Message "Program Ended."
        Sleep 2
        Clear screen

End Main

{The enterpresres function (short for enter presentation reservation) is
the piece of the presresp.4gl program that does most of the work. It is in this
function that the two tables are updated for new reservation information.}

Function enterpresres()
        define addspeaker char(1)
        clear form


{This portion of the enterpresres function accepts contact information input
from the user and checks to see if the contact is already in the file. If
she is, then the program skips to the presentation data entry portion
of the presentation reservation form. If no contact exists with this companyid
```

```
and first and last name, the program prompts for additional contact information
entry.  This portion also records contact information (name, phone) in the
presentation table automatically. And lastly it automatically updates the
presentcoord field in contact to be "y", indicating that this contact is a
presentation coordinator. If this contact is new to the contact table, the
current date is inserted as the value for contactdate.}

        input by name pcontact.companyid thru pcontact.lastname
        select contacttitle, street1, street2, city, state, zipcode,
countrycode, phone, ext into pcontact.contacttitle,
pcontact.street1, pcontact.street2, pcontact.city, pcontact.state,
pcontact.zipcode, pcontact.countrycode, pcontact.phone, pcontact.ext
from contact where companyid = pcontact.companyid and firstname =
pcontact.firstname and lastname = pcontact.lastname
        if status = 0 then
        update contact set presentcoord ="y" where companyid =
                pcontact.companyid and firstname = pcontact.firstname and
                lastname = pcontact.lastname
        message "This contact is already in contact file."
        sleep 3
        message "Proceed to enter presentation specifics."
        sleep 3
        message ""
        end if

        if status = notfound then
        input by name pcontact.contacttitle thru pcontact.ext
        insert into contact (companyid, firstname, lastname,contacttitle,
        street1, street2, city, state, zipcode, countrycode, phone, ext,
        presentcoord, contactdate) values
        (pcontact.companyid,
        pcontact.firstname, pcontact.lastname, pcontact.contacttitle,
        pcontact.street1, pcontact.street2, pcontact.city, pcontact.state,
        pcontact.zipcode, pcontact.countrycode, pcontact.phone, pcontact.ext,
        "y", TODAY)
        message "Contact Added."
        message "Proceed to enter presentation specifics."
        sleep 3
        message ""
        end if

{This section of enterpresres inserts the presentation specifics entered by
the user into the present table.}

        input by name ppresent.presdivision thru ppresent.food
        insert into present (companyid, presdivision, pdate, ptime, location,
                p_s_b, av, food) values (pcontact.companyid,
                ppresent.presdivision thru ppresent.food)

        message "Presentation Added."
        Sleep 3
        message ""

{This portion of enterpresres allows for multiple speakers to be input for
one presentation.}

        prompt "Do you want to enter speaker information at this time (y/n)?"
                for addspeaker

        while addspeaker = "y"

        prompt "Enter speaker's first name: " for pspeakers.speakerfirst
        prompt "Enter speaker's last name: " for pspeakers.speakerlast
        prompt "Enter speaker's title: " for pspeakers.speakertitle
        prompt "Enter speaker's division: " for pspeakers.speakerdiv

        select * from contact where companyid = pcontact.companyid and
        firstname = pcontact.firstname and lastname = pcontact.lastname
        if status = 0 then
        update contact set presentcoord ="y" where companyid =
                pcontact.companyid and firstname = pspeakers.speakerfirst and
                lastname = pspeakers.speakerlast
        end if

        if status = notfound then
        insert into contact (companyid, firstname, lastname,contacttitle,
        division, presentcoord, contactdate) values
        (pcontact.companyid,
        pspeakers.speakerfirst, pspeakers.speakerlast, pspeakers.speakertitle,
        pspeakers.speakerdiv, "y", TODAY)

        end if

        prompt "Do you want to enter another speaker (y/n)?" for addspeaker

        end while

        message "Speakers Added."
        sleep 2
        message ""

end function
```

RBASE PRESENTATION RESERVATION INPUT FORM

```
Press [ESC] for the menu
àééééééééééééééééééééPresentation Reservation Input Formééééééééééééééééééééééééé£
¤                                                                               ¤
¤                                                                               ¤
¤  Company ID:                                                                  ¤
¤  Company Name:                                                                ¤
¤  Presenting Division:                                                         ¤
¤  Contact First Name:                                                          ¤
¤  Contact Last Name:                                                           ¤
¤                                                                               ¤
¤  Presentation Specifics:                                                      ¤
¤  ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ                                                      ¤
¤  Date:                                                                        ¤
¤  Time:                                                                        ¤
¤  Location:                                                                    ¤
¤  Audiovisual:                                                                 ¤
¤  Catering Information:                                                        ¤
¤  Permanent/Summer/Both:                                                       ¤
¤                                                                               ¤
¤                                                                               ¤
àéééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééééé¥

[ESC] Done    [F2] Clear field    [Shift-F2] Clear to end    [Shift-F10] More
Form: presres    Table: company    Field: compid    Page: 1
```

APPENDIX F
ADDS DATABASE FILES AND TABLES

ADDS DATABASE FILES AND TABLES

Aug 23 09:41 1989 entire.txt Page 1

****

CENTRAL ALUMNI DATABASE FILES AND TABLES

****

The following is a list of files and tables that currently exist in the central alumni database, ADDS - Alumni/Development/Donor/Schools. Each file or table is followed by a brief description and is annotated with an * if it contains information relevant for the Alumni and Recruiting Databases (alias the "Database Project" or DB for short) of the Placement Assistant System project. An "N" indicates the file/table is not relevant; and a "P" means that the file/table contains potentially relevant information.

Files

Code	Title	Description
*	Master	A comprehensive master record containing the majority of information about an alumnus. Information such as address, company, position, etc., is available as well as more sensitive information such as donations.
N	Activity	This is a small file which highlights the current interests/activities of alumni. The range of activities is not relevant for the DB project.
N	Club	This is a small file with information on current dues paying members of MIT clubs. Clubs are mainly alumni clubs, the majority of which are by geographic region. May be useful for Sloan Alumni coordinator but not for career development (DB project) purposes, because regardless of whether or not an alumni is active in alumni clubs, students may want to contact him/her .
*	Companies	A file with limited but useful information, such as sic code and company number.
N	History	Historical information on alumni, such as prior business, prior address, parent information.
P	Officer	Records of volunteer activity of alumni and other friends of MIT. Has some club names that might be of interest to students performing job search. We are in the process of determining the currency of the file.
N	Past-Address	Past addresses of alumni.

N	Relation	Records which establish a relationship between entities on the ADDS database. Concerns designation of donation credit.
N	Schooling	In depth information on the schooling of alumni, including course, degree, dates. We don't need because the Master File has all the information we need concerning schooling.
N	Special-Const	Special constituency information for alumni association purposes.
P	Student-Activity	Information on the activities students participated in while they were at school. Mainly pertains to sports. We are in the process of determining whether the activities in this file could be extended to cover Sloan club activity, in which case this information would be relevant.
N	Tables	A file containing information on all of the tables available in the existing alumni database.
P	Zipfile	A file of zip-codes and their relation to various types of information such as state, region, alumni club section, etc. Due to the large size of this file, we may only want hardcopy of this file or perhaps a smaller version of this file. We are working on determining what best meets our needs.
N	Accounting	Accounting information.
N	Acct-Security	Information on account security, such as last update, last user.
N	Actions	Actions taken concerning solicitations for funding.
N	Alumni-History	Concerns alumni donations.
N	Biography	Shows what other systems at MIT have information on individual alumni.
N	Budget-Perc	Budget Percentage. Accounting information.
N	Chart	Accounting information concerning donations/funding.
N	Clearance	Information on clearance for funding of projects.
N	Committment	Travel voucher information.
N	Companies-T	Duplicate information of Companies file, used for

		test purposes.
N	Current-Gifts	Information on individual gifts from donors.
N	Dictionary	Information regarding fields, files, maintenance, etc. for programmer use.
N	Donor-Master	File of Master Records of donors.
P	Enterprise-Forum	Enterprise Forum membership information. Groups of corporations under MIT sponsorship. Alumni Association related. We are determining whether or not this information would be relevant. Currently we are not authorized for this information.
N	Expectancies	Information on donation expectancies.
N	Fund-table	Information on funds.
N	Gift	Information on gifts from companies/alumni.
N	Gift-History	Historical information on gift giving.
N	History-Gifts	Historical information on gift giving.
N	Major-Screeners	Information on funding prospects.
N	Major-Suspects	Information on funding prospects.
N	Masteral	For programming purposes. Used for screen access.
N	Old-Master	Duplicate version of master file information. Used for year end test purposes.
N	Pledge-Payment	Information on payment of pledges.
N	Pledges	Information on pledges.
N	Prod-Schedule	Schedule for the production of reports from Alumni database.
N	Prospects	Extensive information on funding prospects.
N	RD-Accounting	Accounting information.
N	RD-Acct-Sec	Accounting information.
N	RD-Budget-Perc	Accounting/budget information.
N	RD-Commintment	Voucher information.

N	Registrant	Registration and payment information for alumni functions.
N	Registrar	Information from Registrar's office on new graduates. Since we anticipate getting scheduled updates of alumni information, we will not need this file.
N	Reunion-Gifts	Reunion gift giving information.
N	SCF-Stuff	Miscellaneous, unidentified information. Only has 4 fields that are not descriptive but do not appear to be important for our purposes.
N	Schedule	Schedules for alumni association activities.
N	Screening	Prospect Control System. Screening of prospects as evaluated by potential giving level.
N	Security-Table	Information on users of alumni database and their security designations.
N	SMS	Solicitation Management System. Records track prospects for reunion giving.
N	Sustain-Fellows	Prospect Control System, a subsystem of ADDS. Donor information.
N	TDAY	Alumni reception information.
N	Tech-Review	Gift giving information and subscriptions to MIT's Tech-Review.
N	TESTPLD	Donor information.
N	Vol-Con	Volunteer Control. Records of volunteer activity of alumni and other friends of MIT.

Tables

Code	Table #	Title/Description
N	000	List of tables.
N	010	Current Fiscal Year. A one line table consisting of the current fiscal year.
*	100	MIT Courses. Codes for each of the various courses available at MIT.

		For example, Management = 15.
N	101	New Sequence Numbers. New sequence numbers, used to uniquely identify alumni.
N	104	MIT Employee Code. Codes describing type of employment for MIT employees.
P	105	Career Focus Codes. Codes describing the career focus of alumni. Similar to industry and job function. Redundancy of information in this table as compared to others would need to be determined.
N	106	Activities-Interests. Codes covering activities of alumni. Not extensive enough to be relevant for DB purposes.
*	110	Prefixes. Prefixes for addressing individuals. May be useful for students trying to contact alumni.
*	111	Suffixes. Suffixes such as jr., phd, that also might be useful when contacting alumni.
N	112	Nickname Users. Nickname table containing two entries.
N	114	Donor types. Donor information is not relevant to DB project.
N	115	MIT Affiliation Codes. Covers affiliations other than alumni.
N	116	Majors/Minors. Information is basically covered by table 100. In addition, the only information on graduate students available from this table is year of graduation.
N	117	Fund Program Codes. Codes describing fund programs under which donations are categorized.
N	118	Fund Job Codes. Job codes for alumni solicitation efforts.
N	119	Biographic codes. Describes source of funding prospect information.

N	120	Achievement Codes. Noteworthy public achievements of alumni, e.g. nobel laureate.
N	121	Identification codes. For various MIT departments.
N	122	Living Group codes. Codes for student housing arrangements.
*	123	Country of origin. This information will be useful during the job search process, particularly for those alumni interested in identifying alumni with similar backgrounds and job search strategies.
N	125	Address Use. Restrictions on the use of alumni addresses. Pertains mainly to solicitation effort.
*	126	Address preference. Home or business. Indicates preferred address to be contacted at.
N	127	Address Source codes. Codes for sources of alumni addresses.
*	128	Business Position number. Codes are categorized by industry and then title.
*	130	Degree abbreviations. Indicate the type of degree obtained. Includes codes for non-degree studies.
N	131	Degree codes. These codes are actually for groupings of degrees. We will not need because this information will be covered more effectively by our selection of other degree fields.
N	132	School Codes. Codes indicating college/university attended.
N	133	City & Town Area Regions. Geographic locations of address. These codes were used prior to the existence of zip codes. We can cover geographic region by zip code and therefore we don't need this information.
N	134	999 Sequence Series.

		Sequence numbers used by the Fund staff for solicitation purposes.
N	135	City & Town Areas. First two positions of geographic location. Same as for table 133.
N	136	Tech Review Codes. Codes for complementary subscriptions to Tech Review.
N	137	Relationship codes. For designating donation credit.
N	139	Status Within Relationship. Relation codes within relationships establishing donor identity.
N	140	Club File names. Codes for alumni club names.
N	141	Club File members codes. Describes type of member the alumnus is in a club.
N	142	Club File Officer codes. Codes for officers of alumni association clubs.
N	145	Tech Review Initials. To distinguish those who assigned a complementary subscription to an alumnus.
N	146	Tech Review Category Codes. Categories for Tech Review gift giving.
N	147	Tech Review Payment Category codes. Payment categories for Tech Review gift giving.
P	151	Officer File level. Codes for various alumni association special interest groups. Code indicates what level in the Officer file the record is in. We are in the process of determining the currency of this information and whether or not it is relevant.
N	152	Officer file group. Contains extensive list of codes indicating membership group, e.g. year, city, fraternity, etc.. Code explains what group within the level the record is in. At this level the information would not be too useful for job search purposes.

N	153	Officer file position. Indicates the various officer positions held in alumni association/clubs. Again, this level is not useful for job search purposes.
P	155	Zip by state. Shows range of first three digits of zip code for all states. We may not need a table with this information on-line if we can sort with a
N	156	Regional Director codes. Codes for regional directors of alumni association.
N	157	State Codes. Due to the format in which we will be receiving addresses from ADDS, we will not be able to isolate state code. Therefore, this table will not provide us with any additional capabilities that zip codes don't have.
N	158	Country Codes. Same as for table 157.
N	160	Activity File codes. Codes for alumni association activities, such as reunions.
P	161	Student Activity Codes. Pertains to extra-curricular activities of alumni when they were students. Current list does not have any information relevant for job search. We are investigating the possibility of expanding the list to include Sloan clubs such as Finance, New Ventures, etc.
N	162	Special Constituency Codes. Covers special alumni association/club events.
N	163	MIT Schools & Departments. Alpha codes for MIT Schools and Departments. Table 100 will be used to identify specific MIT schools, by course number.
N	199	Long Off Level and Group. A long list of alumni club levels and groups.
N	200	Gift/Pledge source. Codes for the sources of Gifts/Pledges.
N	201	Payment Vehicle. Codes for mode of payment of gifts.

N	202	Acknowledgers. Indicates office that will acknowledge receipt of gift.
N	203	Reminder Codes. Indicates what MIT office should remind individuals about pledges.
N	204	Pledge Payment Frequency. Codes for frequency of pledge payments.
N	205	Carry-Forward codes. A one item table indicating "carry-forward".
N	206	Fund/Donor Purposes. Codes indicating area to which donor has designated a pledge.
N	207	Match Form Status. How the matching gift comes in and how it will be acknowledged.
N	208	Payment types.
N	211	Alumni Fund Projects. Codes for alumni fund projects. For system use only.
N	220	Matching Fund Table. One item table indicating matching fund code. Used for programming purposes.
N	230	V.P. Start dates. Start dates for each Visit Program (V.P.) District.
N	231	V.P.End dates. Ending date codes for each Visit Program District.
N	234	V.P. District Chairperson. Sequence numbers for Visit Program District Chairpersons.
N	240	V.P. Results. Results of Visit Program District solicitation efforts.
N	245	V.P. district codes. Names of Visit Program districts.
N	250	V.P. upgrade codes. Indicates changes in giving patterns.

```
N       255             V.P. Area Directors.
                        Codes for V.P. area directors.

N       260             8283 Status codes.

N       261             Restrictions.
                        Codes for restrictions. Concerns gifts of securities.

N       262             Brokers.
                        Codes for securities brokers who sell gifts of
                        securities.

N       300             Alumni Designation Codes.
                        Five-digit account number towhich a pledge or gift
                        is designated.

N       301             Fund Designation Category.
                        Categories for fund designations.

N       302             Output Designations.
                        List is more extensive than table 300. Used for
                        programming purposes.

N       305             Telethon codes.
                        Codes for telethon solicitations.
N       310             FF's not in CF.

                        A one item table indicating New York Center. Used
                        for programming purposes.

N       500             Events Program codes.
                        Alumni events program codes. Used by the Registrant
                        program.

N       501             Schedule Fee Unit.
                        How the alumni events are charged, e.g. per person,
                        per couple, etc.

N       502             Status codes.
                        Status codes for mailings.

N       503             Dorm codes.
                        Where people are staying during events.

N       504             Special Housing codes.
                        Special housing requirements for alumni during
                        alumni events.

N       505             Event run time.
```

		Alumni event run time codes.
N	515	Registration Codes. For alumni events.
N	520	E.F. City Codes. A two item table of Enterprise Forum City codes.
N	601	Prospect Control System (PCS) Prospect Managers. Initials of staff members who have primary responsibility for prospects.
N	602	Susfells Threshholds. Threshold levels for sustaining fellows.
N	603	PCS Record status.
N	604	PCS Staff codes.
N	605	PCS Research ID's.
N	606	PCS DIMS user ID's.
N	607	PCS Research ID's.
N	608	PCS Sustaining Fellows ID's.
N	609	PCS Expectancy ID's.
N	61A	PCS Online/Batch Reports codes.
N	61B	PCS Event Responses.
N	61C	PCS Expectancy Affiliations.
N	61D	PCS Expectancy Source codes.
N	61E	PCS Solicitation Result Codes.
N	61F	PCS Ask Purpose codes. Indicates designations for gifts.
N	61G	PCS Income Codes. Gives income ranges.
N	61H	PCS Marital Status codes.
N	61J	PCS ASK School/Dept codes.
N	610	PCS prospect status codes.

		Cultivation/Solicitation status of prospect on PCS.
N	611	PCS inclination codes. Indicates probability that donation will be made.
N	612	PCS Sustaining Fellows membership types.
N	613	PCS Geo Staff codes.
N	614	PCS PM-Support ID's.
N	615	PCS Research types.
N	616	PCS Clearance status.
N	617	PCS Clearance results.
N	618	PCS Clearance ID's.
N	619	PCS Funding Organization codes.
N	620	PCS Clearance Staff codes.
N	621	PCS Clearance months off. A one item table showing PCS Clearance Months Off.
N	622	PCS data screen codes.
N	623	PCS Corporation/Foundation ID's.
N	624	PCS Report request codes.
N	625	PCS Camp Committee codes.
N	630 - 650	Campaign For MIT.
N	630	Volunteer Control (Vol-Con) program codes. Campaign programs people may be volunteers in.
N	631	Vol-Con position codes. Volunteer position people may hold in the program for the campaign.
N	632	Vol-Con Status Code. Status of the volunteer in the program of the campaign.
N	633	Campaign Total Date.
N	634	Event Type.

N	635	Event Name.
N	636	Event Date.
N	637	Event Reply Status.
N	638	Event Registration Number.
N	639	Area Market Corr. Table.
N	640	Date Parm.
*	700	SIC codes. We should determine whether these codes are specific enough for our needs.
N	900	Frequency codes. Codes indicating how often a job is run in the production schedule.
N	901	Schedule user codes. Codes for alumni association user groups requesting production schedule output.
N	905	Table of months.
N	906	Uppercase Months. Months in all uppercase letters.
N	950	Current Accounting Yr/Mo. Year/Month of accounting data for programming.
N	951	Committment Records types. Payments & vouchers.
N	952	Accounting Document Sequence Numbers. Sequence numbers for committment accounting documents.

APPENDIX G
ADDS MASTER FILE REVIEW

ADDS MASTER FILE REVIEW

Aug 23 09:41 1989 mstrfile.txt Page 1

MASTER FILE REVIEW

The Master File of the central alumni database consists of approximately 230 fields, only a selection of which are relevant for the CIS/TK Alumni Database project. This list has been compiled based on the judgement of the DB team and consists only of those fields that may be potentially useful for the Alumni DB project.

Each of these fields was then further reviewed to determine relevancy. Any fields that were determined to be important for the alumni database project can be found in the document "Final Selection". This document contains miscellaneous notes or discussions with Emily Magazzu, Sloan's ADDS coordinator, which led to the inclusion or deletion of the field in the final selection.

Field Code	Field Name	Field Description
AV	Achievement-Codes	Noteworthy public achievements of alumnus.
AT	Achievement-Dates	Calendar dates of achievement.
AE	Achievement-Descriptions	Descriptions of achievements.

The use of the above fields was eliminated because the list of achievements they reference is not very extensive nor directly relevant to careers in business.

HM	Alt-Addr-Prf-Test-Hm	Address at which individual prefers to receive mail. May or may not be useful. May relate only to solicitation efforts.
HQ	Alternate-Home-Address	Alternate address (summer/winter). At prespecified date the info in this field is exchanged with home address field.? Fields HF and HG give the from and to dates for this HQ field. May be useful for students because ensures letters get sent to the current address.
HK	Alternate-Home-	Telephone number at alternate home address.
HL	Alt-Home-Tel-Use	Restrictions on use of alt. phone number.

These alternate fields were eliminated because the information automatically switches with the normal address/telephone (HA) when the person changes residence (as indicated by from/to date fields), in which case our refreshing of the database would indicate current address.

QQ	Business-Address	Business address of individual, consisting of 4 lines: Company name, division/street, city/st/zip, country.
QF	Bus-Address-From-Date	Date of last change. This information

```
                                 may be helpful to students in determining
                                 how long an alumnus has been at a particular
                                 location.
QH      Bus-Add-Verified-Date    Date business address last verified by alumni
                                 association.  May help students to identify
                                 the currency of the information and therefore
                                 whether or not it should be verified before
                                 sending out a cover letter.
QF was selected over QH because the verification process is not standardized.

B2      Business-Area            Location of business address.  These are
                                 geographic areas defined by the alumni
                                 association.
B3      Business-Region          Location of business address.  These are
                                 larger geographic regions defined by the
                                 alumni association.

These regions  were established prior to the advent of zipcodes.  We will be
using zipcodes to do sorting.  The only complication with zipcodes may be with
foreign zips, in which case Emily suggests we may want to use region codes. She
is going to do some thinking on this matter.

BP      Business-Position        Position held by individual.
PC      Business-PSN-Code        Position codes from table 128.


QK      Business-Telephone       Business Telephone number of individual.
This is a duplicate field which we will not need.

ZB      Business-Zipcode         Same as in city/st/zip line of QQ, but here
                                 as its own field.  We probably won't need
                                 this if we have information on regions,
                                 however its potential use should be considered.

FO      Career-Focus             General area of career focus, as distinguished
                                 from business position, BP.

This code could be useful because the current position code/description fields
are a bit deceptive: the position code is 5 digits, the first 3 representing
the industry, the last 2 the title.  Unless you know what industry the number
stands for, the title description won't in most cases describe the industry
(e.g. Director). However, Emily is not sure whether this field is being used
currently by central alumni.  If it isn't being used, the validity/currency of
the information may be questionable. Emily is checking on this.

CI      Club-Indicator           An MIT club has identified the alumnus as
                                 being a member.
This field was eliminated because the clubs refer to geographic alumni clubs
for the most part.
```

```
BA      Company-Name              First line of business address.  Same as
                                  first line in QQ.  We need to determine
                                  whether we want the business address infor-
                                  mation line by line or in one field as in
                                  QQ.

SH      Company-Name-Alias        Abbreviated or alias form of company name.
                                  Master file writeout says this is keyed
                                  off of the "LE" field, however that field
                                  is for legal name.  Might be useful for
                                  locating company information.
Emily does not feel this field would be useful to us.  There are other fields
from the company file (see FINAL SELECTION document) which will allow us to
identify companies even though the company name has been spelt differently.

BN      Company-Sequence-#        Sequence number of company named in business
                                  address of individual.

CO      Country-of-Origin         Country of origin of foreign alumnus.  From
                                  table 123.

DD      Death-Date                Date of death of individual.  We need to
                                  ensure that the Alumni DB has only living
                                  alumni, whether it be via this field or by
                                  some other means.
We don't need this field because Emily can screen out alumni that have died
from our update dump.

DP      Department-Affiliation    Dept. Affiliation of alum as MIT undergrad
                                  or most recent dept afil. of grad students.
                                  Might not be detailed enough for Sloan grads,
                                  i.e. might lump all Sloanies together.
This is not useful because other fields cover this information.

BB      Division-Name             Second line of business address, QQ. Usually
                                  division name or postal drop.
This field is not going to be used by central alumni in the very near future.
Other fields will cover this information.

PM      Dont-Mail                 Restrictions on use of address of person
                                  for sending alumni association mail.  Uses
                                  table 125.  May concern solicitation mail
                                  only and thus not a problem for cover letters.
Not useful to us because concerns solicitations.

FS      Formal-Salutation         Formal salutation for use in correspondence.
This information is covered by the prefix field, thus is unnecessary.

FE      Gender-of-Individual      Gender.  May not be necessary if have FS.
```

```
HA      Home-Address              4 line field for home address.  This must be
                                  the main home address?
AD      Home-Add-Verified-Date    Date home address last verified by alumni
                                  association.
Field AC, which is date last changed, will be used instead of AD.

A2      Home-Area                 Geographic areas defined by alumni association.
RH      Home-Area-Region          Larger geographic areas.
A3      Home-Region               Geographic regions within larger geographic
                                  areas.
We will be using zipcodes instead of these region codes.

TN      Home-Telephone            Home telephone number.
HC      Home-Telephone-Use        Restrictions on the use of home number.
                                  Primarily for solicitation purposes?
Emily is going to screen out all phone numbers with restrictions on them from
our dump, so we will not have any sensitive phone numbers in our database. We
won't therefore need this field.

ZP      Home-Zipcode              Zipcode by itself.  Same as in HA.

MD      Maiden-Name               Former name if surname has changed.  May
                                  not be relevant if only current students
                                  have access to system.  But if alumni
                                  can also use, they may want to be able to
                                  identify the business activity of former
                                  classmates.

DG      MIT-Degree-Combination    MIT Degrees or combinations of degrees held by
                                  alumni.
Duplicative information we have covered with other fields.

C1      MIT-Degree-Course-#'s     Course numbers of MIT degrees held or of
                                  studies if no degree.
Y1      MIT-Degree-Dates          Dates of receipt of degrees or of studies if
                                  no degree held.
D1      MIT-Degrees-Held          Degrees held.  How does this differ from DG?

FN      Name-First                Legal first name.
LN      Name-Last                 Legal surname.

LE      Name-Legal                Complete legal name in sequence first, middle,
                                  last.  Which of these fields is more
                                  appropriate for our purposes must be
                                  determined.
We will use the separate name fields, for sorting purposes, rather than this
combined field.

PX      Name-Mailing-Prefix       Prefix to be used for mailing purposes. Uses
```

```
                                 table 110.
SF      Name-Mailing-Suffix      Suffix for mailing purposes.  Uses table 111.

HD      Pref-Address-Indicator   Preference for receipt of mail, business or
                                 home.

HE      Pref-Address-Usage       Restrictions on preferred address usage for
                                 alumni association purposes.
Restrictions relate to solicitations, therefore we don't need this field.

RP      Pref-Area-Region
PA      Preferred-Area           Geographic area of preferred address.
PB      Preferred-Region         Geographic region of preferred address.
We will use zip versus region codes.

PZ      Preffered-Zipcode        Zipcode of preferred address as its own field.
                                 These preferred address fields are somewhat
                                 duplicative of home address, alternate address,
                                 etc..  The best form of address for the alumni
                                 DB project will have to be determined.
Emily suggests this would be a key field for the alumni database to have
because it will make querying much easier.  If you know you want to contact an
alumnus at his/her preferred address, you can sort off of this field rather
than off the preferred address indicator, home address field and business
address field.

SQ      Sequence-Number          Unique number identifying each alumni on the
                                 master file. Links records in other files to
                                 master file.

QD      W-Addr-Use               Restrictions on the use of work address,
                                 for alumni association mail. Table 125.
QC      W-Phone-Use              Restrictions on the use of work telephone
                                 number for alumni association purposes.
These restrictions relate to solicitations, thus they are not applicable.

WD      Withdrawn-Date           Date alumnus withdrew from MIT if no degree
                                 received.  We need to make sure we can tell
                                 who actually received a degree from those who
                                 did not.
We don't need this field because the degree code for non-degree individuals is
distinct from those of degree'd individuals, and we will be using degree codes.

BC      Work-Line-3              Third line of business address.  Usually
                                 street and number.
BD      Work-Line-4              Fourth line.  City/st/zipcode.
These fields will no longer be used by central alumni.

BT      Work-Phone               Business telephone number.
```

```
BX      Work-Prefer-Code        Person prefers to receive mail at business
                                address.
This field is no longer going to be used by central alumni.

TB      Work-Type               Principal type of business engaged in by
                                company.  May be more descriptive than
                                sic code.
Emily believes that this field is not useful, and that our interests will be
covered by the sic-code field.
```

BIBLIOGRAPHY

Banks, Alan, Andrea Flamburis and Laurence Kooper. "Placement Assistant System." Manuscript, M.I.T. 1988.

Baggeroer, William L. and John M. Fox. "Student Information System, Sloan School of Management, Preliminary Analysis and Design." Manuscript, M.I.T. 1974.

Barth, Richard R.. "A Microcomputer Network Design for the Sloan School Admissions Office." Manuscript, M.I.T. 1989.

Hoffer, Jeffrey A. and Fred R. McFadden. Data Base Management. Benjamin/Cummings Publishing Company, 1988.

Kooper, Laurence Stanley. "A Composite Information System for the Sloan Placement Office." Master's Thesis, M.I.T. 1988.

Lavigne, Jean Camille and Denis Henri Porges. "A Management Information and Control System For a School: Its Application to the Sloan School of Management." Master's Thesis, M.I.T. 1971.

Madnick, Stuart E. and Y. Richard Wang. "Logical Connectivity: Applications, Requirements, and an Architecture." CIS/TK Working Paper #2061-88, M.I.T. 1988.

Orlikowski, Wanda, et al. "DBMS Software Review." Manuscript, M.I.T. 1989.

Redwine, Samuel T. "The Implementation of a Management Information and Control System for the Sloan School of Management." Master's Thesis, M.I.T. 1972.

Tener, Lisa. "Organizational Issues Involved in the Development of the Placement Assistant System at the Sloan School of Management." Manuscript, M.I.T. 1988.

Treacy, Michael E., et al. "Systems Requirements Study of the Masters Program Office of the Sloan School of Management." Manuscript, M.I.T. 1986.