THESIS

DESIGN OF A FINANCIAL MANAGEMENT SYSTEM FOR THE ACADEMIC DEPARTMENTS AT THE NAVAL POSTGRADUATE SCHOOL

by

Alan E. Pires

March, 1997

Thesis Advisor: C. Thomas Wu

Approved for public release; distribution is unlimited.
<table>
<thead>
<tr>
<th>1. AGENCY USE ONLY (Leave blank)</th>
<th>2. REPORT DATE</th>
<th>3. REPORT TYPE AND DATES COVERED</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>March, 1997</td>
<td>Master's Thesis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4. TITLE AND SUBTITLE DESIGN OF A FINANCIAL MANAGEMENT SYSTEM FOR THE ACADEMIC DEPARTMENTS AT THE NAVAL POSTGRADUATE SCHOOL</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>5. FUNDING NUMBERS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>6. AUTHOR(S) Alan E. Pires</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naval Postgraduate School</td>
</tr>
<tr>
<td>Monterey CA 93943-5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8. PERFORMING ORGANIZATION REPORT NUMBER</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>10. SPONSORING/MONITORING AGENCY REPORT NUMBER</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>12a. DISTRIBUTION/AVAILABILITY STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approved for public release; distribution is unlimited.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12b. DISTRIBUTION CODE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>13. ABSTRACT (maximum 200 words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>This thesis examines the requirements and design of a financial management system for the academic departments at the Naval Postgraduate School. Existing systems are difficult to maintain and/or provide out-of-date information. A system is needed that is easy to use, easy to maintain, and provides current account status information so that the academic departments can make intelligent financial decisions.</td>
</tr>
</tbody>
</table>

We examined existing methods and tools for designing and building client/server applications. After comparing the traditional waterfall approach to the rapid prototyping approach, we elected to use rapid prototyping in order to develop the system quickly and to help the users determine their own requirements. We decided to use the Powersoft Portfolio tool set from Powersoft Corporation because it is scalable, transportable, affordable, and compliant with the Open Database Connectivity standard. |

The result of this thesis is a prototype financial management system that users have found easy to use and maintain. The system provides summary and detail information on departmental financial accounts, to include balances and expenditures in the funding categories of faculty and support labor, equipment, travel, and contracts. |

<table>
<thead>
<tr>
<th>14. SUBJECT TERMS Accounting System, Database, Client/Server Application, Rapid Prototyping,</th>
</tr>
</thead>
</table>

| 15. NUMBER OF PAGES | 177 |

<table>
<thead>
<tr>
<th>16. PRICE CODE</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>17. SECURITY CLASSIFICATION OF REPORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>18. SECURITY CLASSIFICATION OF THIS PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>19. SECURITY CLASSIFICATION OF ABSTRACT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unclassified</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>20. LIMITATION OF ABSTRACT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>UL</th>
</tr>
</thead>
</table>

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 259-18 298-102
Approved for public release; distribution is unlimited.

DESIGN OF A FINANCIAL MANAGEMENT SYSTEM FOR THE ACADEMIC DEPARTMENTS AT THE NAVAL POSTGRADUATE SCHOOL

Alan E. Pires
B.S., United States Military Academy, 1980

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

Author: Alan E. Pires

Approved by: C. Thomas Wu, Thesis Advisor
John A. Daley, Second Reader
Ted Lewis, Chairman
Department of Computer Science
ABSTRACT

This thesis examines the requirements and design of a financial management system for the academic departments at the Naval Postgraduate School. Existing systems are difficult to maintain and/or provide out-of-date information. A system is needed that is easy to use, easy to maintain, and provides current account status information so that the academic departments can make intelligent financial decisions.

We examined existing methods and tools for designing and building client/server applications. After comparing the traditional waterfall approach to the rapid prototyping approach, we elected to use rapid prototyping in order to develop the system quickly and to help the users determine their own requirements. We decided to use the Powersoft Portfolio tool set from Powersoft Corporation because it is scalable, transportable, affordable, and compliant with the Open Database Connectivity standard.

The result of this thesis is a prototype financial management system that users have found easy to use and maintain. The system provides summary and detail information on departmental financial accounts, to include balances and expenditures in the funding categories of faculty and support labor, equipment, travel, and contracts.
# TABLE OF CONTENTS

## I. INTRODUCTION

A. BACKGROUND .............................................. 1

B. REVIEW OF EXISTING SYSTEMS ............................ 1
   1. Operations Research Department System .......... 1
   2. Computer Science Department System .......... 2

## II. SYSTEM REQUIREMENTS AND DESIGN .................. 5

A. PROJECT SCHEDULE ........................................ 5

B. SYSTEM REQUIREMENTS ..................................... 5
   1. General Requirements ................................. 5
   2. Read Access (Queries) ............................... 6
   3. Write Access (Updates) ............................. 6
   4. Report Generation .................................. 7

C. SELECTION OF SOFTWARE TOOLS ......................... 7

D. DATABASE DESIGN .......................................... 8
   1. The Enhanced Entity Relationship Diagram ........ 8
   2. The Physical Data Model ............................ 8

## III. FINANCIAL MANAGEMENT SYSTEM ..................... 13

A. CLIENT/SERVER PROCESSING DECISION ............... 13
1. Database (Back-end) Processing .............................. 13
2. Application (Front-end) Processing .......................... 15

B. APPLICATION DEVELOPMENT ............................... 16
   1. Background .................................................. 16
   2. Implementation ............................................. 17
      a. Financial Management System Modules ............... 17
      b. Staff Module Components .............................. 18
      c. Rapid Application Development ...................... 20

C. APPLICATION DEPLOYMENT .................................. 21

IV. ANALYSIS ....................................................... 29
   A. TOOLS ......................................................... 29
      1. Database Modeling ....................................... 29
         a. Strong Points ......................................... 29
         b. Weak Points ............................................ 31
      2. Application Development ................................. 32
         a. Strong Points ......................................... 32
         b. Weak Points ............................................ 33

B. DATABASE SERVER ............................................. 34
   1. Strong Points .............................................. 35
   2. Weak Points ................................................. 35

C. PROTOTYPE .................................................... 36
ACKNOWLEDGEMENTS

I would like to thank Dr. C. Thomas Wu for his continual support, guidance, patience, and sense of humor during this project. I would also like to thank LCDR John A. Daley for his help and support in editing.

I also wish to thank the Chairman of the Operations Research Department, CAPT Frank Petho, and my supervisor, Alan Jones, for their support. Without their support, this work would not have been possible. I also wish to express my gratitude to the staff of the Operations Research Department who tested the system and provided invaluable input for its design and development and to my co-workers who provided encouragement and support.

I am very grateful to my parents for their support and faith in me. Their positive attitude and encouragement have always been a source of inspiration. Most importantly, I am indebted to my wife, Kristine, and my daughters, Jessica and Monica, for their constant love, patience, and understanding during the work involved in this thesis project.
I. INTRODUCTION

A. BACKGROUND

The academic departments of the Naval Postgraduate School need a method to provide current status information for their numerous financial accounts. Reports from the Comptroller are quarterly and are frequently out-of-date when received. Without up-to-date information, the departments cannot make intelligent financial decisions. Although solutions to this problem have been developed, they do not provide a complete or efficient solution to the problem. This thesis determines the requirements and design for a financial management system for the academic departments.

B. REVIEW OF EXISTING SYSTEMS

1. Operations Research Department System

The Operations Research Department has a system that was developed using Borland Paradox for DOS. It was loosely based on a system that had been developed for the Administrative Science Department (now known as the Systems Management Department) using dBase IV [Ref. 1, 2, 3, 4, 5]. Neither the Administrative Science Department’s database nor the Operations Research Department’s database was designed using proper database design techniques, i.e., no data modeling was done such as through the use of Entity-Relationship (ER) diagrams or Enhanced Entity-Relationship (EER) diagrams [Ref. 6]. The Administrative Science Department’s system was not easy to maintain and not easily transportable to other departments.
The Operations Research Department's system, named the "Paradox-based Financial Management Information System (PFMIS), allowed the inputting of account, labor, equipment, and travel information but only calculated the balance of accounts for the labor category. The version of Paradox used does not support storage of embedded code, such as Structured Query Language (SQL) code, in the database. Instead, scripts written in the "Paradox Application Language" have to be manually executed to perform calculations such as those needed to determine the balance of an account. More sophisticated databases allow embedded code, known as triggers and stored procedures, which can cause calculations or other actions to happen automatically upon insertion, modification, or deletion of data in the database.

2. Computer Science Department System

The Computer Science Department system is based on the Microsoft Excel spreadsheet. As such, it does not have many of the important features of a database system. For example, it cannot check that the user is inputting valid data, it cannot provide various levels of security to the data such as allowing some users read-only access and other users read-write access, it cannot provide transaction tracking and the ability to cancel transactions, it cannot provide the necessary protection to data that would allow simultaneous inputting of data by multiple users, and it cannot easily provide on-line access to individual professors of the status of their accounts. To provide account status information to the professors, the individual who inputs the data into Excel runs a program that converts a spreadsheet containing summary status information into a HyperText Markup Language (HTML) document. The HTML document is then posted on a World Wide Web page where the
professor can view it. A database system, on the other hand, would allow the professors to access the database at any time to view the status of an account or the database system could be set to automatically update a Web page whenever new data was entered. In short, the Computer Science Department is attempting to solve a database problem using a spreadsheet.

This thesis uses an approach that will use modern design techniques to provide a robust financial accounting system that is easy to use and maintain.
II. SYSTEM REQUIREMENTS AND DESIGN

A. PROJECT SCHEDULE

The first step in the project was to develop a project schedule. A copy of the schedule is given in Appendix A. The project was divided into three main phases: a design phase, a development phase, and a test/debug phase. Each of these phases consisted of a variety of tasks. It was determined that many of the tasks could be done in parallel. To begin the project, system requirements were determined and software tools were selected. The Operations Research Department was selected as the test department for the project.

B. SYSTEM REQUIREMENTS

System requirements were developed by studying the existing system in the Operations Research Department and by conducting interviews with key personnel in that department to determine what tasks they needed to perform [Ref. 7]. The system requirements were determined to be as follows.

1. General Requirements

- Track the department’s financial accounts. All type of accounts need to be tracked, e.g., Reimbursable Research (RR), Direct Research (DR), Direct Teach (DT), etc.

- Track the total dollar amount of each account, as well as the subcategories that the funds are broken out to, i.e., faculty labor, support labor, travel, OPTAR, and contracts.
• Data must be exportable, i.e., the user\(^1\) must be able to bring data from the system into a spreadsheet or other program for manipulation.

• Security down to the “field” level so that only authorized users can read and/or write fields, records, and tables.

• The “front end” of the system must be compatible with Windows 3.1x, Windows 95, Mac OS, and common variations of the Unix operating system, such as Sun Solaris.

2. Read Access (Queries)

• Determine the balance in an account broken out into the following subcategories: faculty labor, support labor, travel, OPTAR, and contracts.

• List all charges against an account and see which charges are obligations (funds committed but not spent) versus actual expenditures.

3. Write Access (Updates)

• Write access (updates) must be limited to authorized users in the department to help ensure the accuracy of the database.

• Authorized users should be able to enter information about initial funds in an account and charges against accounts. Charges against accounts will be in the subcategories of faculty labor, support labor, travel, OPTAR, and contracts. If possible, this information should come from other systems, e.g., SACONS (Standard Automated Contracting System), to avoid duplicate entry of data.

\(^1\)For these requirements, the term “user” refers to any authorized user of the system, e.g., a staff member who inputs data, the department chairman, and faculty members who are the Principle Investigators for accounts.
4. **Report Generation**

- The user should be able to produce the faculty and staff labor certification reports for each pay period. These reports show the number of hours of labor each week charged to specific accounts for each employee. The system should include some calendar functions so that it will automatically account for holidays, etc.

- The system must have the ability to easily produce custom reports such as lists of accounts and employees, lists of expenditures on accounts, and so on.

C. **SELECTION OF SOFTWARE TOOLS**

At the same time that the requirements were being developed, software tools to aid in the design of the database and the development of the application were examined. The desired features of the tools were:

- Affordable

- Scalable

- An established product. By purchasing an established product, it would more likely have support available through a variety of sources to include user groups and third-party books.

- Ease of use. The tools needed to be relatively easy to learn to use.

- Require a minimum of coding. By minimizing coding the resulting system would be easier to maintain.

- Transportable. In other words, able to implement on an IBM-compatible PC, Macintosh, or Unix-based system.

- Compliant with the ODBC (Open Database Connectivity) standard developed by Microsoft. Compliance with this standard would allow the application to interface with any ODBC compliant database such as Oracle or Sybase SQL Server. This would prevent the design from being locked in on one product/vendor for implementation.
The products that were considered included: *Powersoft Portfolio*, *Symantec Enterprise Developer*, *Oracle Database Server* and *Oracle Power Objects*, and *Borland Delphi*. The decision was made to select *Powersoft Portfolio* because it provided a database design tool (*S-Designer AppModeler*, formerly, *StarDesignor*), an application development tool (*PowerBuilder Desktop*), and a database server (*Sybase SQL Anywhere*, formerly, *Watcom SQL Server*), it met all of the desired features, and it was the most affordable.

D. DATABASE DESIGN

1. The Enhanced Entity Relationship Diagram

After the system requirements had been determined, the database was designed using an Enhanced Entity-Relationship (EER) diagram [Ref. 6]. The EER diagram, minus the attributes, is shown in Figure 1. The attributes for each entity and relationship are shown in Tables 1 and 2 respectively. The EER diagram was developed based on the system requirements, interviews with users of the system, and desired reports (output) from the system. The completed EER diagram was used to determine what tables to create, what attributes to have in each table, and what relationship existed between tables [Ref. 6].

2. The Physical Data Model

The database design tool included with *Powersoft Portfolio*, *S-Designer AppModeler*, could not be used to create EER diagrams. Instead, the user graphically creates database tables, enters the attributes for each table, and then creates the relationships between tables. This is what *S-Designer AppModeler* refers to as the “physical data model.” Once the physical data model is complete, the user can generate any number of ODBC compliant databases, such as Oracle, Sybase SQL Anywhere, Microsoft Access, Borland Paradox, etc.
For this project, once the physical data model had been created from the EER diagram, the physical data model was used as the design for the database. In other words, as the design was changed over time, the physical data model was updated, not the EER diagram. This was done for practical reasons. Changes could easily be made to the physical data model using S-Designtor AppModeler. No tool was available to easily change the EER diagram. After making changes to the physical data model, the database could be modified automatically using S-Designtor AppModeler to generate and execute the SQL code. Making changes to the EER diagram could not, of course, be used to change the database automatically since S-Designtor AppModeler could not work with the EER diagram. The physical data model is shown in Figure 2.

The user of S-Designtor AppModeler does have to provide some of the intelligence for modifying the database, i.e., S-Designtor AppModeler cannot successfully implement all modifications to the database. If multiple changes need to be made to the database, the user might have to enter one change at a time to the physical data model and have S-Designtor AppModeler modify the database after each change to the physical data model in order to have the changes implemented properly. This is not always the case. It depends on what changes are being made to the database. For example, if non-key attributes (fields) are being added to some of the tables, this could be done all at once. If, however, a key attribute was being added or removed from a table along with other changes to the same table, the changes would have to be done individually.
Figure 1. Enhanced Entity Relationship Diagram (Minus Attributes)
<table>
<thead>
<tr>
<th>Employee ID Code</th>
<th>Account</th>
<th>Travel</th>
<th>Labor</th>
<th>Sponsor</th>
<th>Other Leave</th>
<th>Contractor</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>Budget Page Date</td>
<td>TO #</td>
<td>TO Date</td>
<td>A.L. Hours</td>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>First Name</td>
<td>Fund Type</td>
<td>Proj Cost</td>
<td>Holiday Hrs</td>
<td></td>
<td>Address</td>
<td>Num Hrs</td>
</tr>
<tr>
<td>Mi</td>
<td>Labor JON</td>
<td>Actual Cost</td>
<td>S.L. Hours</td>
<td></td>
<td>Phone</td>
<td>Name</td>
</tr>
<tr>
<td>Last Name</td>
<td>MIPR #</td>
<td>Trav Start Date</td>
<td>L.W.O.P. Hrs</td>
<td></td>
<td></td>
<td>Address</td>
</tr>
<tr>
<td>Base Salary</td>
<td>Title</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Phone</td>
</tr>
<tr>
<td>Accel Rate</td>
<td>Serial #1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bldg #</td>
<td>Serial #2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Room #</td>
<td>Date Recvd</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Work Phone</td>
<td>Expir Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home Phone</td>
<td>Unit Fac Labor $</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Street Addr</td>
<td>Unit Spt Labor $</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City</td>
<td>Init OPTAR $</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td>Init Travel $</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip</td>
<td>Init Contract $</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Categories of Employee:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Faculty</td>
<td>Staff</td>
<td>Military</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Civ Grade</td>
<td>Civ Grade</td>
<td>Mil Grade</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step</td>
<td>Step</td>
<td>Service</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Attributes of Entities.

<table>
<thead>
<tr>
<th>Requests OPTAR Item From</th>
<th>Charges</th>
<th>Contracts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc #</td>
<td>Hours</td>
<td>PO #</td>
</tr>
<tr>
<td>PO #</td>
<td>Overtime Hours</td>
<td>Proj Cost</td>
</tr>
<tr>
<td>Proj Cost</td>
<td></td>
<td>Actual Cost</td>
</tr>
<tr>
<td>Actual Cost</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PO Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date Recvd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Order Date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADP Proj #</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Attributes of Relationships.
Figure 2. Physical Data Model
III. FINANCIAL MANAGEMENT SYSTEM

A. CLIENT/SERVER PROCESSING DECISION

We (my thesis advisor and I) decided to call the system the “Financial Management System” (FMS). Once the design of the Financial Management System database was complete, the development phase began. The solution being implementing utilized the “client/server” model of computing [Ref. 8] where some of the computing (processing) is done by the database residing on a “server” (a PC running the database server, in our case) and some of the computing is done by the application which runs on the “client” machine (again a PC in our case). A key part of the development phase was determining what would be done by the database (“back-end”), and what would be done by the application (“front-end”).

1. Database (Back-end) Processing

The database (back-end) handles the referential integrity constraints using triggers and it handles the calculation of the balance of the accounts using stored procedures. The reason for handling the referential integrity constraints using triggers is that S-Designer AppModeler automatically generated most of the triggers to enforce referential integrity thus having the tool do most of the work and making the database easier to maintain. The reason for calculating the balance of the accounts using stored procedures is so that the procedure would have to be written only once. It can be called by any trigger that would affect the balance of an account. Otherwise the code to calculate the balance of an account would have had to be
placed in every trigger that affects the balance of an account. A listing of the triggers is given in Appendix B, and a listing of the stored procedures is given in Appendix C.

Handling "referential integrity constraints" refers to ensuring the consistency of the data. In a relational database, a parent-child relationship can exist between tables. With a parent-child relationship, one or more records in the "child" table can refer to a record in the "parent" table. For example, in the FMS database there is a "parent" table called "DEPARTMENT" that contains information about academic departments such as the department code, department name, etc. A "child" table of DEPARTMENT is the table called "EMPLOYEE" which contains information about employees to include the department code of the department they belong to. The referential integrity constraint triggers in a database ensure that, for example, a record in the DEPARTMENT table cannot be deleted if EMPLOYEE records still exist with that department code (i.e., there are one or more records in the "child" EMPLOYEE table which reference the record to be deleted in the "parent" DEPARTMENT table). Figure 3 shows the attributes of the EMPLOYEE and DEPARTMENT tables and the arrow in the Figure from the attribute DEPT_CODE in the EMPLOYEE table to the attribute by the same name in the DEPARTMENT table illustrates the reference.

These integrity constraint "triggers" are Structured Query Language (SQL) code [Ref. 6] that are automatically executed upon occurrence of an event. The events that cause triggers to execute ("fire") are inserting, updating, and deleting of records. Triggers can be set to occur either before or after each of these events. S-Designer AppModeler automatically creates integrity constraints triggers for tables that have parent-child
relationships. The tasks performed by the triggers automatically created by *S-Designor AppModeler* include:

- The insert triggers ensure that a “parent” record exists (in the parent table) for every record inserted in a “child” table. If the parent record does not exist, the trigger does not allow the child record to be inserted.

- If the parent-child relationship is set to “delete prohibit,” delete triggers will not allow the deletion of a “parent” record if a “child” record still exists. However, if the relationship between a parent and child table has been set to “cascade” delete, the delete triggers will automatically delete child records if a parent record is deleted.

- The update triggers ensure that the field of a parent record which links it to a child record cannot be changed unless the trigger is set to automatically change the corresponding field in the child record.

The stored procedures which calculate the balance of each account are also SQL code. These stored procedures are called by triggers. When an event occurs that would change the balance of an account, such as the insertion of a travel record (i.e., a travel expense), the trigger causes the stored procedure to execute that calculates the travel balance of the account to be charged.

2. Application (Front-end) Processing

The application handles data validation. In other words, it only allows the user to enter data which meets data integrity constraints. For example, the application will not allow the user to enter a negative number for the number of days an individual was on travel. Of course the application cannot stop the user from entering incorrect data. For example, the user could enter that an individual was on travel for five days when they were actually on travel for three
days. The application would not catch the incorrect entry because five is in the range of valid numbers allowed to be entered in the field.

B. APPLICATION DEVELOPMENT

1. Background

As stated previously, a product called PowerBuilder Desktop was used to develop the application (front-end) of the FMS. PowerBuilder is a graphical application development tool for developing client/server applications that access databases. PowerBuilder provides pre-made standard window controls such as buttons, radiobuttons, checkboxes, dropdown listboxes, etc., to minimize the amount of coding that needs to be done by the developer. It also provides a scripting language with built-in functions which also help to minimize coding. Typically scripts are executed when an event occurs such as when a user clicks on a button.

A PowerBuilder application is made up of objects such as windows and menus. Objects are stored in PowerBuilder libraries and retrieved from these libraries when the application is run. Some of the types of PowerBuilder objects are:

- Application Object: the entry point into an application which defines application-level behavior such as what the default text font is and what processing should be done when the application begins or ends.

- Window Objects: the interface between the application and the user. They request information and display information.

- DataWindow Objects: used for retrieving and manipulating data from a relational database or other source such as a spreadsheet. It also determines the style of presentation of data such as tabular or freeform. Output from the database such as reports are retrieved and displayed using DataWindow objects.

- Menus: provides the user of the application with a list of choices (actions) to select from such as listing reports that can be produced.
• Global Functions: independent objects that perform general-purpose processing such as string handling.

• Queries: a SQL statement that is used to retrieve data from a relational database and saved with a name so that it can be reused. Normally they provide data for a DataWindow object.

• Structures: a collection of one or more related variables of possibly different data types grouped under a single name. This corresponds to the data structure called a “record” in Pascal and other programming languages. Structures allow the developer to refer to a set of related items as a single unit, rather than having to refer to multiple items.

• User Objects: an application feature defined by the user so that it can be reused in one or more applications.

• Libraries: as stated previously, PowerBuilder libraries are used to store objects. Applications retrieve the objects from the libraries so libraries can be shared by multiple applications.

• Projects: packages the application for execution by the application user(s). The application can be packaged as a stand-alone executable or as an executable that links to PowerBuilder dynamic libraries at execution time.

2. Implementation

   a. Financial Management System Modules

   The FMS, when complete, will consist of three modules (projects, in PowerBuilder terminology) — a staff module, a faculty module, and a chairman module. The purpose of the staff module is to provide the means for the academic department’s administrative staff to input data into the system and produce reports. The purpose of the faculty module is provide the means for the academic department’s faculty to check the status of the research accounts for which they are assigned as the principal investigator. The purpose of the chairman module is provide the means for the academic department’s chairman
to check the status of all of the department’s accounts and to perform planning and other accounting functions unique to the department chair. The staff module was developed as the prototype system for this thesis research project. The faculty module is developed but will not be discussed in this thesis.

b. **Staff Module Components**

The staff module of the FMS revolves around two main components as reflected by the majority of window objects used in the module. These window objects are employee related windows and account related windows. For both employees and accounts, there are list windows for providing a listing of all records with a minimum of attributes shown, detail windows for showing all of the attributes of one record, and search windows for searching for a specified employee or account record. From the employee detail window, the user can add or modify an employee record. (Note: employee records are normally not deleted. If an individual ceases to be a Naval Postgraduate School employee for whatever reason, an employment termination date attribute is filled in. If an employee record needs to be deleted because it was added in error, the staff member who made the entry asks the database administrator to delete the record.)

A screen shot of the employee detail window is shown in Figure 4. The employee detail window shows the accounts (if any) the employee is the principal investigator for. Every research account is assigned one or more principal investigators who are responsible for overseeing the research and authorizing the expenditure of funds in the research account in support of the research. Funding for the account is broken out into the following
categories: faculty labor, support labor, OPTAR (equipment), travel, and contracts (broken out as MIPR, IPA, and other contracts).

The account detail window displays details about the account such as the expiration date of the account, the account sponsor, and the initial and current balance of the account in each of the funding categories. A screen shot of the account detail window is shown in Figure 5.

As can be seen from Figure 5, there is a tab for each general funding category of the account. By clicking on a tab, the user can display more details about expenditures in that category. Example screen shots of expenditures for the labor, OPTAR, and travel funding categories of an account are shown in Figures 6, 7, and 8, respectively. When the user (staff member) clicks on a funding category tab, she can then add, modify, or delete records of expenditures for that funding category of the displayed account.

The PowerBuilder objects used by the staff module are stored in seven PowerBuilder libraries. The libraries are:

- **fms_main.pbl.** This object contains the main objects for the FMS staff module such as the main menu, the main window, the password window for logging in to the system, the “about” window which gives version and authorship information about FMS, and the toolbar configuration window which allows the user to select where to place the toolbar (sometimes known as a buttonbar). The toolbar allows the user to readily access employee, account and other windows by clicking on the buttons on the toolbar.

- **fms_emp.pbl.** This object contains employee related objects such as the employee detail window, the employee list window, the employee search window, and an employee list DataWindow for printing a list of employees.
• *fms_acct.pbl*. This object contains account related objects such as DataWindows for labor, OPTAR, travel, and contract expenditure listings for an account. These objects are shared by the faculty module of the FMS.

• *fms_acc2.pbl*. This object contains account related objects used solely by the staff module of the FMS such as the account list window, the account detail window, and the account search window.

• *fms_mnt.pbl*. This object contains maintenance related objects such as windows and DataWindows for adding, modifying or deleting records of labor, OPTAR, travel, and contract expenditures and adding, modifying or deleting records of sponsors of research accounts. These objects are shared by the faculty module of the FMS.

• *fms_mnt2.pbl*. This object contains maintenance related objects used solely by the staff module of the FMS such as windows and DataWindows for adding, modifying, and deleting employee and account records.

• *fms_rpt.pbl*. This object contains report related objects such as DataWindows for producing reports on labor, OPTAR, travel, and contract expenditures.

A complete listing of the objects contained in each *PowerBuilder* library of the FMS staff module is in Appendix D.

c. **Rapid Application Development**

A methodology that was used in developing the FMS staff module is known as Rapid Application Development (RAD) [Ref. 9]. This methodology, also known as ‘Rapid Prototyping,’ seeks to speed the development of a system by developing a quick prototype of the system, demonstrating the prototype to the eventual users of the system for their input, making changes to the system based on the users' input, and repeating the cycle until a deliverable product is developed [Ref. 10, 11]. As we developed the FMS staff module, we demonstrated it every two to four weeks to the Operations Research Department staff members who would be using the system. At times, the staff input not only resulted in
changes to the design of the application but also to the design of the database. Fortunately, the tools we were using, *S-Designer AppModeler* and *PowerBuilder Desktop*, allowed us to make changes to the database design relatively easily and with minimal impact on the application.

C. APPLICATION DEPLOYMENT

Once the FMS staff module prototype was developed to the point of being usable and with no obvious bugs, it was installed in the Operations Research Department for testing and debugging. Staff members were given a brief instruction on how to use the system and asked to use the system in parallel with existing systems to check the accuracy of the FMS. Staff members were also asked to report in writing all bugs they discovered and to request desired enhancements to the system in writing. Bug reports were evaluated to determine if an actual bug existed or whether the problem was due to operator error. If an actual bug existed, it was fixed and the fix was installed as soon as possible. Enhancement requests were evaluated to determine if they could reasonably be implemented. If so, the enhancement was made and installed. If not, the requester was notified why the requested enhancement could not be made to the system.
Figure 3. Parent-child Relationship of Employee and Department Tables
Figure 4. Employee Detail Window
### Detail Account Information

**Title:** LARGE SCALE OPTIMIZATION

<table>
<thead>
<tr>
<th>JON</th>
<th>Labor JON</th>
<th>Fund Type</th>
<th>Sponsor</th>
<th>Page Date</th>
<th>Rcv'd</th>
<th>Expiration</th>
</tr>
</thead>
<tbody>
<tr>
<td>RV1BR</td>
<td>RV1JRC</td>
<td>RR</td>
<td>ORN</td>
<td>11/12/96</td>
<td>11/12/96</td>
<td>9/30/97</td>
</tr>
</tbody>
</table>

**Remark:**

<table>
<thead>
<tr>
<th>OVERALL BUDGET SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>$116,000.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Summary</th>
<th>Summary</th>
<th>Labor</th>
<th>OPTAR</th>
<th>Travel</th>
<th>Cont MPR</th>
<th>Cont IPy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alloc</td>
<td>$100,321.00</td>
<td>$0.00</td>
<td>$21,367.00</td>
<td>$11,679.00</td>
<td>$4,000.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Used</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$21,367.00</td>
<td>$8,574.00</td>
<td>$0.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Bal</td>
<td>$100,321.00</td>
<td>$0.00</td>
<td>$0.00</td>
<td>$3,105.00</td>
<td>$4,000.00</td>
<td>$0.00</td>
</tr>
</tbody>
</table>
## Account Detail Information

**Title**: LARGE-SCALE OPTIMIZATION

<table>
<thead>
<tr>
<th>PI</th>
<th>Code</th>
<th>Serial #</th>
<th>Seg #</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ORBW</td>
<td>500EL-999EL</td>
<td></td>
</tr>
</tbody>
</table>

### OVERALL BUDGET SUMMARY

<table>
<thead>
<tr>
<th>Allocated</th>
<th>Used</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>$87,488.00</td>
<td>$57,265.71</td>
<td>$30,222.29</td>
</tr>
</tbody>
</table>

### Labor

<table>
<thead>
<tr>
<th>Pay Period</th>
<th>Code</th>
<th>Employee</th>
<th>Regular Hours</th>
<th>Overtime</th>
<th>Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>09-NOV-96</td>
<td>ORBW</td>
<td>BROWN</td>
<td>80</td>
<td>0</td>
<td>$6,243.95</td>
</tr>
<tr>
<td></td>
<td>ORBZ</td>
<td>BRADLEY</td>
<td>80</td>
<td>0</td>
<td>$5,879.02</td>
</tr>
<tr>
<td></td>
<td>ORWD</td>
<td>WOOD</td>
<td>80</td>
<td>0</td>
<td>$4,766.19</td>
</tr>
</tbody>
</table>

**Pay Period Total**: $16,891.16

<table>
<thead>
<tr>
<th>Pay Period</th>
<th>Code</th>
<th>Employee</th>
<th>Regular Hours</th>
<th>Overtime</th>
<th>Charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>26-OCT-96</td>
<td>ORBW</td>
<td>BROWN</td>
<td>72</td>
<td>0</td>
<td>$5,519.58</td>
</tr>
</tbody>
</table>

### Account Actions
- Search
- List
- Add
- Modify

### Close
### OPTAR Expenses

#### Overall Budget Summary

<table>
<thead>
<tr>
<th></th>
<th>Allocated</th>
<th>Used</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Budget</td>
<td>$51,000.00</td>
<td>$33,090.29</td>
<td>$17,909.71</td>
</tr>
</tbody>
</table>

#### OPTAR Entries

<table>
<thead>
<tr>
<th>Doc No</th>
<th>Description</th>
<th>PO #</th>
<th>ADP #</th>
<th>CYL</th>
<th>Est Cost</th>
<th>Act Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>96RG004J</td>
<td>MICROSOFT WINS NT RESOURCE KIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$168</td>
</tr>
<tr>
<td>96RG005J</td>
<td>DOS/4G UPGRADE FOR VIVATCOM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$999</td>
</tr>
<tr>
<td>96RG006J</td>
<td>VISUAL BASIC 4.0 PRO &amp; MASTER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$214</td>
</tr>
<tr>
<td>96RG007J</td>
<td>AMPL PLUS (INCLUDES AMPL ANNEX)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$1,258</td>
</tr>
<tr>
<td>96RG008J</td>
<td>COURIER DUAL STD INTERNAL MO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$1,294</td>
</tr>
<tr>
<td>96RG011J</td>
<td>MKS SOURCE INTEGRITY 7.1 CD-R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$150</td>
</tr>
<tr>
<td>96RG012J</td>
<td>UPDATE ONNET32 (5 PACK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$455</td>
</tr>
</tbody>
</table>

---

Figure 7. Account Detail Window Showing OPTAR Expenses
Figure 8. Account Detail Window Showing Travel Expenses

```
<table>
<thead>
<tr>
<th>TO#</th>
<th>Trav Date</th>
<th>#Days</th>
<th>Est Cost</th>
<th>Act Cost</th>
<th>Traveler</th>
</tr>
</thead>
<tbody>
<tr>
<td>n6227197300RA</td>
<td>11/1/96</td>
<td>2</td>
<td>$556.00</td>
<td>$400.00</td>
<td>Bus</td>
</tr>
<tr>
<td>N6227197TO301RA</td>
<td>11/2/96</td>
<td>5</td>
<td>$1,022.00</td>
<td>$1,022.00</td>
<td>Bus</td>
</tr>
</tbody>
</table>
```
IV. ANALYSIS

A. TOOLS

1. Database Modeling

The database modeling tool used, *S-Designer AppModeler* from Powersoft Corporation, allows the user to create a graphical representation of some of the components of a relational database. This includes tables, table attributes, relationships between tables, and views. These components are stored in what *S-Designer AppModeler* refers to as the "physical data model." Other components of the relational database, such as indexes, triggers, and stored procedures, can be created as part of the physical data model using *S-Designer AppModeler* but are not shown in the graphical representation.

Overall, we found *S-Designer AppModeler* (hereafter referred to as *AppModeler*) to be a very useful database modeling tool. As with any software tool, it has its strong points and weak points.

   a. Strong Points

- Overall ease of use. The user interface is fairly simple and straightforward. We were able to start using it with only a minimal amount of reading of the *User's Guide* and the on-line help. Sample physical data models were provided which also helped with learning how to use *AppModeler*. For preparing the graphical portion of the physical data model, several *AppModeler* tools are available in a tool palette: a table tool, a reference tool (for indicating the relationship between tables), a view tool, and so on. These tools in the tool palette make it simple for the user to create the tables, relationships, and views that are part of a database. A screen shot of *AppModeler* with the tool palette and the FMS physical data model is shown in Figure 9.
Automatic generation of the database. Once the user has completed a physical data model, with the click of the mouse, the database can be generated. The user has the option of having AppModeler generate the database, or generate an SQL script which can be executed separately to generate the database. Before the database or SQL script are generated, AppModeler automatically checks the model for correctness. The user can generate the database for any of a number of target databases such as Sybase SQL Anywhere and Oracle. Many other options are available. A screen shot of the AppModeler database generation screen is shown in Figure 10.

Automatic modification of the database. Automatic modification of the database is both a strong point and a weak point (see below). To modify the database, the user archives the current (prior to the changes) physical data model, makes changes to the physical data model, and then selects the Modify Database command. The user can choose to modify all tables or specify which tables to modify, modify all indexes or specify which indexes to modify, and modify all triggers and procedures or specify which triggers and procedures to modify. As with the automatic generation of the database, the user can choose to modify the database directly or to have an SQL script generated which can be executed separately to modify the database. It was very useful to select the option to generate the SQL script to check over what AppModeler was going to do to modify the database. If it appeared that the script would accomplish the intended modification, then the option to directly modify the database was selected. A screen shot of the AppModeler database modification screen is shown in Figure 11.

Automatic generation of indexes. Indexes provide an ordered list of the records of a table based on a key field. There are two types of key fields, primary and foreign. A primary key consists of one or more fields (attributes) that uniquely identify a record in a table. A foreign key is a field that depends on and migrates from a primary key in another table. With a few mouse clicks, the database indexes for key fields (both primary and foreign) can be automatically generated or, after modification of the database, regenerated.

Ease of creating relationships between tables. As mentioned previously, there is a “reference” tool in the AppModeler tool palette for creating relationships between tables. The user clicks on the Reference tool in the tool palette, clicks on the child table and drags the reference to the parent table. If the foreign key in the child table has the same name as the primary key in the parent table, those fields are automatically selected for the relationship. The user can specify which fields to use for the relationship if the correct fields are not automatically selected.

Automatic generation of referential integrity constraint triggers. AppModeler automatically created referential integrity constraint triggers for tables with parent-
child relationships. In every case, the triggers automatically generated by *AppModeler* worked correctly.

- Ease of creating and modifying triggers and stored procedures. In order to have the balance of the various funding categories of accounts calculated automatically, we had to create and modify some triggers and stored procedures. *AppModeler* made this task relatively easy by providing the means to list all triggers and procedures, listing triggers by table, and allowing the user to edit them with a simple but adequate text editor. As mentioned previously, once the user had created or modified the trigger or stored procedure, he could automatically add it to the database or modify it in the database using the automatic modification feature of *AppModeler*.

- Automatic documentation (report) generation. *AppModeler* can automatically generate three types of reports: a full report which contains all main model items, a standard report which contains physical data model graphics, and all table-dependent items, and a list report which contains a single title item and all list-type items. User-defined reports can also be created. The user can print the report or save it in "Rich Text Format" to a file. Additionally, the user can choose to print the physical data model graph in color or black and white and can have *AppModeler* automatically scale the graph so that it fits on one page (an extremely useful feature). Part of the *AppModeler* full report (database schema information) for the FMS physical data model is given in Appendix E.

### b. Weak Points

- Automatic modification of the database. If too many changes were attempted at once, *AppModeler* did not have the intelligence to perform them in an order that would achieve the desired results and thus end up with a physical data model that did not match the actual database. That is why it is extremely helpful for the user to first have *AppModeler* generate the SQL script and to check the script before having *AppModeler* directly modify the database. The other problem observed was that frequently *AppModeler* could not perform modification of a key field because it did not have the intelligence to perform the necessary steps. Modifying a key field usually had to be done manually in several steps. First, the data from the table had to be exported to a comma-delimited file. Then the user had to delete any relationships with the table and the table itself and use the automatic modification feature to implement this on the database. Then the user had to recreate the table with the desired change to the key field and recreate the relationships for that table and again use the automatic modification feature to implement the changes on the database. Finally, the user had to import the data.
from the comma-delimited file back into the table. On occasion the user had to first manipulate the contents of the comma-delimited file (using a spreadsheet or other program) to get it into a form that would be accepted by the modified table before importing it into the modified table. In other words, the automatic modification feature was, at times, dangerous and/or time-consuming.

- Graphical representation of the database. This was a weakness in the sense that AppModeler could not work with an EER diagram. A preferable method is to create and modify an EER diagram and have AppModeler generate the table, attributes, relationships, and so on, from that.

- Automatic generation of relationships. The automatic generation of relationships (references) in AppModeler created a relationship between every primary and foreign key with the same name. In our case, this created many relationships that were not intended and so we found it far easier to manually create the desired relationships using the Reference tool in the tool palette.

2. Application Development

The application development tool used was PowerBuilder Desktop from Powersoft Corporation. PowerBuilder is a tool for developing graphical client/server applications that access relational databases. As such it attempts to minimize the amount of coding done by the developer in order to make it easier and faster to develop and maintain the application.

Overall, we found that PowerBuilder did live up to its stated purpose of easing the development and maintenance of an application. Some of its strong and weak points are listed here.

a. Strong Points

- Pre-made standard window controls. PowerBuilder made it easy to design menus and other standard windowing controls and thus saved a great deal of coding.

- Ability of multiple applications to share libraries. Some of the libraries were used for multiple modules (projects) of the FMS, which made it much quicker to develop the modules and maintain them.
• Reusable objects. PowerBuilder objects we created, such as DataWindows, were saved in libraries and reused within a module (project) and by multiple modules.

• *PowerBuilder* Painters. Similar to the tool palette of *AppModeler*, *PowerBuilder* had “painters” for creating *PowerBuilder* objects such as DataWindows, Applications, Projects, Menus, and so on. These painters provided an easy to use interface for creating these objects.

• Support. *PowerBuilder* is a fairly widely used product and consequently there exists a support forum for it on the computer service called CompuServe. The support forum is available at no extra charge for CompuServe subscribers and is made up of users of *PowerBuilder* (not Powersoft employees). On the occasions where we ran into problems with *PowerBuilder* that we could not solve, we posted a message detailing the problem on the support forum on CompuServe and received an answer usually within twenty-four hours that solved the problem. This form of support was important for keeping the cost of the project down since technical support from Powersoft is not free.

\[b. \quad \textit{Weak Points}\]

• Difficulty in changing fonts and font sizes. For various reasons, the font and/or font size for some of the windows and reports were changed several times. Unfortunately there was no means available to make a global change. Consequently, each text object had to be changed individually, making it a very tedious and time consuming process.

• Scripting language awkward. The scripting language is not designed logically. Too many features are ad hoc add-ons.

• The executable is not truly compiled. It requires the application's dynamic library files in order to work.

• Inadequate documentation. The manual for *PowerBuilder* was the smallest of the manuals for the three programs that made up *Powersoft Portfolio*. Not only was it the smallest but it was also the least adequate. We found it necessary to purchase third-party books about *PowerBuilder* to supplement the manual.
B. DATABASE SERVER

The database server used is Sybase SQL Anywhere. Powersoft Portfolio included a four-user version of Sybase SQL Anywhere. That means that four individuals can concurrently be logged in to the database server (users accessing the FMS application are logged in to the database server). This database server, in previous releases, was know as Watcom SQL Server. The dialect of SQL implemented by Sybase SQL Anywhere is Watcom-SQL. (Note: Every database server implements its own “dialect” of SQL that consist of what might be called “standard” SQL plus some extensions to it. It is similar to the various implementations of programming languages such as Pascal, BASIC, FORTRAN, and so on, by software vendors.) The database server allows a database application to communicate with a database over a network and it handles the processing done by the database, i.e., the “back-end” processing of a client/server application. Users must enter a valid user ID and password to make a connection (log in) to the database server. The Sybase SQL Anywhere server will run on a variety of platforms including: Novell NetWare, Windows 95, Windows NT, OS/2, Windows 3.x, and DOS. No matter what platform that Sybase SQL Anywhere is running on, it can be accessed by clients operating with different operating systems, such as DOS, Windows 95, Macintosh, running on different kinds of networks such as Novell NetWare, Windows NT, and Banyan Vines.

Overall, we were pleased with the Sybase SQL Anywhere database server. Some of its strong and weak points are listed here.
1. **Strong Points**

- Runs on multiple platforms. At first we ran the database server on a Novell NetWare server. During a time period when we were having a problem with the database server, occurrence of certain events could cause the database server to crash. When trying to recover the database server from the crash, it would sometimes cause the Novell server to crash. Because **Sybase SQL Anywhere** runs on a variety of platforms, we were able to move it to run on a networked PC running Windows 95 so that if the database server crashed, it did not affect the Novell server.

- Ease of use. **Sybase SQL Anywhere** was very easy to start up and administer.

- Support. As with **PowerBuilder**, a support forum is available on CompuServe for **Sybase SQL Anywhere** that is free for CompuServe subscribers. Also as with **PowerBuilder**, we posted problems we had with **Sybase SQL Anywhere** on the forum and received correct solutions usually within twenty-four hours.

- Documentation. **Powersoft Portfolio** contained three manuals for **Sybase SQL Anywhere**. These included a Watcom-SQL reference that we made good use of for writing the stored procedures and triggers for the FMS. These manuals were also available on-line so the user can easily search for specific topics.

2. **Weak Points**

- No automatic backup of the database. When the database server is running, the database files are open. Software for tape backup systems cannot backup files that are open. We wanted to have regular backups of the database but that meant we had to shut down the database server at the end of the workday (the tape backup automatically ran at night) and then start it up again at the beginning of the workday. It would have been very helpful if the database server could have been automatically scheduled to start and stop at specified times.

- Database server crash caused Novell server crash. As mentioned in the strong point about **Sybase SQL Anywhere** running on multiple platforms, for a time we had a problem with the database server crashing and, in turn, causing the Novell server it was running on to crash. That was very disruptive to the users of the Novell server and was totally unsatisfactory. We did receive information via the forum on CompuServe on how to fix the problem but we decided to move the database server off the Novell server to a PC just to be safe.
• Inability to handle a query with many outer joins. The event that caused the database server to crash was the execution of a query with many outer joins. This problem was a bug that had purportedly been fixed in an earlier release of Sybase SQL Anywhere but had apparently been reintroduced into the version we were using. The end result was that the queries had to be rewritten without the outer joins since Sybase SQL Anywhere could not handle them even though it was supposed to be able to do so.

C. PROTOTYPE

The FMS prototype was installed in the Operations Research Department for testing and debugging in September 1996. As with any new system, many bugs have been discovered and a variety of enhancements have been requested but overall, we believe the system has been well received. A listing of strong and weak points follow.

1. Strong Points

• Ease of use. The users of the FMS were provided with very brief instructions on how to log into the application and do a few simple tasks. They have been able to effectively use the system without any additional instruction.

• Maintainability. We have been able to make changes to the system to fix bugs and to implement enhancement requests with relative ease. Bugs are usually fixed within a few hours. Simple enhancement requests have also been completed within a few hours but the more complex enhancement requests (ones that involved a design change) have taken a couple of days to implement (lapsed time -- the actual work took no more than a day per added feature). The ease of maintainability is due in large part to the software tools we have been using as discussed earlier in this chapter.

2. Weak Points

• Error messages. Due to a lack of time, we have not prepared error messages for all of the situations that users can cause errors. In situations where the FMS does not trap errors and provide an error message, error messages are generated by the
Sybase SQL Anywhere database server. Probably the most frequent error the user makes is to attempt an action that violates referential integrity. The error messages produced by the database server in these (and all other) situations are not comprehensible to the ordinary user. Instead, the error messages confuse the user and discourage him from using the system. We are correcting this deficiency as time permits.

- Lack of user generated reports. We have not provided the user with a means to generate reports of his own design. The complexities involved in providing such a capability to the user dictate that if it is implemented, it will provide a fairly rudimentary report generation capability. It may be possible, however, to train the users to utilize a Powersoft product called InfoModeler to produce reports. One of the purposes of InfoModeler is to provide an easy means for end-users to produce reports from a Sybase SQL Anywhere database.
Figure 9. AppModeler with the Tool Palette Displayed.
Figure 10: AppModeler Database Generation Window
Figure 11. AppModeler Database Modification Window
V. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE STUDY

A. CONCLUSIONS

The prototype Financial Management System currently deployed in the Operations Research Department is nearly a production system which, with some modifications, could be used as an accounting system for all the academic departments at the Naval Postgraduate School to track their financial accounts. The prototype has demonstrated that even though user requirements frequently change, it can be changed to meet new requirements relatively quickly and easily. Comparing the EER diagram in Figure 1 to the physical data model in Figure 2, it is obvious that the design of the FMS changed a great deal over the course of this thesis project. Yet, the majority of changes were implemented within a few a days of the decision to change the design. This quick turn-around for implementing design changes would not have been possible if this project had been prepared using only a programming language such as C++.

The tools used (those contained in Powersoft Portfolio) were an invaluable part of this project and very inexpensive when compared to some of the other tools on the market. That is not to say that Powersoft Portfolio is the best client/server application development tool set available for those on a tight budget. It did, however, meet the needs of this project and we would recommend it for use by others with similar needs and resources.

Changes and additions need to be made to the FMS. The faculty module has been developed but it needs to be deployed for testing and debugging. Error conditions in the staff module need to be trapped and clear error messages displayed when errors occur. An on-line
help system needs to be added and the users need to be able to easily produce rudimentary reports from the data available. These changes and additions can be made to the system relatively easily using the tools we have available when time permits.

B. RECOMMENDATIONS FOR FUTURE STUDY

The system could be extended to become an automated aid for the academic departments. By extending the database and the application, the system could be used for property management, scheduling classes, and managing other databases used by the departments. This would prevent the same data from being entered multiple times into separate databases. For example, accountable property is tagged with a minor or plant property tag and entered into a database with various attributes about each piece of property. Much of this property is purchased by academic departments from their various accounts and many of the same attributes about this property are stored in the FMS table called OPTAR_REQ as are stored in the property database. Since the FMS is a relational database, it could be made to interface with this property database, i.e., have relationships created with a modified form of the property database tables. Another relation could be created for property that was maintained by staff members at the school, such as computer hardware, so those staff members could keep a record of maintenance performed on the property. Other existing systems at the Naval Postgraduate School such as SACONS (Standard Automated Contracting System) could also be made to interface with the FMS to further reduce multiple entries of the same data and other problems associated with having separate databases that contain essentially the same information. In fact, these existing systems should also be
analyzed for possible changes to maximize the benefits available through the use of client/server database applications.

A “chairman’s” module still needs to be developed for the FMS to assist the academic department chairman in planning the expenditure of funds, especially at the beginning of each fiscal year.

A course information database would be another useful addition to the FMS. It could be used to relate planned instruction (courses) to the expenditure of funds for supplies and labor needed to support instruction.
LIST OF REFERENCES


APPENDIX B. FMS DATABASE TRIGGERS

%% ===============================================================================
%% Database name:  FMS
%% DBMS name:  Watcom SQL 4.0
%% Created on:  2/3/97  4:52 PM
%% ===============================================================================

% Before insert trigger "tib_account" for table "ACCOUNT"
create trigger tib_account before insert on ACCOUNT
referencing new as new_ins for each row
begin
  declare user_defined_exception exception for SQLSTATE '99999';
  declare found integer;

  % Parent "SPONSOR" must exist when inserting a child in "ACCOUNT"
  if (new_ins.SPON_ID_CODE is not null) then
    begin
      set found = 0;
      select 1
      into found
      from dummy
      where exists (select 1
                      from SPONSOR
                      where SPON_ID_CODE = new_ins.SPON_ID_CODE);
      if found <> 1 then
        signal user_defined_exception
      end if;
    end
  end if;
end
/

% After insert trigger "tia_account" for table "ACCOUNT"
create trigger tia_account after insert on ACCOUNT
referencing new as new_ins for each row
begin
  call CALC_BAL_CONTRACT(new_ins.JON,'M');
  call CALC_BAL_CONTRACT(new_ins.JON,'I');
  call CALC_BAL_CONTRACT(new_ins.JON,'O');
  call CALC_BAL_FAC_LABOR(new_ins.JON);
  call CALC_BAL_SPT_LABOR(new_ins.JON);
  call CALC_BAL_OPTAR(new_ins.JON);
  call CALC_BAL_TRAV(new_ins.JON);
end
/

% Update trigger "tua_account" for table "ACCOUNT"
create trigger tua_account after update of INIT_FAC_LABOR $,
     INIT_SPT_LABOR $,
     INIT_TRAVEL $,
on ACCOUNT
referencing new as new_upd old as old_upd for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;
    call CALC_BAL_CONTRACT(new_upd.JON,'M');
    call CALC_BAL_CONTRACT(new_upd.JON,'I');
    call CALC_BAL_CONTRACT(new_upd.JON,'O');
    call CALC_BAL_FAC_LABOR(new_upd.JON);
    call CALC_BAL_SPT_LABOR(new_upd.JON);
    call CALC_BAL_OPTAR(new_upd.JON);
    call CALC_BAL_TRAV(new_upd.JON);
end
/

% Before insert trigger "tib_adp_proj_info" for table "ADP_PROJ_INFO"
create trigger tib_adp_proj_info before insert on ADP_PROJ_INFO
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "DEPARTMENT" must exist when inserting a child in
    "ADP_PROJ_INFO"
    if (new_ins.DEPT_CODE is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                          from DEPARTMENT
                          where DEPT_CODE = new_ins.DEPT_CODE);
            if found <> 1 then
                signal user_defined_exception
            end if;
        end
    end if;

    % Parent "EMPLOYEE" must exist when inserting a child in
    "ADP_PROJ_INFO"
    if (new_ins.PROJ_MGR_CODE is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
from EMPLOYEE
where EMP_ID_CODE = new_ins.PROJ_MGR_CODE;
if found <> 1 then
  signal user_defined_exception
end if;
end if;

% Parent "EMPLOYEE" must exist when inserting a child in
"ADP_PROJ_INFO"
if (new_ins.POC_CODE is not null) then
begin
  set found = 0;
  select 1
  into found
  from dummy
  where exists (select 1
    from EMPLOYEE
    where EMP_ID_CODE = new_ins.POC_CODE);
  if found <> 1 then
    signal user_defined_exception
  end if;
end if;
end
/

% Before insert trigger "tib_contracts" for table "CONTRACTS"
create trigger tib_contracts before insert on CONTRACTS
referencing new as new_ins for each row
begin
  declare user_defined_exception exception for SQLSTATE '99999';
  declare found integer;

  % Parent "ACCOUNT" must exist when inserting a child in "CONTRACTS"
  if (new_ins.JON is not null) then
  begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
      from ACCOUNT
      where JON = new_ins.JON);
    if found <> 1 then
      signal user_defined_exception
    end if;
  end if;
end if;

% Parent "EMPLOYEE" must exist when inserting a child in "CONTRACTS"
if (new_ins.REQUESTER is not null) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
                  from EMPLOYEE
                  where EMP_ID_CODE = new_ins.REQUESTER);
    if found <> 1 then
        signal user_defined_exception
    end if;
end
end if;
end
/

% After insert trigger "tia_contracts" for table "CONTRACTS"
create trigger tia_contracts after insert on CONTRACTS
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';

    call CALC_BAL_CONTRACT(new_ins.JON,new_ins.CONTRACT_TYPE)
end
/

% Before update trigger "tub_contracts" for table "CONTRACTS"
create trigger tub_contracts before update of JON,
CONTRACT_TYPE,
REQUESTER,
DOC_

on CONTRACTS
referencing new as new_upd old as old_upd for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "ACCOUNT" must exist when updating a child in "CONTRACTS"
    if (new_upd.JON is not null and
        ((old_upd.JON is null) or
         (new_upd.JON <> old_upd.JON))) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                          from ACCOUNT
                          where JON = new_upd.JON);
            if found <> 1 then
                signal user_defined_exception

            end if;
        end
    end if;
end
end if;
end
end if;

% Parent "EMPLOYEE" must exist when updating a child in "CONTRACTS"
if (new_upd.REQUESTER is not null and
    ((old_upd.REQUESTER is null) or
     (new_upd.REQUESTER <> old_upd.REQUESTER))) then
begin
    set found = 0;
    select 1
    into found from dummy
    where exists (select 1
                  from EMPLOYEE
                  where EMP_ID_CODE = new_upd.REQUESTER);
    if found <> 1 then
        signal user_defined_exception
    end if;
end
end if;

% Cannot modify parent code of "EMPLOYEE" in child "CONTRACTS"
if ((new_upd.REQUESTER is null and old_upd.REQUESTER is not null) or
    new_upd.REQUESTER <> old_upd.REQUESTER ) then
    signal user_defined_exception
end if;
/

% Update trigger "tua_contracts" for table "CONTRACTS"
create trigger tua_contracts after update of JON,
    CONTRACT TYPE,
    CONTRACTOR_ID,
    PROJ_COST,
    ACTUAL_COST
on CONTRACTS
referencing new as new_upd old as old_upd for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    call CALC_BAL_CONTRACT(new_upd.JON,new_upd.CONTRACT_TYPE)
end
/

% After delete trigger "tda_contracts" for table "CONTRACTS"
create trigger tda_contracts after delete on CONTRACTS
referencing old as old_del for each row
begin
end
declare user_defined_exception exception for SQLSTATE '99999';
declare found integer;

call CALC_BAL_CONTRACT(old_del.JON,old_del.CONTRACT_TYPE)
end /

% Before insert trigger "tib_employee" for table "EMPLOYEE"
create trigger tib_employee before insert on EMPLOYEE
referencing new as new_ins for each row
begin
  declare user_defined_exception exception for SQLSTATE '99999';
declare found integer;

  % Parent "DEPARTMENT" must exist when inserting a child in "EMPLOYEE"
  if (new_ins.DEPT_CODE is not null) then
    begin
      set found = 0;
      select 1
      into  found
      from  dummy
      where exists (select 1
                    from  DEPARTMENT
                    where  DEPT_CODE = new_ins.DEPT_CODE);
      if found <> 1 then
        signal user_defined_exception
      end if;
    end
  end if;
end /

% Before insert trigger "tib_faculty" for table "FACULTY"
create trigger tib_faculty before insert on FACULTY
referencing new as new_ins for each row
begin
  declare user_defined_exception exception for SQLSTATE '99999';
declare found integer;

  % Parent "EMPLOYEE" must exist when inserting a child in "FACULTY"
  if (new_ins.EMP_ID_CODE is not null) then
    begin
      set found = 0;
      select 1
      into  found
      from  dummy
      where exists (select 1
                    from  EMPLOYEE
                    where  EMP_ID_CODE = new_ins.EMP_ID_CODE);
      if found <> 1 then

signal user_defined_exception
end if;
end
end if;
end
/

% Before insert trigger "tib_labor_chgs" for table "LABOR_CHGS"
create trigger tib_labor_chgs before insert on LABOR_CHGS

begin

    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "LABOR_LES" must exist when inserting a child in "LABOR_CHGS"
    if (new_ins.EMP_ID_CODE is not null and
        new_ins.PPE_DATE is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                from LABOR_LES
                where EMP_ID_CODE = new_ins.EMP_ID_CODE
                and PPE_DATE = new_ins.PPE_DATE);

            if found <> 1 then
                signal user_defined_exception
            end if;
        end
    end if;

    % Parent "ACCOUNT" must exist when inserting a child in "LABOR_CHGS"
    if (new_ins.JON is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                from ACCOUNT
                where JON = new_ins.JON);

            if found <> 1 then
                signal user_defined_exception
            end if;
        end
    end if;

    % Parent "EMPLOYEE" must exist when inserting a child in "LABOR_CHGS"
    if (new_ins.EMP_ID_CODE is not null) then
        begin

55
set found = 0;
select 1
  into found
  from dummy
where exists (select 1
  from EMPLOYEE
  where EMP_ID_CODE = new_ins.EMP_ID_CODE);
if found <> 1 then
  signal user_defined_exception
end if;
end if;
end
/

% After insert trigger "tia_labor_chgs" for table "LABOR_CHGS"
create trigger tia_labor_chgs after insert on LABOR_CHGS
  referencing new as new_ins for each row
begin
  declare user_defined_exception exception for SQLSTATE '99999';
  declare found integer;
  declare emp_cat char(1);
  declare jcn_type char(2);
  declare base_sal numeric(10,2);
  declare hourly_rate numeric (7,2);
  declare hourly_ot_rate numeric(7,2);
  declare otm_cap numeric(7,2);
  declare yr_hrs integer;
  declare rr_ot_fac numeric(6,4);
  declare sal_eff date;
  declare acc_rate decimal(3,2);

  select OT_CAP into otm_cap from FMS_CFG;
  select YR_LABOR_HRS into yr_hrs from FMS_CFG;
  select RR_OT_RATE_FACT into rr_ot_fac from FMS_CFG;

  %Calculate the "TOTALCHG" field
  if (new_ins.EMP_ID_CODE is not null) then
    begin
      set found=0;
      select 1
        into found
        from dummy
      where exists (select 1
        from EMPLOYEE
        where EMP_ID_CODE=new_ins.EMP_ID_CODE);

      select EFF_SAL_DATE into sal_eff from EMPLOYEE
      where new_ins.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;

      if (new_ins.PPE_DATE >= sal_eff) then
        begin

          56
select BASE_SALARY into base_sal from EMPLOYEE
where new_ins.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;
select ACCEL_RATE into acc_rate from EMPLOYEE
where new_ins.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;
end
else
begin
select BASE_SALARY into base_sal from SALARY_HISTORY
where new_ins.EMP_ID_CODE=SALARY_HISTORY.EMP_ID_CODE
and new_ins.PPE_DATE >= SALARY_HISTORY.BEGIN_DATE
and new_ins.PPE_DATE <= SALARY_HISTORY.END_DATE;
select ACCEL_RATE into acc_rate from SALARY_HISTORY
where new_ins.EMP_ID_CODE=SALARY_HISTORY.EMP_ID_CODE
and new_ins.PPE_DATE >= SALARY_HISTORY.BEGIN_DATE
and new_ins.PPE_DATE <= SALARY_HISTORY.END_DATE;
end
end if;

set hourly_rate=base_sal/yr_hrs;

if ((hourly_rate*1.5) > otm_cap) then
  set hourly_ot_rate=otm_cap
else
  set hourly_ot_rate=hourly_rate*1.5
end if;

select FUND_TYPE into jon_type from ACCOUNT
where new_ins.JON=ACCOUNT.JON;

select CATEGORY into emp_cat from EMPLOYEE
where new_ins.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;

if (jon_type='RR') then
  begin
    if (emp_cat='F') then
      begin
        update LABOR_CHGS,EMPLOYEE
        set TOTAL_CHG=(HOURS*hourly_rate*acc_rate)
where LABOR_CHGS.EMP_ID_CODE=new_ins.EMP_ID_CODE
and LABOR_CHGS.PPE_DATE=new_ins.PPE_DATE
and LABOR_CHGS.JON=new_ins.JON
and new_ins.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE
      else
        if (emp_cat='S') then
          begin
            update LABOR_CHGS,EMPLOYEE
            set TOTAL_CHG=(HOURS*hourly_rate*acc_rate)+
            (OT_HOURS*hourly_ot_rate*rr_ot_fac)
where LABOR_CHGS.EMP_ID_CODE=new_ins.EMP_ID_CODE
and LABOR_CHGS.PPE_DATE=new_ins.PPE_DATE
and LABOR_CHGS.JON=new_ins.JON
and new_ins.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE
          end if
        end if
    end if
end if
end
else
begin
if (emp_cat='F') then
update LABOR_CHGS
set TOTAL_CHG=(HOURS*hourly_rate)
where LABOR_CHGS.EMP_ID_CODE=new_ins.EMP_ID_CODE
and LABOR_CHGS.PPE_DATE=new_ins.PPE_DATE
and LABOR_CHGS.JON=new_ins.JON
else
if (emp_cat='S') then
update LABOR_CHGS
set TOTAL_CHG=(HOURS*hourly_rate)+
(OT_HOURS*hourly_ot_rate)
where LABOR_CHGS.EMP_ID_CODE=new_ins.EMP_ID_CODE
and LABOR_CHGS.PPE_DATE=new_ins.PPE_DATE
and LABOR_CHGS.JON=new_ins.JON
end if
end if
end if;
if (emp_cat='F') then
  call CALC_BAL_FAC_LABOR(new_ins.JON)
else
  if (emp_cat='S') then
    call CALC_BAL_SPT_LABOR(new_ins.JON)
  end if
end if;
if (found <> 1) then
  signal user_defined_exception
end if;
end if;
end;
/

% Update trigger "tua_labor_chgs" for table "LABOR_CHGS"
create trigger tua_labor_chgs after update of EMP_ID_CODE,
PPE_DATE,
JON,
HOURS,
OT_HOURS
on LABOR_CHGS
referencing new as new_upd old as old_upd for each row
begin
declare user_defined_exception exception for SQLSTATE '99999';
declare found integer;
declare emp_cat char(1);
declare jon_type char(2);
declare base_sal numeric(10,2);
declare hourly_rate numeric (7,2);
declare hourly_ot_rate numeric(7,2);
declare otm_cap numeric(7,2);
declare yr_hrs integer;
declare rr_ot_fac numeric(6,4);
declare sal_eff date;
declare acc_rate decimal(3,2);

select OT_CAP into otm_cap from FMS_CFG;
select YR_LABOR_HRS into yr_hrs from FMS_CFG;
select RR_OT_RATE_FACT into rr_ot_fac from FMS_CFG;

/*Calculate the "TOTALCHG" field*/
if ((new_upd.HOURS<>old_upd.HOURS) or
   (new_upd.OT_HOURS<>old_upd.OT_HOURS)) then
begin
    set found=0;
    select 1
    into found
    from dummy
    where exists (select 1
                   from EMPLOYEE
                   where EMP_ID_CODE=new_upd.EMP_ID_CODE);

    select EFF_SAL_DATE into sal_eff from EMPLOYEE
        where new_upd.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;

    if (new_upd.PPE_DATE >= sal_eff) then
begin
    select BASE_SALARY into base_sal from EMPLOYEE
        where new_upd.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;
    select ACCEL_RATE into acc_rate from EMPLOYEE
        where new_upd.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;
end
else
begin
    select BASE_SALARY into base_sal from SALARY_HISTORY
        where new_upd.EMP_ID_CODE=SALARY_HISTORY.EMP_ID_CODE
        and new_upd.PPE_DATE >= SALARY_HISTORY.BEGIN_DATE
        and new_upd.PPE_DATE <= SALARY_HISTORY.END_DATE;
    select ACCEL_RATE into acc_rate from SALARY_HISTORY
        where new_upd.EMP_ID_CODE=SALARY_HISTORY.EMP_ID_CODE
        and new_upd.PPE_DATE >= SALARY_HISTORY.BEGIN_DATE
        and new_upd.PPE_DATE <= SALARY_HISTORY.END_DATE;
end
end if;

set hourly_rate=base_sal/yr_hrs;

if (((hourly_rate*1.5) > otm_cap) then
    set hourly_ot_rate=otm_cap
else
    set hourly_ot_rate=hourly_rate*1.5
end if;

select FUND_TYPE into jon_type from ACCOUNT
where new_upd.JON=ACCOUNT.JON;

select CATEGORY into emp_cat from EMPLOYEE
where new_upd.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;

if (jon_type='RR') then
begin
    if (emp_cat='F') then
        update LABOR_CHGS,EMPLOYEE
            set TOTAL_CHG=(HOURS*hourly_rate*acc_rate)
            where LABOR_CHGS.EMP_ID_CODE=new_upd.EMP_ID_CODE
                and LABOR_CHGS.PPE_DATE=new_upd.PPE_DATE
                and LABOR_CHGS.JON=new_upd.JON
                and new_upd.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE
    else
        if (emp_cat='S') then
            update LABOR_CHGS,EMPLOYEE
                set TOTAL_CHG=(HOURS*hourly_rate*acc_rate)+'
                    (OT_HOURS*hourly_ot_rate*rr_ot_fac)
                where LABOR_CHGS.EMP_ID_CODE=new_upd.EMP_ID_CODE
                    and LABOR_CHGS.PPE_DATE=new_upd.PPE_DATE
                    and LABOR_CHGS.JON=new_upd.JON
                    and new_upd.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE
        end if
    end if
end if
else
begin
    if (emp_cat='F') then
        update LABOR_CHGS
            set TOTAL_CHG=(HOURS*hourly_rate)
            where LABOR_CHGS.EMP_ID_CODE=new_upd.EMP_ID_CODE
                and LABOR_CHGS.PPE_DATE=new_upd.PPE_DATE
                and LABOR_CHGS.JON=new_upd.JON
    else
        if (emp_cat='S') then
            update LABOR_CHGS
                set TOTAL_CHG=(HOURS*hourly_rate)+
                    (OT_HOURS*hourly_ot_rate)
                where LABOR_CHGS.EMP_ID_CODE=new_upd.EMP_ID_CODE
                    and LABOR_CHGS.PPE_DATE=new_upd.PPE_DATE
                    and LABOR_CHGS.JON=new_upd.JON
        end if
    end if
end if
end if;
if (emp_cat='F') then
    call CALC_BAL_FACLABOR(new_upd.JON)
else
    if (emp_cat='S') then
        call CALC_BAL_SPTLABOR(new_upd.JON)
    end if
end if;

if (found <> 1) then
    signal user_defined_exception
end if;
end
end if;
/

% After delete trigger "tda_labor_chgs" for table "LABOR_CHGS"
create trigger tda_labor_chgs after delete on LABOR_CHGS
referencing old as old_del for each row
begin

    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;
    declare emp_cat char(1);

    select CATEGORY into emp_cat from EMPLOYEE
        where old_del.EMP_ID_CODE=EMPLOYEE.EMP_ID_CODE;

    if (emp_cat='F') then
        call CALC_BAL_FACLABOR(old_del.JON)
    elseif (emp_cat='S') then
        call CALC_BAL_SPTLABOR(old_del.JON)
    end if;
end
/

% Before insert trigger "tib_labor_les" for table "LABOR_LES"
create trigger tib_labor_les before insert on LABOR_LES
referencing new as new_ins for each row
begin

    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "EMPLOYEE" must exist when inserting a child in "LABOR_LES"
    if (new_ins.EMP_ID_CODE is not null) then
        begin
            set found = 0;
            select 1
                into found
            from dummy
            where exists (select 1
from EMPLOYEE
   where EMP_ID_CODE = new_ins.EMP_ID_CODE);
   if found <> 1 then
      signal user_defined_exception
   end if;
   end if;
end
/

% Before insert trigger "tib_military" for table "MILITARY"
create trigger tib_military before insert on MILITARY
   referencing new as new_ins for each row
begin
   declare user_defined_exception exception for SQLSTATE '99999';
   declare found integer;

   % Parent "EMPLOYEE" must exist when inserting a child in "MILITARY"
   if (new_ins.EMP_ID_CODE is not null) then
      begin
         set found = 0;
         select 1
         into found
         from dummy
         where exists (select 1
                        from EMPLOYEE
                        where EMP_ID_CODE = new_ins.EMP_ID_CODE);
         if found <> 1 then
            signal user_defined_exception
         end if;
      end
   end if;
end
/

% Before insert trigger "tib_optar_req" for table "OPTAR_REQ"
create trigger tib_optar_req before insert on OPTAR_REQ
   referencing new as new_ins for each row
begin
   declare user_defined_exception exception for SQLSTATE '99999';
   declare found integer;

   % Parent "EMPLOYEE" must exist when inserting a child in "OPTAR_REQ"
   if (new_ins.EMP_ID_CODE is not null) then
      begin
         set found = 0;
         select 1
         into found
         from dummy
         where exists (select 1
                        from EMPLOYEE
                        where EMP_ID_CODE = new_ins.EMP_ID_CODE);
if found <> 1 then
    signal user_defined_exception
end if;
end if;

% Parent "ACCOUNT" must exist when inserting a child in "OPTAR_REQ"
if (new_ins.JON is not null) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
                   from ACCOUNT
                   where JON = new_ins.JON);
    if found <> 1 then
        signal user_defined_exception
    end if;
end
end if;

% Parent "ADP_PROJ_INFO" must exist when inserting a child in
"OPTAR_REQ"
if (new_ins.ADPROJ_# is not null) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
                   from ADP_PROJ_INFO
                   where ADPROJ_# = new_ins.ADPROJ_#);
    if found <> 1 then
        signal user_defined_exception
    end if;
end
end if;
/

% After insert trigger "tia_optar_req" for table "OPTAR_REQ"
create trigger tia_optar_req after insert on OPTAR_REQ
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    call CALC_BAL_OPTAR(new_ins.JON);
end
/
% Update trigger "tua_optar_req" for table "OPTAR_REQ"
cREATE TRIGGER tua_optar_req AFTER UPDATE OF JON,
    EMP_ID_CODE,
    DOC_#
    PROJ_COST,
    ACTUAL_COST,
    ADP_PROJ_
ON OPTAR_REQ
REFERENCING NEW AS new_upd OLD AS old_upd FOR EACH ROW
BEGIN
    DECLARE user_defined_exception EXCEPTION FOR SQLSTATE '99999';
    DECLARE found INTEGER;

    CALL CALC_BAL_OPTAR(new_upd.JON);
END /

% After delete trigger "tda_optar_req" for table "OPTAR_REQ"
cREATE TRIGGER tda_optar_req AFTER DELETE ON OPTAR_REQ
REFERENCING OLD AS old_del FOR EACH ROW
BEGIN
    DECLARE user_defined_exception EXCEPTION FOR SQLSTATE '99999';
    DECLARE found INTEGER;

    CALL CALC_BAL_OPTAR(old_del.JON);
END /

% Before insert trigger "tib_other_leave" for table "OTHER_LEAVE"
cREATE TRIGGER tib_other_leave BEFORE INSERT ON OTHER_LEAVE
REFERENCING NEW AS new_ins FOR EACH ROW
BEGIN
    DECLARE user_defined_exception EXCEPTION FOR SQLSTATE '99999';
    DECLARE found INTEGER;

    % Parent "LABOR_LES" must exist when inserting a child in "OTHER_LEAVE"
    IF (new_ins.EMP_ID_CODE IS NOT NULL AND
    new_ins.PPE_DATE IS NOT NULL) THEN
    BEGIN
        SET found = 0;
        SELECT 1
        INTO found
        FROM dummy
        WHERE EXISTS (SELECT 1
            FROM LABOR_LES
            WHERE EMP_ID_CODE = new_ins.EMP_ID_CODE
            AND PPE_DATE = new_ins.PPE_DATE);
    END
if found <> 1 then
    signal user_defined_exception
end if;
end
end if;

% Parent "OTHER_LV_TYPE" must exist when inserting a child in
"OTHER_LEAVE"
if (new_ins.TYPE is not null) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
        from OTHER_LV_TYPE
        where OTHER_LV_TYPE_CODE = new_ins.TYPE);
    if found <> 1 then
        signal user_defined_exception
    end if;
end
end if;
end
/

% Before update trigger "tub_other_leave" for table "OTHER_LEAVE"
create trigger tub_other_leave before update of EMP_ID_CODE,
        PPE_DATE,
        TYPE
on OTHER_LEAVE
referencing new as new_upd old as old_upd for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "LABOR_LES" must exist when updating a child in
"OTHER_LEAVE"
if (new_upd.EMP_ID_CODE is not null and
    new_upd.PPE_DATE is not null and
    ((old_upd.EMP_ID_CODE is null and
        old_upd.PPE_DATE is null) or
    (new_upd.EMP_ID_CODE <> old_upd.EMP_ID_CODE or
        new_upd.PPE_DATE <> old_upd.PPE_DATE))) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
        from LABOR_LES
        where EMP_ID_CODE = new_upd.EMP_ID_CODE
            and PPE_DATE = new_upd.PPE_DATE);
end

if found <> 1 then
    signal user_defined_exception
end if;
end if;

% Parent "OTHER LV_TYPE" must exist when updating a child in "OTHER_LEAVE"
if (new_upd.TYPE is not null and
    ((old_upd.TYPE is null) or
    (new_upd.TYPE <> old_upd.TYPE))) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
        from OTHER LV_TYPE
        where OTHER LV_TYPE_CODE = new_upd.TYPE);
    if found <> 1 then
        signal user_defined_exception
    end if;
end if;

% Cannot modify parent code of "OTHER LV_TYPE" in child "OTHER_LEAVE"
if ((new_upd.TYPE is null and old_upd.TYPE is not null) or
    new_upd.TYPE <> old_upd.TYPE ) then
    signal user_defined_exception
end if;
/

% Before insert trigger "tib_pi" for table "PI"
create trigger tib_pi before insert on PI
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "EMPLOYEE" must exist when inserting a child in "PI"
    if (new_ins.EMP_ID_CODE is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                from EMPLOYEE
                where EMP_ID_CODE = new_ins.EMP_ID_CODE);
            if found <> 1 then
                signal user_defined_exception
            end if;
        end
    end if;
end if;
end if;
end if;

% Parent "ACCOUNT" must exist when inserting a child in "PI"
if (new_ins.JON is not null) then
begin
set found = 0;
select 1
into found
from dummy
where exists (select 1
    from ACCOUNT
    where JON = new_ins.JON);
if found <> 1 then
    signal user_defined_exception
end if;
end
end if;
end
/

% Before insert trigger "tib_salary_history" for table "SALARY_HISTORY"
create trigger tib_salary_history before insert on SALARY_HISTORY
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
    declare found integer;

    % Parent "EMPLOYEE" must exist when inserting a child in
    "SALARY_HISTORY"
    if (new_ins.EMP_ID_CODE is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                from EMPLOYEE
                where EMP_ID_CODE = new_ins.EMP_ID_CODE);
            if found <> 1 then
                signal user_defined_exception
            end if;
        end
        end if;
    end
/

% Before insert trigger "tib_staff" for table "STAFF"
create trigger tib_staff before insert on STAFF
referencing new as new_ins for each row
begin
declare user_defined_exception exception for SQLSTATE '99999';
declare found integer;

% Parent "EMPLOYEE" must exist when inserting a child in "STAFF"
if (new_ins.EMP_ID_CODE is not null) then
begin
    set found = 0;
    select 1
    into found
    from dummy
    where exists (select 1
                  from EMPLOYEE
                  where EMP_ID_CODE = new_ins.EMP_ID_CODE);
    if found <> 1 then
        signal user_defined_exception
    end if;
end
end if;
end
/

% Before insert trigger "tib_travel" for table "TRAVEL"
create trigger tib_travel before insert on TRAVEL
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';
declare found integer;

    % Parent "ACCOUNT" must exist when inserting a child in "TRAVEL"
    if (new_ins.JON is not null) then
        begin
            set found = 0;
            select 1
            into found
            from dummy
            where exists (select 1
                           from ACCOUNT
                           where JON = new_ins.JON);
            if found <> 1 then
                signal user_defined_exception
            end if;
        end
    end
end if;
end
/

% After insert trigger "tia_travel" for table "TRAVEL"
create trigger tia_travel after insert on TRAVEL
referencing new as new_ins for each row
begin
    declare user_defined_exception exception for SQLSTATE '99999';

    end
call CALC_BAL_TRAV(new_ins.JON);
end
/

% Update trigger "tua_travel" for table "TRAVEL"
cREATE TRIGGER tua_travel
ON TRAVEL
REFERENCING NEW AS new_upd OLD AS old_upd
BEGIN
  DECLARE user_defined_exception exception FOR SQLSTATE '99999';
  DECLARE found integer;
  CALL CALC_BAL_TRAV(new_upd.JON);
END
/

% After delete trigger "tda_travel" for table "TRAVEL"
cREATE TRIGGER tda_travel
ON TRAVEL
REFERENCING OLD AS old_del
BEGIN
  DECLARE user_defined_exception exception FOR SQLSTATE '99999';
  DECLARE found integer;
  CALL CALC_BAL_TRAV(old_del.JON);
END
/

% Before insert trigger "tib_travel_requests" for table "TRAVEL_REQUESTS"
cREATE TRIGGER tib_travel_requests
ON TRAVEL_REQUESTS
REFERENCING NEW AS new_ins
BEGIN
  DECLARE user_defined_exception exception FOR SQLSTATE '99999';
  DECLARE found integer;

  % Parent "TRAVEL" must exist when inserting a child in "TRAVEL_REQUESTS"
  IF (new_ins.TO# IS NOT NULL) THEN
    BEGIN
      SET found = 0;
      SELECT 1
      INTO found
      FROM dummy
      WHERE EXISTS (SELECT 1
                    FROM TRAVEL
where TO# = new_ins.TO#)
    if found <> 1 then
        signal user_defined_exception
    end if;
    end if;
end
/


APPENDIX C. FMS DATABASE STORED PROCEDURES

%********************************************************************************************%
% Procedure CALC_BAL_CONTRACT
%********************************************************************************************%
create procedure %PROC% (IN jo_num char(5), cont_type char(1))
begin
    declare current_fy_end date;
    declare sum_actual numeric(12,2);
    declare sum_proj numeric(12,2);
    declare sum_cont numeric(12,2);
    declare begin_date date;

    select CURRENT_FY_END_DATE into current_fy_end from FMS_CFG;

    select DATE_RECEIVED into begin_date from ACCOUNT
    where ACCOUNT.JON=jo_num;

    select sum(ACTUAL_COST) into sum_actual from CONTRACTS
    where CONTRACTS.JON = jo_num
    and CONTRACTS.CONTRACT_TYPE = cont_type
    and CONTRACTS.FY_ENDING >= begin_date
    and CONTRACTS.FY_ENDING <= current_fy_end;

    if (sum_actual is null) then
        set sum_actual = 0.00
    end if;

    select sum(PROJ_COST) into sum_proj from CONTRACTS
    where CONTRACTS.JON = jo_num
    and CONTRACTS.CONTRACT_TYPE = cont_type
    and CONTRACTS.ACTUAL_COST is null
    and CONTRACTS.FY_ENDING >= begin_date
    and CONTRACTS.FY_ENDING <= current_fy_end;

    if (sum_proj is null) then
        set sum_proj = 0.00
    end if;

    set sum_cont = sum_actual + sum_proj;
if (cont_type = 'M') then
  update ACCOUNT
  set BAL_CONT_MIPR = INIT_CONT_MIPR - sum_cont
  where ACCOUNT.JON = jo_num
else
  if (cont_type = 'T') then
    update ACCOUNT
    set BAL_CONT_IPA = INIT_CONT_IPA - sum_cont
    where ACCOUNT.JON = jo_num
  else
    if (cont_type = 'O') then
      update ACCOUNT
      set BAL_CONT_OTH = INIT_CONT_OTH - sum_cont
      where ACCOUNT.JON = jo_num
    end if
  end if
end if;
end
/

%***************************************************************************%

%***************************************************************************%
% Procedure CALC_BAL_FAC_LABOR
%***************************************************************************%
create procedure %PROC% (IN jo_num char(5))
begin
  declare current_fy_end date;
  declare begin_date date;
  declare sum_chg numeric(12,2);

  select CURRENT_FY_END_DATE into current_fy_end from FMS_CFG;

  select DATE_RECEIVED into begin_date from ACCOUNT
  where ACCOUNT.JON=jo_num;

  select sum(TOTAL_CHG) into sum_chg from LABOR_CHGS, FACULTY
  where FACULTY.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
  and LABOR_CHGS.JON = jo_num
  and LABOR_CHGS.FY_ENDING >= begin_date

  72
and LABOR_CHGS.FY_ENDING <= current_fy_end;

if (sum_chg is null) then
    set sum_chg = 0.00
end if;

update ACCOUNT
    set BAL_FAC_LABOR = INIT_FAC_LABOR_ - sum_chg
        where ACCOUNT.JON = jo_num;
end
/
%***********************************************************************%

%***********************************************************************%
% Procedure CALC_BAL_OPTAR
%***********************************************************************%
create procedure %PROC% (IN jo_num char(5))
begin
    declare current_fy_end date;
    declare sum_actual numeric(12,2);
    declare sum_proj numeric(12,2);
    declare sum_optar numeric(12,2);
    declare begin_date date;

    select CURRENT_FY_END_DATE into current_fy_end from FMS_CFG;

    select DATE_RECEIVED into begin_date from ACCOUNT
        where ACCOUNT.JON = jo_num;

    select sum(ACTUAL_COST) into sum_actual from OPTAR_REQ
        where OPTAR_REQ.JON = jo_num
        and OPTAR_REQ.FY_ENDING >= begin_date
        and OPTAR_REQ.FY_ENDING <= current_fy_end;

    if (sum_actual is null) then
        set sum_actual = 0.00
    end if;

    select sum(PROJ_COST) into sum_proj from OPTAR_REQ
        where OPTAR_REQ.JON = jo_num

73
and OPTAR_REQ.ACTUAL_COST is null
and OPTAR_REQ.FY_ENDING >= begin_date
and OPTAR_REQ.FY_ENDING <= current_fy_end;

if (sum_proj is null) then
    set sum_proj = 0.00
end if;

set sum_optar = sum_actual + sum_proj;

update ACCOUNT
    set BAL_OPTAR = INIT_OPTAR_ $ - sum_optar
    where ACCOUNT.JON = jo_num;
end /
%

%******************************************************************************%
% Procedure CALC_BAL_SPT_LABOR
%******************************************************************************%
create procedure %PROC% (IN jo_num char(5))
begin
   declare current_fy_end date;
   declare begin_date date;
   declare sum_chg numeric(12,2);

   select CURRENT_FY_END_DATE into current_fy_end from FMS_CFG;

   select DATE_RECEIVED into begin_date from ACCOUNT
       where ACCOUNT.JON=jo_num;

   select sum(TOTAL_CHG) into sum_chg from LABOR_CHGS, STAFF
       where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
       and LABOR_CHGS.JON = jo_num
       and LABOR_CHGS.FY_ENDING >= begin_date
       and LABOR_CHGS.FY_ENDING <= current_fy_end;

   if (sum_chg is null) then
       set sum_chg = 0.00
   end if;

74
update ACCOUNT
    set BAL_SPT_LABOR = INIT_SPT_LABOR_$ - sum_chg
    where ACCOUNT.JON = jo_num;
end /

%******************************************************************************%

%******************************************************************************%
% Procedure CALC_BAL_TRAV
%******************************************************************************%
create procedure %PROC% (IN jo_num char(5))
begin
    declare current_fy_end date;
    declare sum_actual numeric(12,2);
    declare sum_proj numeric(12,2);
    declare sum_trav numeric(12,2);
    declare begin_date date;

    select CURRENT_FY_END_DATE into current_fy_end from FMS_CFG;

    select DATE_RECEIVED into begin_date from ACCOUNT
        where ACCOUNT.JON=jo_num;

    select sum(ACTUAL_COST) into sum_actual from TRAVEL
        where TRAVEL.JON = jo_num
        and TRAVEL.FY_ENDING >= begin_date
        and TRAVEL.FY_ENDING <= current_fy_end;

    if (sum_actual is null) then
        set sum_actual = 0.00
    end if;

    select sum(PROJ_COST) into sum_proj from TRAVEL
        where TRAVEL.JON = jo_num
        and TRAVEL.ACTUAL_COST is null
        and TRAVEL.FY_ENDING >= begin_date
        and TRAVEL.FY_ENDING <= current_fy_end;

    if (sum_proj is null) then
        set sum_proj = 0.00
end if;

set sum_trav = sum_actual + sum_proj;

update ACCOUNT
set BAL_TRAVEL = INIT_TRAVEL_$ - sum_trav
where ACCOUNT.JON = jo_num;
end
/
%*******************************************************************************
%
APPENDIX D. FMS POWERBUILDER LIBRARY OBJECT LISTING

The FMS PowerBuilder library object listing is shown on the next page.
APPENDIX E. FMS APPMODELER REPORT

The partial AppModeler report produced from the FMS physical data model begins on the next page.
Full PDM report
Physical Data Model

---

**Model Information**

<table>
<thead>
<tr>
<th>Project Name:</th>
<th>fms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Code:</td>
<td>FMS</td>
</tr>
<tr>
<td>Database:</td>
<td>Watcom SQL 4.0</td>
</tr>
<tr>
<td>Name:</td>
<td>fms</td>
</tr>
<tr>
<td>Code:</td>
<td>FMS</td>
</tr>
<tr>
<td>Label:</td>
<td>Ops Research Dept Financial Management System</td>
</tr>
<tr>
<td>Author:</td>
<td>Alan E. Pires</td>
</tr>
<tr>
<td>Version:</td>
<td>1.01</td>
</tr>
<tr>
<td>Created On:</td>
<td>11/30/95 8:01 AM</td>
</tr>
<tr>
<td>Modified On:</td>
<td>2/3/97 4:51 PM</td>
</tr>
</tbody>
</table>

**Model Description**

Financial Management System for the Operations Research Department

**Begin Script**

**End Script**

**Business Rules**

**Domains**

**Tables**

**Table List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>ACCOUNT</td>
<td>0</td>
</tr>
<tr>
<td>adp_proj_info</td>
<td>ADP_PROJ_INFO</td>
<td>0</td>
</tr>
<tr>
<td>contracts</td>
<td>CONTRACTS</td>
<td>0</td>
</tr>
<tr>
<td>department</td>
<td>DEPARTMENT</td>
<td>0</td>
</tr>
</tbody>
</table>
### Physical Data Model

<table>
<thead>
<tr>
<th>Table</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>employee</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td>faculty</td>
<td>FACULTY</td>
</tr>
<tr>
<td>fms_cfg</td>
<td>FMS_CFG</td>
</tr>
<tr>
<td>labor_chgs</td>
<td>LABOR_CHGS</td>
</tr>
<tr>
<td>labor_les</td>
<td>LABOR_LES</td>
</tr>
<tr>
<td>military</td>
<td>MILITARY</td>
</tr>
<tr>
<td>optar_req</td>
<td>OPTAR_REQ</td>
</tr>
<tr>
<td>other_leave</td>
<td>OTHER_LEAVE</td>
</tr>
<tr>
<td>other_lv_type</td>
<td>OTHER_LV_TYPE</td>
</tr>
<tr>
<td>pi</td>
<td>PI</td>
</tr>
<tr>
<td>salary_history</td>
<td>SALARY_HISTORY</td>
</tr>
<tr>
<td>sponsor</td>
<td>SPONSOR</td>
</tr>
<tr>
<td>staff</td>
<td>STAFF</td>
</tr>
<tr>
<td>travel</td>
<td>TRAVEL</td>
</tr>
<tr>
<td>travel_requests</td>
<td>TRAVEL_REQUESTS</td>
</tr>
</tbody>
</table>

### Table account

- **Name**: account
- **Code**: ACCOUNT
- **Label**: Account Information
- **Number**: 
- **PK constraint**: 

### Options

### Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>bal_cont_ipa</td>
<td>BAL_CONT_IPA</td>
<td>decimal(12 ,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bal_cont_mipr</td>
<td>BAL_CONT_MIPR</td>
<td>decimal(12 ,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bal_cont_oth</td>
<td>BAL_CONT_OTH</td>
<td>decimal(12 ,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bal_fac_labor</td>
<td>BAL_FAC_LABOR</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------</td>
<td>---------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>bal_optar</td>
<td>BAL_OPTAR</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bal_spt_labor</td>
<td>BAL_SPT_LABOR</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bal_travel</td>
<td>BAL_TRAVEL</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>budget_page_date</td>
<td>BUDGET_PAGE_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>date_received</td>
<td>DATE_RECEIVED</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>expir_date</td>
<td>EXPIR_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>fund_type</td>
<td>FUND_TYPE</td>
<td>char(2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>indirect_cost</td>
<td>INDIRECT_COST</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_cont_ipa</td>
<td>INIT_CONT_IPA</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_cont_mipr</td>
<td>INIT_CONT_MIPR</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_cont_oth</td>
<td>INIT_CONT_OTH</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_fac_labor_($)</td>
<td>INIT_FAC_LABOR_$</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_optar_($)</td>
<td>INIT_OPTAR_$</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_spt_labor_($)</td>
<td>INIT_SPT_LABOR_$</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>init_travel_($)</td>
<td>INIT_TRAVEL_$</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>jon</td>
<td>JON</td>
<td>char(5)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>labor_jon</td>
<td>LABOR_JON</td>
<td>char(5)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>remarks</td>
<td>REMARKS</td>
<td>char(100)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>segment_#s</td>
<td>SEGMENT_#S</td>
<td>char(9)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>serial_#s</td>
<td>SERIAL_#S</td>
<td>char(11)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>spon_id_code</td>
<td>SPON_ID_CODE</td>
<td>char(6)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>title</td>
<td>TITLE</td>
<td>char(40)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**BAL_CONT_IPA**

**Check**
Physical Data Model

| **Domain:** |  |
| **Low value:** |  |
| **High value:** |  |
| **Default value:** |  |
| **Unit:** |  |
| **Format:** |  |
| **Uppercase:** | No | **Lowercase:** | No | **Can't modify:** | No |
| **List of values:** |  |

**BAL_CONT_MIPR**

**Check**

| **Domain:** |  |
| **Low value:** |  |
| **High value:** |  |
| **Default value:** |  |
| **Unit:** |  |
| **Format:** |  |
| **Uppercase:** | No | **Lowercase:** | No | **Can't modify:** | No |
| **List of values:** |  |

**BAL_CONT_OTH**

**Check**

| **Domain:** |  |
| **Low value:** |  |
| **High value:** |  |
| **Default value:** |  |
| **Unit:** |  |
| **Format:** |  |
| **Uppercase:** | No | **Lowercase:** | No | **Can't modify:** | No |
| **List of values:** |  |
### BAL_FAC_LABOR

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
</tbody>
</table>

### BAL_OPTAR

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
</tbody>
</table>

### BAL_SPT_LABOR

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
</tbody>
</table>
Physical Data Model

<table>
<thead>
<tr>
<th>Uppercase:</th>
<th>No</th>
<th>Lowercase:</th>
<th>No</th>
<th>Can't modify:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BAL_TRAVEL

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
<th>Low value:</th>
<th></th>
<th>High value:</th>
<th></th>
<th>Default value:</th>
<th></th>
<th>Unit:</th>
<th></th>
<th>Format:</th>
<th></th>
<th>Uppercase:</th>
<th>No</th>
<th>Lowercase:</th>
<th>No</th>
<th>Can't modify:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BUDGET_PAGE_DATE

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
<th>Low value:</th>
<th></th>
<th>High value:</th>
<th></th>
<th>Default value:</th>
<th></th>
<th>Unit:</th>
<th></th>
<th>Format:</th>
<th></th>
<th>Uppercase:</th>
<th>No</th>
<th>Lowercase:</th>
<th>No</th>
<th>Can't modify:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DATE_RECEIVED

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
<th>Low value:</th>
<th></th>
</tr>
</thead>
</table>

S-Designer March 11, 1997 Page 86
<table>
<thead>
<tr>
<th>Physical Data Model</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>Uppercase: No</th>
<th>Lowercase: No</th>
<th>Can't modify: No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of values:</th>
</tr>
</thead>
</table>

**EXPIR_DATE**

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uppercase: No</th>
<th>Lowercase: No</th>
<th>Can't modify: No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of values:</th>
</tr>
</thead>
</table>

**FUND_TYPE**

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uppercase: No</th>
<th>Lowercase: No</th>
<th>Can't modify: No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of values:</th>
</tr>
</thead>
</table>

**INDIRECT_COST**
Check

| Domain: | Low value: 0.00 | High value: | Default value: 0.00 | Unit: | Format: | Uppercase: No | Lowercase: No | Can't modify: No | List of values: |

INIT_CONT_IPA

Check

| Domain: | Low value: 0 | High value: | Default value: 0 | Unit: | Format: | Uppercase: No | Lowercase: No | Can't modify: No | List of values: |

INIT_CONT_MIPR

Check

| Domain: | Low value: 0 | High value: | Default value: 0 | Unit: | Format: | Uppercase: No | Lowercase: No | Can't modify: No | List of values: |
### INIT_CONT_OTH

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### INIT_FAC_LABOR_

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

### INIT_OPTAR_

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INIT_SPT_LABOR_

Check

Domain:
Low value: 0.00
High value:
Default value: 0.00
Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

INIT_TRAVEL_

Check

Domain:
Low value: 0.00
High value:
Default value: 0.00
Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

JON

Check

Domain:
### LABOR_JON

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### REMARKS

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### SEGMENT_##S
### Physical Data Model

#### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**SERIAL_#S**

#### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**SPON_ID_CODE**

#### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
### TITLE

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

| List of values: | |
|-----------------| |

### Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT_FK1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>SPON_ID_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td>ACCOUNT_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>JON</td>
<td>ASC</td>
</tr>
</tbody>
</table>

### Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPONSOR</td>
<td>SPON_ID_CODE</td>
<td>SPON_ID_CODE</td>
</tr>
</tbody>
</table>

### Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>LABOR_CHGS</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>OPTAR_REQ</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>PI</td>
<td>JON</td>
<td>JON</td>
</tr>
</tbody>
</table>

### Table adp_proj_info
Name: adp_proj_info
Code: ADP_PROJ_INFO
Label: ADP Project Information
Number:
PK constraint:

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>adp_proj_#</td>
<td>ADP_PROJ_#</td>
<td>char(7)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>dept_code</td>
<td>DEPT_CODE</td>
<td>char(2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>fy-ending</td>
<td>FY_ENDING</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>poc_code</td>
<td>POC_CODE</td>
<td>char(4)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>proj_cost_auth</td>
<td>PROJ_COST_AUTH</td>
<td>decimal(12,.2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>proj_mgr_code</td>
<td>PROJ_MGR_CODE</td>
<td>char(4)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>proj_name</td>
<td>PROJ_NAME</td>
<td>char(40)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

ADP_PROJ_#

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

DEPT_CODE
Physical Data Model

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

FY_ENDING

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td>09/30/97</td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

POC_CODE

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

S-Designer March 11, 1997 Page 95
### PROJ_COST_AUTH

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PROJ_MGR_CODE

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PROJ_NAME

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADP_PROJ_INFO_FK1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>DEPT_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td>ADP_PROJ_INFO_FK2</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>PROJ_MGR_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td>ADP_PROJ_INFO_FK3</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>POC_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td>ADP_PROJ_INFO_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>ADP_PROJ_#</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT</td>
<td>DEPT_CODE</td>
<td>DEPT_CODE</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>PROJ_MGR_CODE</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>POC_CODE</td>
</tr>
</tbody>
</table>

Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTAR_REQ</td>
<td>ADP_PROJ_#</td>
<td>ADP_PROJ_#</td>
</tr>
</tbody>
</table>

Table contracts

Name: contracts
Code: CONTRACTS
Label: Departmental Contracts (charged to departmental accounts)
Number: 
PK constraint: 

Options
Description

Departmental Contracts (charged to departmental accounts)

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual_cost</td>
<td>ACTUAL_COST</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>contract_type</td>
<td>CONTRACT_TYPE</td>
<td>char(1)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>contractor</td>
<td>CONTRACTOR</td>
<td>char(20)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>delivery_date</td>
<td>DELIVERY_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>description</td>
<td>DESCRIPTION</td>
<td>char(50)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>doc_#</td>
<td>DOC_#</td>
<td>char(9)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>fy-ending</td>
<td>FYENDING</td>
<td>date</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>jon</td>
<td>JON</td>
<td>char(5)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>order_date</td>
<td>ORDER_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>po_#</td>
<td>PO_DATE</td>
<td>char(12)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>po_date</td>
<td>PROJ_COST</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>proj_cost</td>
<td></td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>requester</td>
<td>REQUESTER</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**ACTUAL_COST**

Check

- Domain:
- Low value: 
- High value: 
- Default value: 
- Unit: 
- Format: 
- Uppercase: No 
- Lowercase: No 
- Can't modify: No 
- List of values:

**CONTRACT_TYPE**
Physical Data Model

Check

**CONTRACTOR**

Check

**DELIVERY_DATE**

Check
DESCRIPTION

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  
Lowercase:  No  
Can't modify:  No  
List of values:  

DOC_#

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  
Lowercase:  No  
Can't modify:  No  
List of values:  

FY_ENDING

Check

Domain:  
Low value:  
High value:  
Default value:  9/30/97  
Unit:  

S-Designer  March 11, 1997  Page 100
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

JON

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

ORDER_DATE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

PO_

Check

Domain:
PO_DATE

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  
Lowercase:  No  
Can't modify:  No

List of values:  

PROJ_COST

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  
Lowercase:  No  
Can't modify:  No

List of values:  

REQUESTER

S-Designer  March 11, 1997  Page 102
Physical Data Model

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No 
Lowercase: No 
Can't modify: No 
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONTRACTS_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>JON CONTRACT_TYPE REQUESTER DOC_#</td>
<td>ASC ASC ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>REQUESTER</td>
</tr>
</tbody>
</table>

Table department

| Name: | department |
| Code: | DEPARTMENT |
| Label: | Department Info |
| Number: | |
| PK constraint: | |

Options
Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>chair_code</td>
<td>CHAIR_CODE</td>
<td>char(4)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>dept_code</td>
<td>DEPT_CODE</td>
<td>char(2)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>dept_name</td>
<td>DEPT_NAME</td>
<td>char(40)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**CHAIR_CODE**

Check

- Domain:
- Low value:
- High value:
- Default value:
- Unit:
- Format:
- Uppercase: No
- Lowercase: No
- Can't modify: No
- List of values:

**DEPT_CODE**

Check

- Domain:
- Low value:
- High value:
- Default value:
- Unit:
- Format:
- Uppercase: No
- Lowercase: No
- Can't modify: No
- List of values:

**DEPT_NAME**
### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
</table>

List of values:

### Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>DEPT_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

### Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>DEPT_CODE</td>
<td>DEPT_CODE</td>
</tr>
<tr>
<td>ADP_PROJ_INFO</td>
<td>DEPT_CODE</td>
<td>DEPT_CODE</td>
</tr>
</tbody>
</table>

### Table employee

<table>
<thead>
<tr>
<th>Name:</th>
<th>employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td>Label:</td>
<td>Employee Information</td>
</tr>
<tr>
<td>Number:</td>
<td></td>
</tr>
<tr>
<td>PK constraint:</td>
<td></td>
</tr>
</tbody>
</table>

### Options

### Column List
## Physical Data Model

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>accel_rate</td>
<td>ACCEL_RATE</td>
<td>decimal(3, 2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>base_salary</td>
<td>BASE_SALARY</td>
<td>decimal(10, 2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bldg_#</td>
<td>BLDG_#</td>
<td>char(3)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>category</td>
<td>CATEGORY</td>
<td>char</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>city</td>
<td>CITY</td>
<td>char(15)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>dept_code</td>
<td>DEPT_CODE</td>
<td>char(2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>eff_sal_date</td>
<td>EFF_SAL_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>emp_code</td>
<td>EMP_CODE</td>
<td>char(2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>first_name</td>
<td>FIRST_NAME</td>
<td>char(15)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>home_phone</td>
<td>HOME_PHONE</td>
<td>char(13)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>last_name</td>
<td>LAST_NAME</td>
<td>char(15)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>mi</td>
<td>MI</td>
<td>char(1)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>room_#</td>
<td>ROOM_#</td>
<td>char(5)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>spouse_fname</td>
<td>SPOUSE_FNAME</td>
<td>char(15)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ssn</td>
<td>SSN</td>
<td>char(11)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>state</td>
<td>STATE</td>
<td>char(2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>street_address</td>
<td>STREET_ADDRESS</td>
<td>char(20)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>term_date</td>
<td>TERM_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>work_phone</td>
<td>WORK_PHONE</td>
<td>char(13)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>zipcode</td>
<td>ZIPCODE</td>
<td>char(10)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

### ACCEL_RATE

#### Check

- **Domain:**
- **Low value:**
- **High value:**
- **Default value:** 1.43
- **Unit:**
- **Format:**
- **Uppercase:** No  
  **Lowercase:** No  
  **Can't modify:** No
- **List of values:**

---

S-Designer  
March 11, 1997  
Page 106  
106
### BASE_SALARY

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

List of values:

### BLDG_#

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

List of values:

### CATEGORY

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Data Model</td>
<td>fms</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Uppercase: | No |
| Lowercase: | No |
| Can't modify: | No |

| List of values: |

**CITY**

**Check**

| Domain: |
| Low value: |
| High value: |
| Default value: |
| Unit: |
| Format: |
| Uppercase: | No |
| Lowercase: | No |
| Can't modify: | No |

| List of values: |

**DEPT_CODE**

**Check**

| Domain: |
| Low value: |
| High value: |
| Default value: |
| Unit: |
| Format: |
| Uppercase: | No |
| Lowercase: | No |
| Can't modify: | No |

| List of values: |

**EFF_SAL_DATE**

**Check**

| Domain: |
| Low value: |
Physical Data Model

| High value: |
| Default value: | 10/01/95 |
| Unit: |
| Format: |
| Uppercase: | No |
| Lowercase: | No |
| Can't modify: | No |
| List of values: |

EMP_CODE

Check

| Domain: |
| Low value: |
| High value: |
| Default value: |
| Unit: |
| Format: |
| Uppercase: | No |
| Lowercase: | No |
| Can't modify: | No |
| List of values: |

EMP_ID_CODE

Check

| Domain: |
| Low value: |
| High value: |
| Default value: |
| Unit: |
| Format: |
| Uppercase: | No |
| Lowercase: | No |
| Can't modify: | No |
| List of values: |

FIRST_NAME
### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase: No</th>
<th>Lowercase: No</th>
<th>Can't modify: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### HOME_PHONE

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase: No</th>
<th>Lowercase: No</th>
<th>Can't modify: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### LAST_NAME

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase: No</th>
<th>Lowercase: No</th>
<th>Can't modify: No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MI

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ROOM_#

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SPOUSE_FNAME

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SSN

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Upper case: No  
Lower case: No  
Can't modify: No  
List of values:  

STATE

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Upper case: No  
Lower case: No  
Can't modify: No  
List of values:  

STREET_ADDRESS

Check

Domain:
Physical Data Model

Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase: No  Lowercase: No  Can't modify: No  
List of values:

TERM_DATE

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase: No  Lowercase: No  Can't modify: No  
List of values:

WORK_PHONE

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase: No  Lowercase: No  Can't modify: No  
List of values:

ZIPCODE

S-Designer  March 11, 1997  Page 113
Physical Data Model

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
</table>

List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE_FK1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>DEPT_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td>EMPLOYEE_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT</td>
<td>DEPT_CODE</td>
<td>DEPT_CODE</td>
</tr>
</tbody>
</table>

Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABOR_CHGS</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>OPTAR_REQ</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>SALARY_HISTORY</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>PI</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>EMP_ID_CODE</td>
<td>REQUESTER</td>
</tr>
<tr>
<td>LABOR_LES</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>FACULTY</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>STAFF</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>MILITARY</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>ADP_PROJ_INFO</td>
<td>EMP_ID_CODE</td>
<td>PROJ_MGR_CODE</td>
</tr>
<tr>
<td>ADP_PROJ_INFO</td>
<td>EMP_ID_CODE</td>
<td>POC_CODE</td>
</tr>
</tbody>
</table>

S-Designer          March 11, 1997

Page 114
Table faculty

<table>
<thead>
<tr>
<th>Name</th>
<th>faculty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>FACULTY</td>
</tr>
<tr>
<td>Label</td>
<td>Faculty Specialization of Employee Table</td>
</tr>
<tr>
<td>Number</td>
<td></td>
</tr>
<tr>
<td>PK constraint</td>
<td></td>
</tr>
</tbody>
</table>

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>civ_grade</td>
<td>CIV_GRADE</td>
<td>char(5)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>step</td>
<td>STEP</td>
<td>char(2)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

CIV_GRADE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No
Lowercase: No
Can't modify: No
List of values:

EMP_ID_CODE

Check
Physical Data Model

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

STEP

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>FACULTY_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
</tbody>
</table>

Table fms_cfg

<table>
<thead>
<tr>
<th>Name:</th>
<th>fms_cfg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>FMS_CFG</td>
</tr>
</tbody>
</table>

S-Designor March 11, 1997 Page 116
Label: FMS Configuration Info
Number: 
PK constraint: 

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>current_fy_end_date</td>
<td>CURRENT_FY_END_DATE</td>
<td>date</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ot_cap</td>
<td>OT_CAP</td>
<td>decimal(10,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>rr_ot_rate_fact</td>
<td>RR_OT_RATE_FACT</td>
<td>decimal(6,4)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>yr_labor_hrs</td>
<td>YR_LABOR_HRS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

CURRENT_FY_END_DATE

Check

Domain: 
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No 
Lowercase: No 
Can't modify: No 
List of values: 

OT_CAP

Check

Domain: 
Low value: 

S-Designer March 11, 1997 Page 117
### Physical Data Model

<table>
<thead>
<tr>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of values:</th>
</tr>
</thead>
</table>

#### RR_OT_RATE_FACT

##### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of values:</th>
</tr>
</thead>
</table>

#### YR_LABOR_HRS

##### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List of values:</th>
</tr>
</thead>
</table>

#### Index List

S-Designer  March 11, 1997  Page 118
Physical Data Model

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMS_CFG_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>CURRENT_FY_END_DATE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

**Table labor_chgs**

Name: labor_chgs  
Code: LABOR_CHGS  
Label: Labor charges made against accounts  
Number:  
PK constraint:

**Options**

**Description**

This table contains the labor charges made against accounts by pay period ending date and employee.

**Column List**

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>fy-ending</td>
<td>FY_ENDING</td>
<td>date</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>hours</td>
<td>HOURS</td>
<td>integer</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>jon</td>
<td>JON</td>
<td>char(5)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ot_hours</td>
<td>OT_HOURS</td>
<td>integer</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>ppe_date</td>
<td>PPE_DATE</td>
<td>date</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>total_chg</td>
<td>TOTAL_CHG</td>
<td>decimal(12,2)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**EMP_ID_CODE**

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
</tr>
</thead>
</table>

S-Designor March 11, 1997
### FY_ENDING

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td>9/30/97</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### HOURS

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### JON

**Check**

S-Designor: March 11, 1997

Page 120
Domain:
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No  
Lowercase: No  
Can't modify: No 
List of values: 

OT_HOURS

Check

Domain:
Low value: 0 
High value: 0 
Default value: 
Unit: 
Format: 
Uppercase: No  
Lowercase: No  
Can't modify: No 
List of values: 

PPE_DATE

Check

Domain:
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No  
Lowercase: No  
Can't modify: No 
List of values: 

TOTAL_CHG

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  
Lowercase:  No  
Can't modify:  No  
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABOR_CHGS_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PPE_DATE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JON</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABOR_LES</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td></td>
<td>PPE_DATE</td>
<td>PPE_DATE</td>
</tr>
<tr>
<td>ACCOUNT</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
</tbody>
</table>

Table labor_les

Name: labor_les
Code: LABOR_LES
Label: Labor -- Leave and Holiday Charges
Number: 
PK constraint: 

S-Designor  March 11, 1997  Page 122
Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>al_hours</td>
<td>AL_HOURS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>hol_hours</td>
<td>HOL_HOURS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>lwop_hours</td>
<td>LWOP_HOURS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ppe_date</td>
<td>PPE_DATE</td>
<td>date</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>sl_hours</td>
<td>SL_HOURS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**AL_HOURS**

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No  List of values:

**EMP_ID_CODE**

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
HOL_HOURS

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No
Lowercase: No
Can't modify: No
List of values:

LWOP_HOURS

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No
Lowercase: No
Can't modify: No
List of values:

PPE_DATE

Check

Domain:
Low value:
Physical Data Model

High value:  
Default value:  
Unit:  
Format:  
Uppercase: No  
Lowercase: No  
Can't modify: No  
List of values:

SL_HOURS

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase: No  
Lowercase: No  
Can't modify: No  
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABOR_LES_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PPE_DATE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
</tbody>
</table>

Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABOR_CHGS</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td></td>
<td>PPE_DATE</td>
<td>PPE_DATE</td>
</tr>
</tbody>
</table>

S-Designer March 11, 1997 Page 125
Table military

Name: military
Code: MILITARY
Label: Military Specialization of Employee Table
Number: 
PK constraint:

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>mil_grade</td>
<td>MIL GRADE</td>
<td>char(5)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>service</td>
<td>SERVICE</td>
<td>char(4)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

EMP_ID_CODE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

MIL_GRADE

S-Designer  March 11, 1997  Page 126
Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No   Lowercase: No   Can't modify: No
List of values:

SERVICE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No   Lowercase: No   Can't modify: No
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>MILITARY_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
</tbody>
</table>

Table optar_req
Name: optar_req  
Code: OPTAR_REQ  
Label: OPTAR Request Information  
Number:  
PK constraint:  

Options

Description
OPTAR Request Information

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual_cost</td>
<td>ACTUAL_COST</td>
<td>decimal(11,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>adp_proj_#</td>
<td>ADP_PROJ_#</td>
<td>char(7)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>category</td>
<td>CATEGORY</td>
<td>char(1)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>description</td>
<td>DESCRIPTION</td>
<td>char(50)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>doc_#</td>
<td>DOC_#</td>
<td>char(9)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>fy_ending</td>
<td>FY_ENDING</td>
<td>date</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>issued_by</td>
<td>ISSUED_BY</td>
<td>char(15)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>jon</td>
<td>JON</td>
<td>char(5)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>order_date</td>
<td>ORDER_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>po_#</td>
<td>PO_#</td>
<td>char(12)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>po_date</td>
<td>PO_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>proj_cost</td>
<td>PROJ_COST</td>
<td>decimal(11,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>recv_date</td>
<td>RECV_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

**ACTUAL_COST**

Check
<table>
<thead>
<tr>
<th>Domain:</th>
<th>fms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

**ADP_PROJ_#**

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

**CATEGORY**

**Check**

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>
DESCRIPTION

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  Lowercase:  No  Can't modify:  No  
List of values:  

DOC_#

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  
Uppercase:  No  Lowercase:  No  Can't modify:  No  
List of values:  

EMP_ID_CODE

Check

Domain:  
Low value:  
High value:  
Default value:  
Unit:  
Format:  

S-Designer  March 11, 1997  Page 130
FY_ENDING

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>9/30/97</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Uppercase: No  Lowercase: No  Can't modify: No

List of values:

ISSUED_BY

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Uppercase: No  Lowercase: No  Can't modify: No

List of values:

JON

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

S-Designer  March 11, 1997  Page 131
Physical Data Model

<table>
<thead>
<tr>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

List of values:

ORDER_DATE

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

List of values:

PO_

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

List of values:

PO_DATE
### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### PROJ_COST

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### RECVD_DATE

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTAR_REQ_FK1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>ADP_PROJ_#</td>
<td>ASC</td>
</tr>
<tr>
<td>OPTAR_REQ_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>JON</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DOC_#</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>ACCOUNT</td>
<td>JON</td>
<td>JON</td>
</tr>
<tr>
<td>ADP_PROJ_INFO</td>
<td>ADP_PROJ_#</td>
<td>ADP_PROJ_#</td>
</tr>
</tbody>
</table>

Table other_leave

Name: other_leave
Code: OTHER_LEAVE
Label: "Other" leave info per employee per pay period
Number:
PK constraint:

Options

Description

"Other" leave info per employee per pay period

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>hours</td>
<td>HOURS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ppe_date</td>
<td>PPE_DATE TYPE</td>
<td>date</td>
<td>char(2)</td>
<td>Yes</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>------</td>
<td>---------</td>
<td>-----</td>
</tr>
</tbody>
</table>

**EMP_ID_CODE**

**Check**

- Domain:
- Low value:
- High value:
- Default value:
- Unit:
- Format:
  - Uppercase: No
  - Lowercase: No
  - Can't modify: No
- List of values:

**HOURS**

**Check**

- Domain:
- Low value:
- High value:
- Default value:
- Unit:
- Format:
  - Uppercase: No
  - Lowercase: No
  - Can't modify: No
- List of values:

**PPE_DATE**

**Check**

- Domain:
- Low value:
- High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No  List of values:

TYPE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No  List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER_LEAVE_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PPE_DATE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TYPE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>LABOR_LES</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td></td>
<td>PPE_DATE</td>
<td>PPE_DATE</td>
</tr>
<tr>
<td>OTHER_LV_TYPE</td>
<td>OTHER_LV_TYPE_CODE</td>
<td>TYPE</td>
</tr>
</tbody>
</table>

Table other_lv_type

Name: other_lv_type
Physical Data Model

Code: OTHER_LV_TYPE
Label: Other Leave Type Lookup Table
Number:
PK constraint:

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td>DESCRIPTION</td>
<td>char(25)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>other_lv_type_code</td>
<td>OTHER_LV_TYPE_CODE</td>
<td>char(2)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

DESCRIPTION

Check

Domain: 
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No 
Lowercase: No 
Can't modify: No 
List of values:

OTHER_LV_TYPE_CODE

Check

Domain: 
Low value: 
High value: 
Default value: 
Unit: 

S-Designer March 11, 1997 Page 137
Physical Data Model

Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER_LV_TYPE_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>OTHER_LV_TYPE_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTHER_LEAVE</td>
<td>OTHER_LV_TYPE_CODE</td>
<td>TYPE</td>
</tr>
</tbody>
</table>

Table pi

Name: pi
Code: PI
Label: Principal Investigator
Number:
PK constraint:

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>jon</td>
<td>JON</td>
<td>char(5)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

EMP_ID_CODE

Check

S-Designor  March 11, 1997  Page 138

138
Physical Data Model

Domain:
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No  Lowercase: No  Can't modify: No 
List of values: 

JON

Check

Domain:
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No  Lowercase: No  Can't modify: No 
List of values: 

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JON</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
<tr>
<td>ACCOUNT</td>
<td>JON</td>
<td>JON</td>
</tr>
</tbody>
</table>

Table salary_history
Name: salary_history
Code: SALARY_HISTORY
Label: Employee salary history (including acceleration rate)
Number: 
PK constraint:

Options

Description
Employee salary history (including acceleration rate)

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>accel_rate</td>
<td>ACCEL_RATE</td>
<td>decimal(3, 2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>base_salary</td>
<td>BASE_SALARY</td>
<td>decimal(10,2)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>begin_date</td>
<td>BEGIN_DATE</td>
<td>date</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>end_date</td>
<td>END_DATE</td>
<td>date</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

ACCEL_RATE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

S-Designor
March 11, 1997
Page 140
BASE_SALARY

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
<th>List of values:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

BEGIN_DATE

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
<th>Uppercase:</th>
<th>Lowercase:</th>
<th>Can't modify:</th>
<th>List of values:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

EMP_ID_CODE

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Physical Data Model**

<table>
<thead>
<tr>
<th>Uppercase:</th>
<th>No</th>
<th>Lowercase:</th>
<th>No</th>
<th>Can't modify:</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**END_DATE**

**Check**

- **Domain:**
- **Low value:**
- **High value:**
- **Default value:**
- **Unit:**
- **Format:**
- **Uppercase:** No
- **Lowercase:** No
- **Can't modify:** No
- **List of values:**

**Index List**

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALARY_HISTORY_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BEGIN_DATE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

**Reference to List**

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
</tbody>
</table>

**Table sponsor**

- **Name:** sponsor
- **Code:** SPONSOR
- **Label:** Research Sponsor Info
- **Number:**
- **PK constraint:**

---

S-Designer                              March 11, 1997

Page 142
Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>ADDRESS</td>
<td>char(40)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>city</td>
<td>CITY</td>
<td>char(15)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>fax</td>
<td>FAX</td>
<td>char(13)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>name</td>
<td>NAME</td>
<td>char(30)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>phone</td>
<td>PHONE</td>
<td>char(13)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>spon_id_code</td>
<td>SPON_ID_CODE</td>
<td>char(6)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>state</td>
<td>STATE</td>
<td>char(2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>zipcode</td>
<td>ZIPCODE</td>
<td>char(10)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

ADDRESS

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td>Lowercase:</td>
<td>No</td>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CITY

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>High value:</th>
<th>Default value:</th>
<th>Unit:</th>
<th>Format:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Physical Data Model

Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

FAX

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

NAME

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No Lowercase: No Can't modify: No
List of values:

PHONE

Check
Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

SPON_ID_CODE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

STATE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:
ZIPCODE

Check

Domain: 
Low value: 
High value: 
Default value: 
Unit: 
Format: 
Uppercase: No  Lowercase: No  Can't modify: No  
List of values: 

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPONSOR_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>SPON_ID_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
<td>SPON_ID_CODE</td>
<td>SPON_ID_CODE</td>
</tr>
</tbody>
</table>

Table staff

Name: staff 
Code: STAFF 
Label: Staff Specialization of Employee Table 
Number: 
PK constraint: 

Options 

Column List

S-Designor  March 11, 1997  Page 146
### Physical Data Model

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>civ_grade</td>
<td>CIV_GRADE</td>
<td>char(5)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>emp_id_code</td>
<td>EMP_ID_CODE</td>
<td>char(4)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>step</td>
<td>STEP</td>
<td>char(2)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

#### CIV_GRADE

**Check**

- **Domain:**
- **Low value:**
- **High value:**
- **Default value:**
- **Unit:**
- **Format:**
- **Uppercase:** No
- **Lowercase:** No
- **Can't modify:** No

#### EMP_ID_CODE

**Check**

- **Domain:**
- **Low value:**
- **High value:**
- **Default value:**
- **Unit:**
- **Format:**
- **Uppercase:** No
- **Lowercase:** No
- **Can't modify:** No

#### STEP

**Check**

- **Domain:**

---

S-Designor March 11, 1997 Page 147
Physical Data Model

<table>
<thead>
<tr>
<th>Low value:</th>
<th>High value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value:</td>
<td>Unit:</td>
</tr>
<tr>
<td>Format:</td>
<td>Uppercase: No Lowercase: No Can't modify: No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>STAFF_PK</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>EMP_ID_CODE</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE</td>
<td>EMP_ID_CODE</td>
<td>EMP_ID_CODE</td>
</tr>
</tbody>
</table>

Table travel

| Name: | travel |
| Code: | TRAVEL |
| Label: | Travel Order Info |
| Number: | |
| PK constraint: | |

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>actual_cost</td>
<td>ACTUAL_COST</td>
<td>decimal(10,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>destination</td>
<td>DESTINATION</td>
<td>char(20)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

S-Designer            March 11, 1997

Page 148
Physical Data Model

<table>
<thead>
<tr>
<th>fy_ending</th>
<th>FY_ENDING</th>
<th>date</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>jon</td>
<td>JON</td>
<td>char(5)</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>num_trav_days</td>
<td>NUM_TRAV_DAYS</td>
<td>integer</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>proj_cost</td>
<td>PROJ_COST</td>
<td>decimal(10,2)</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>to#</td>
<td>TO#</td>
<td>char(15)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>to_date</td>
<td>TO_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>trav_start_date</td>
<td>TRAV_START_DATE</td>
<td>date</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

ACTUAL_COST

Check

- Domain:
  - Low value: 0.00

- High value:

- Default value:

- Unit:

- Format:

- Uppercase: No  lowercase: No  Can't modify: No

- List of values:

DESTINATION

Check

- Domain:
  - Low value:

- High value:

- Default value:

- Unit:

- Format:

- Uppercase: No  lowercase: No  Can't modify: No

- List of values:

FY_ENDING

S-Designor

March 11, 1997

Page 149
### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td>09/30/1997</td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

### JON

### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

### NUM_TRAV_DAYS

### Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td>1</td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

S-Designer: March 11, 1997
PROJ_COST

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th>0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TO##

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>High value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TO_DATE

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th>Low value:</th>
</tr>
</thead>
<tbody>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td></td>
</tr>
<tr>
<td>Lowercase:</td>
<td></td>
</tr>
<tr>
<td>Can't modify:</td>
<td></td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>
Physical Data Model

Format:
Uppercase: No  Lowercase: No  Can’t modify: No
List of values:

TRAV_START_DATE

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can’t modify: No
List of values:

Index List

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL_FK1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>JON</td>
<td>ASC</td>
</tr>
<tr>
<td>TRAVEL_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>TO#</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
<td>JON</td>
<td>JON</td>
</tr>
</tbody>
</table>

Reference by List

<table>
<thead>
<tr>
<th>Referenced by</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL_REQUESTS</td>
<td>TO#</td>
<td>TO#</td>
</tr>
</tbody>
</table>

Table travel_requests

| Name: | travel_requests |
Physical Data Model

<table>
<thead>
<tr>
<th>Code:</th>
<th>TRAVEL_REQUESTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label:</td>
<td>Information on travelers for a specific Travel Order</td>
</tr>
<tr>
<td>Number:</td>
<td></td>
</tr>
<tr>
<td>PK constraint:</td>
<td></td>
</tr>
</tbody>
</table>

Options

Column List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Type</th>
<th>P</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>to#</td>
<td>TO#</td>
<td>char(15)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>trav_fname</td>
<td>TRAV_FNAME</td>
<td>char(15)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>trav_lname</td>
<td>TRAV_LNAME</td>
<td>char(15)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>trav_mi</td>
<td>TRAV_MI</td>
<td>char(1)</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

TO#

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
<tr>
<td>Default value:</td>
<td></td>
</tr>
<tr>
<td>Unit:</td>
<td></td>
</tr>
<tr>
<td>Format:</td>
<td></td>
</tr>
<tr>
<td>Uppercase:</td>
<td>No</td>
</tr>
<tr>
<td>Lowercase:</td>
<td>No</td>
</tr>
<tr>
<td>Can't modify:</td>
<td>No</td>
</tr>
<tr>
<td>List of values:</td>
<td></td>
</tr>
</tbody>
</table>

TRAV_FNAME

Check

<table>
<thead>
<tr>
<th>Domain:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Low value:</td>
<td></td>
</tr>
<tr>
<td>High value:</td>
<td></td>
</tr>
</tbody>
</table>

S-Designer         | March 11, 1997        | Page 153
Physical Data Model

Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

TRAV_LNAME

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

TRAV_MI

Check

Domain:
Low value:
High value:
Default value:
Unit:
Format:
Uppercase: No  Lowercase: No  Can't modify: No
List of values:

Index List
Physical Data Model

<table>
<thead>
<tr>
<th>Index Code</th>
<th>P</th>
<th>F</th>
<th>U</th>
<th>C</th>
<th>Column Code</th>
<th>Sort</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL_REQUESTS_PK</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>TO#</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TRAV_LNAME</td>
<td>ASC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TRAV_FNAME</td>
<td>ASC</td>
</tr>
</tbody>
</table>

Reference to List

<table>
<thead>
<tr>
<th>Reference to</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAVEL</td>
<td>TO#</td>
<td>TO#</td>
</tr>
</tbody>
</table>

Views

View List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Upd</th>
<th>Gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>dr_chgs</td>
<td>DR_CHGS</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>dt_chgs</td>
<td>DT_CHGS</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>rr_chgs</td>
<td>RR_CHGS</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>st_fmt_chgs</td>
<td>ST_FMT_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>st_ind_chgs</td>
<td>ST_IND_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>st_ir_chgs</td>
<td>ST_IR_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>st_omn_chgs</td>
<td>ST_OMN_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>st_ot_chgs</td>
<td>ST_OT_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>st_rr_chgs</td>
<td>ST_RR_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>st_tuit_chgs</td>
<td>ST_TUIT_CHGS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

View dr_chgs

Name: dr_chgs
Code: DR_CHGS
Label: DR Charges View
Usage: Query Only
Generate View

Code
select FACULTY.EMP_ID_CODE, LABOR_CHGS.PPE_DATE, LABOR_CHGS.HOURS
from ACCOUNT, FACULTY, LABOR_CHGS
where FACULTY.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and ACCOUNT.JON = LABOR_CHGS.JON
and ACCOUNT.FUND_TYPE = 'DR'

View dt_chgs

<table>
<thead>
<tr>
<th>Name:</th>
<th>dt_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>DT_CHGS</td>
</tr>
<tr>
<td>Label:</td>
<td>DT Charges View</td>
</tr>
<tr>
<td>Usage:</td>
<td>Query Only, Generate View</td>
</tr>
</tbody>
</table>

Code

select FACULTY.EMP_ID_CODE, LABOR_CHGS.PPE_DATE, LABOR_CHGS.HOURS
from ACCOUNT, FACULTY, LABOR_CHGS
where FACULTY.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and ACCOUNT.JON = LABOR_CHGS.JON
and ACCOUNT.FUND_TYPE = 'DT'

View rr_chgs

<table>
<thead>
<tr>
<th>Name:</th>
<th>rr_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>RR_CHGS</td>
</tr>
<tr>
<td>Label:</td>
<td>RR Charges View</td>
</tr>
<tr>
<td>Usage:</td>
<td>Query Only, Generate View</td>
</tr>
</tbody>
</table>

Code

select FACULTY.EMP_ID_CODE, ACCOUNT.LABOR_JON, LABOR_CHGS.PPE_DATE, LABOR_CHGS.HOURS
from ACCOUNT, FACULTY, LABOR_CHGS
where FACULTY.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and ACCOUNT.JON = LABOR_CHGS.JON
and ACCOUNT.FUND_TYPE = 'RR'
View `st_fmt_chgs`

<table>
<thead>
<tr>
<th>Name:</th>
<th>st_fmt_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>ST_FMT_CHGS</td>
</tr>
<tr>
<td>Label:</td>
<td>st_fmt_chgs</td>
</tr>
<tr>
<td>Usage:</td>
<td>Updatable</td>
</tr>
<tr>
<td></td>
<td>Generate View</td>
</tr>
<tr>
<td></td>
<td>With check option</td>
</tr>
</tbody>
</table>

**Code**

```sql
select STAFF.EMP_ID_CODE, LABOR_CHGS.HOURS, LABOR_CHGS.PPE_DATE
from STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and LABOR_CHGS.JON = 'FMT'
```

View `st_ind_chgs`

<table>
<thead>
<tr>
<th>Name:</th>
<th>st_ind_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>ST_IND_CHGS</td>
</tr>
<tr>
<td>Label:</td>
<td>st_ind_chgs</td>
</tr>
<tr>
<td>Usage:</td>
<td>Updatable</td>
</tr>
<tr>
<td></td>
<td>Generate View</td>
</tr>
<tr>
<td></td>
<td>With check option</td>
</tr>
</tbody>
</table>

**Code**

```sql
select STAFF.EMP_ID_CODE, LABOR_CHGS.HOURS, LABOR_CHGS.PPE_DATE
from STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and LABOR_CHGS.JON = 'IND'
```

View `st_ir_chgs`

<table>
<thead>
<tr>
<th>Name:</th>
<th>st_ir_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td>ST_IR_CHGS</td>
</tr>
<tr>
<td>Label:</td>
<td>st_ir_chgs</td>
</tr>
<tr>
<td>Usage:</td>
<td>Updatable</td>
</tr>
</tbody>
</table>
Generate View
With check option

Code

select ACCOUNT.LABOR_JON, STAFF.EMP_ID_CODE, LABOR_CHGS.HOURS, LABOR_CHGS.PPE_DATE
from ACCOUNT, STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and ACCOUNT.JON = LABOR_CHGS.JON
and ACCOUNT.FUND_TYPE = 'IR'

View st_omn_chgs

<table>
<thead>
<tr>
<th>Name</th>
<th>st_omn_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>ST_OMN_CHGS</td>
</tr>
<tr>
<td>Label</td>
<td>st_omn_chgs</td>
</tr>
<tr>
<td>Usage</td>
<td>Updatable</td>
</tr>
<tr>
<td></td>
<td>Generate View</td>
</tr>
<tr>
<td></td>
<td>With check option</td>
</tr>
</tbody>
</table>

Code

select STAFF.EMP_ID_CODE, LABOR_CHGS.HOURS, LABOR_CHGS.PPE_DATE
from STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and LABOR_CHGS.JON = 'O&MN'

View st_ot_chgs

<table>
<thead>
<tr>
<th>Name</th>
<th>st_ot_chgs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>ST_OT_CHGS</td>
</tr>
<tr>
<td>Label</td>
<td>st_ot_chgs</td>
</tr>
<tr>
<td>Usage</td>
<td>Updatable</td>
</tr>
<tr>
<td></td>
<td>Generate View</td>
</tr>
<tr>
<td></td>
<td>With check option</td>
</tr>
</tbody>
</table>

Code

select ACCOUNT.LABOR_JON, STAFF.EMP_ID_CODE, LABOR_CHGS.OT_HOURS, LABOR_CHGS.PPE_DATE
from ACCOUNT, STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and ACCOUNT.JON = LABOR_CHGS.JON
and LABOR_CHGS.OT_HOURS > 0

View st_rr_chgs

Name: st_rr_chgs
Code: ST_RR_CHGS
Label: st_rr_chgs
Usage: Updatable
        Generate View
        With check option

Code

select ACCOUNT.LABOR_JON, STAFF.EMP_ID_CODE, LABOR_CHGS.HOURS, LABOR_CHGS.PPE_DATE
from ACCOUNT, STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and ACCOUNT.JON = LABOR_CHGS.JON
and ACCOUNT.FUND_TYPE = 'RR'

View st_tuit_chgs

Name: st_tuit_chgs
Code: ST_TUIT_CHGS
Label: st_tuit_chgs
Usage: Updatable
        Generate View
        With check option

Code

select STAFF.EMP_ID_CODE, LABOR_CHGS.HOURS, LABOR_CHGS.PPE_DATE
from STAFF, LABOR_CHGS
where STAFF.EMP_ID_CODE = LABOR_CHGS.EMP_ID_CODE
and LABOR_CHGS.JON = 'TUIT'

Triggers
## Trigger List

<table>
<thead>
<tr>
<th>Table</th>
<th>Trigger</th>
<th>User-Defined</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
<td>tib_account</td>
<td>No</td>
</tr>
<tr>
<td>ACCOUNT</td>
<td>tia_account</td>
<td>Yes</td>
</tr>
<tr>
<td>ACCOUNT</td>
<td>tua_account</td>
<td>Yes</td>
</tr>
<tr>
<td>ADP_PROJ_INFO</td>
<td>tib_adp_proj_info</td>
<td>No</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>tib_contracts</td>
<td>No</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>tia_contracts</td>
<td>Yes</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>tub_contracts</td>
<td>No</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>tua_contracts</td>
<td>Yes</td>
</tr>
<tr>
<td>CONTRACTS</td>
<td>tda_contracts</td>
<td>Yes</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>tib_employee</td>
<td>No</td>
</tr>
<tr>
<td>FACULTY</td>
<td>tib_faculty</td>
<td>No</td>
</tr>
<tr>
<td>LABOR_CHGS</td>
<td>tib_labor_chgs</td>
<td>No</td>
</tr>
<tr>
<td>LABOR_CHGS</td>
<td>tia_labor_chgs</td>
<td>Yes</td>
</tr>
<tr>
<td>LABOR_CHGS</td>
<td>tua_labor_chgs</td>
<td>Yes</td>
</tr>
<tr>
<td>LABOR_CHGS</td>
<td>tda_labor_chgs</td>
<td>Yes</td>
</tr>
<tr>
<td>LABOR_LES</td>
<td>tib_labor_les</td>
<td>No</td>
</tr>
<tr>
<td>MILITARY</td>
<td>tib_military</td>
<td>No</td>
</tr>
<tr>
<td>OPTAR_REQ</td>
<td>tib_optar_req</td>
<td>No</td>
</tr>
<tr>
<td>OPTAR_REQ</td>
<td>tia_optar_req</td>
<td>Yes</td>
</tr>
<tr>
<td>OPTAR_REQ</td>
<td>tua_optar_req</td>
<td>Yes</td>
</tr>
<tr>
<td>OPTAR_REQ</td>
<td>tda_optar_req</td>
<td>Yes</td>
</tr>
<tr>
<td>OTHER_LEAVE</td>
<td>tib_other_leave</td>
<td>No</td>
</tr>
<tr>
<td>OTHER_LEAVE</td>
<td>tub_other_leave</td>
<td>No</td>
</tr>
<tr>
<td>PI</td>
<td>tib_pi</td>
<td>No</td>
</tr>
<tr>
<td>SALARY_HISTORY</td>
<td>tib_salary_history</td>
<td>No</td>
</tr>
<tr>
<td>STAFF</td>
<td>tib_staff</td>
<td>No</td>
</tr>
<tr>
<td>TRAVEL</td>
<td>tib_travel</td>
<td>No</td>
</tr>
<tr>
<td>TRAVEL</td>
<td>tia_travel</td>
<td>Yes</td>
</tr>
<tr>
<td>TRAVEL</td>
<td>tua_travel</td>
<td>Yes</td>
</tr>
<tr>
<td>TRAVEL</td>
<td>tda_travel</td>
<td>Yes</td>
</tr>
<tr>
<td>TRAVEL_REQUESTS</td>
<td>tib_travel_requests</td>
<td>No</td>
</tr>
</tbody>
</table>

## Procedures
## Procedure List

<table>
<thead>
<tr>
<th>Name</th>
<th>Code</th>
<th>Func</th>
</tr>
</thead>
<tbody>
<tr>
<td>calc_bal_contract</td>
<td>CALC_BAL_CONTRACT</td>
<td>No</td>
</tr>
<tr>
<td>calc_bal_fac_labor</td>
<td>CALC_BAL_FAC_LABOR</td>
<td>No</td>
</tr>
<tr>
<td>calc_bal_optar</td>
<td>CALC_BAL_OPTAR</td>
<td>No</td>
</tr>
<tr>
<td>calc_bal_spt_labor</td>
<td>CALC_BAL_SPT_LABOR</td>
<td>No</td>
</tr>
<tr>
<td>calc_bal_trav</td>
<td>CALC_BAL_TRAV</td>
<td>No</td>
</tr>
</tbody>
</table>
**INITIAL DISTRIBUTION LIST**

<table>
<thead>
<tr>
<th>No. Copies</th>
<th>Address</th>
</tr>
</thead>
</table>
| 2          | Defense Technical Information Center ........................ 8725 John J. Kingman Rd., STE 0944  
Ft. Belvoir, VA 22060-6218 |
| 2          | Dudley Knox Library ........................................... Naval Postgraduate School  
411 Dyer Rd.  
Monterey, CA 93943-5101 |
| 1          | Chairman, Computer Science Department ..................... Naval Postgraduate School  
Monterey, CA 93943-5000 |
| 1          | Chairman, Operations Research Department .................. Naval Postgraduate School  
Monterey, CA 93943-5000 |
| 1          | Dr. C. Thomas Wu ............................................. Naval Postgraduate School  
Monterey, CA 93943-5000 |
| 1          | Dr. James Emery .............................................. Naval Postgraduate School  
Monterey, CA 93943-5000 |
| 1          | LCDR John A. Daley ........................................... Naval Postgraduate School  
Monterey, CA 93943-5000 |
8. Alan E. Pires ................................................................. 2
   Code OR
   Operations Research Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

9. Mr. and Mrs. Robert D. Pires ......................................... 1
   920-35th Street
   Richmond, CA 94805