American Short Line Railroad Database System Phase II

Douglas Benson
Trent Byberg

UGPTI Staff Paper No. 134
September 1996

American Short Line Railroad Database System

A Technical Report on Phase II Development

Douglas Benson
Trent Byberg

UGPTI Staff Paper No. 134

Upper Great Plains Transportation Institute
North Dakota State University
Fargo, North Dakota

September 1996

TABLE OF CONTENTS

List of Tables

List of Figures

INTRODUCTION

The short line railroad industry is an important sector within the transportation system in the United States. To reflect the importance of this specific transportation sector, information is collected on the short line railroads from an annual survey. This survey information is divided by subject content, then stored in a database, and its data is organized in a specific, predefined order. The organization of that data allows users to efficiently extract information on the short line railroad industry from the database.

Before the creation of the American Short Line Railroad Database, information describing the distinct characteristics of this transportation sector were not readily available. The current database has two years of information on the industry, 1993 and 1994, and collection of information for the 1995 calendar year is underway.

This report describes the different programming functions used now and discusses future programming issues. The first three chapters of the report give the details of the software and programming techniques utilized for the current database system. Information on the current database, data entry software, and data analysis software used all are covered. In the final three chapters, ideas and products are examined for potential implementation into the project. Ideas presented include future database software, data entry software, and data analysis software.

Chapter 1 examines and outlines the database. The current Microsoft Access database design is presented, and positive and negative aspects of the design are described in detail. Some of the issues covered include table design, referential integrity, data entry interface, database security, and compatibility.

Chapter 2 contains an in-depth look at the current data analysis techniques and reporting methods utilized. Different ways of analyzing the data and different types of statistical tools utilized are described and reviewed. Software packages used in the analysis are also examined.

Chapter 3 presents current data collection methods and describes current electronic data collection and software distribution methods used. Both positive and negative aspects of the software design and distribution are discussed.

The next three chapters of the report examine and review future considerations for the information system. The subject matter in these three chapters describe enhancements that can be applied to improve the system. Chapter 4 reviews possible improvements to the current database design. Issues covering speed, accuracy, and security of the information in the database are discussed.

A central database analysis interface is reviewed in Chapter 5. This chapter includes an in-depth discussion on developing an interface that will integrate several Windows applications used for editing and analyzing the database information. Reviews of different software development packages and statistical analysis software make up the bulk of the chapter.

Chapter 6 discusses strategies for the data entry software program. Improvements on the previous Dentry 2.0 software are discussed in detail, and new Visual Basic techniques that can be employed for multi-year data are considered. Different tools such as Visual FoxPro, the Internet, and EDI are reviewed for possible implementation.

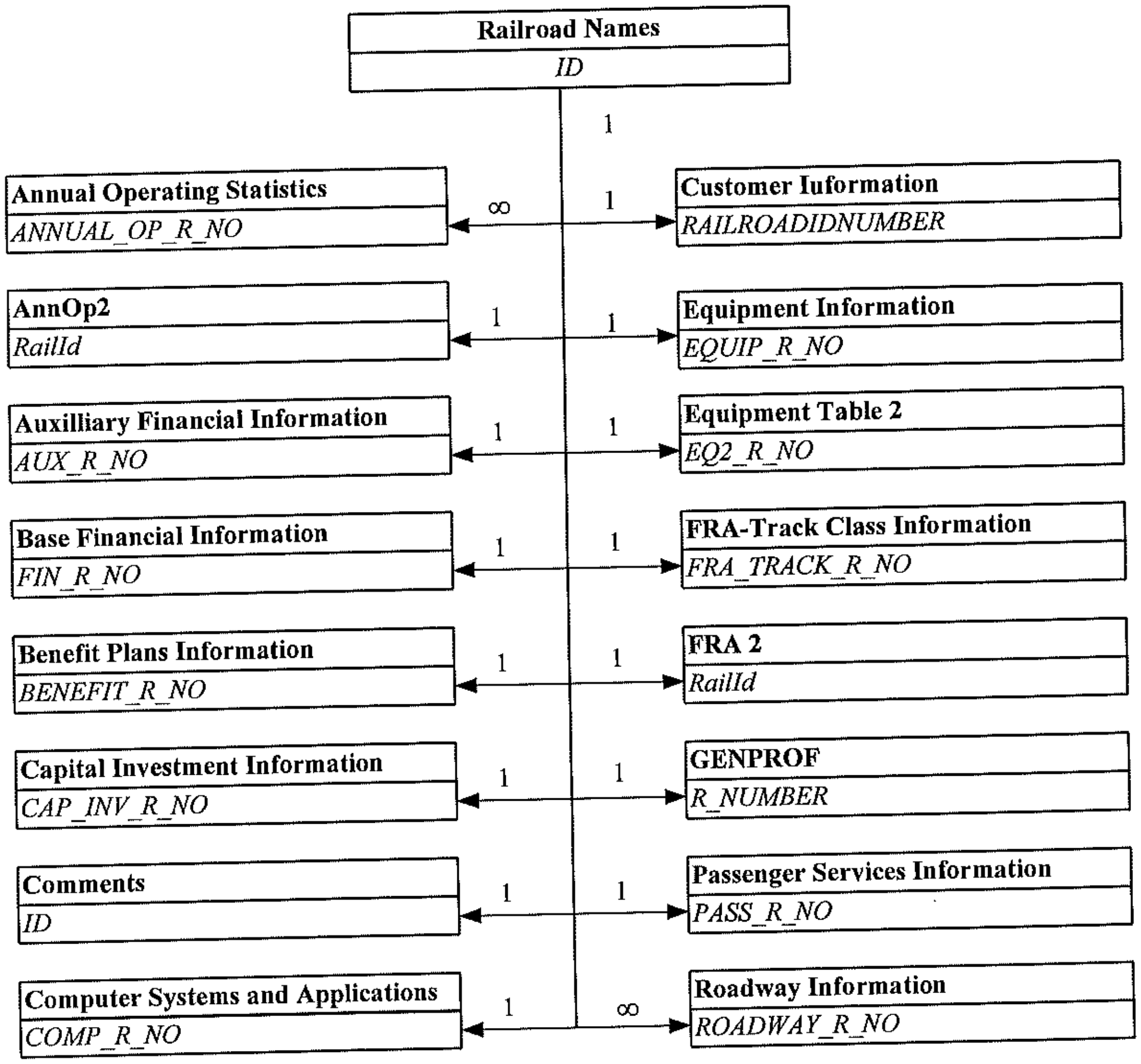
The final part of the report is made up of several appendices. Appendix A and B contain the SAS code and SAS output from the Univariate procedure. Appenix C contains diagrams depicting the flow of Dentry 2.0 procedures during certain events. A list of all the database fields within each table of the 1995 database is in Appendix D. Appendix E is the list of files compiled with Dentry 2.0. Appendix F is the list of files distributed with the Dentry 2.0 setup disks.

CHAPTER 1. THE DATABASE DESIGN

Database Structure

The 1995 short line database is in the Microsoft Access 2.0 format. Survey data is stored in 17 different tables. Each table contains a primary identification number unique to each railroad in the database. The primary key from which all other tables are linked is the ID field within the Railroad Names table. These links demonstrate relationships between records in two tables. Figure 1.1 shows each table and its relationship to the Railroad Names' ID field.

Figure 1.1: Database Table Relationships



As shown in the chart, two tables in the database have one-to-many relationships with the Railroad Names table Annual Operating Statistics and Roadway Information. A particular value of the ID field in the Railroad Names table can appear more than once in a one-to-many related table. The other 14 tables in the database only allow a one-to-one relationship with the Railroad Names table. A particular value of the ID field in the Railroad Names table can appear only once in a one-to-one related table. Appendix D lists the fields in each database table.

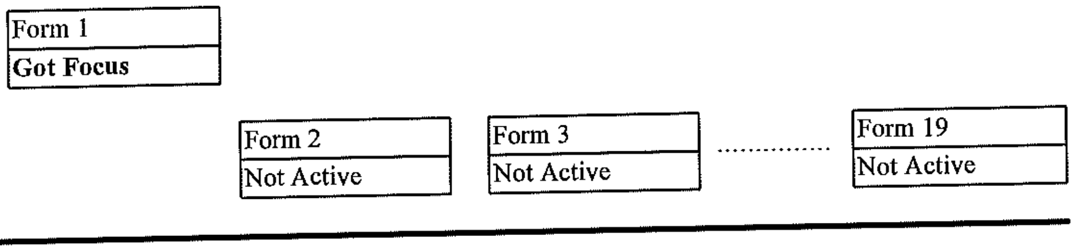
For security reasons, the database is assigned a login and password. Only the database administrators have access to the login and password. Whenever Microsoft Access starts up, a form appears prompting for the login and password. Once the correct login and password are entered, one may access the database. When a change is needed to the information in the database, the change will occur only through the database administrators. This allows the administrators to keep the information in the database consistent.

Data Entry Interface

The data entry interface is composed of forms designed in Microsoft Access. The 1995 ASLRA data entry interface consists of 19 forms. The design of the forms in Microsoft Access is similar to the process of designing forms in Visual Basic. In Microsoft Access, a form design window is utilized to create and modify forms, and each form is constructed to be similar in appearance to the paper survey. Interface objects placed on the form include textbox fields, combo boxes, and checkboxes. These interface objects provide a direct link between the user and the database fields. Other objects such as labels, frames, and graphics are placed on the forms to group information and help describe the interface objects. Command buttons on the top of the forms allow a user to move between the forms, delete records, add records, and stop the data entry. Currently, Microsoft Access automatically saves a record to the database when the focus is moved to a different record or the active form or database is closed. For this interface, each time a new form is opened, the previous form loses focus. Thus, changes made with the previous form controls are automatically saved to the database, as illustrated by Figure 1.2.

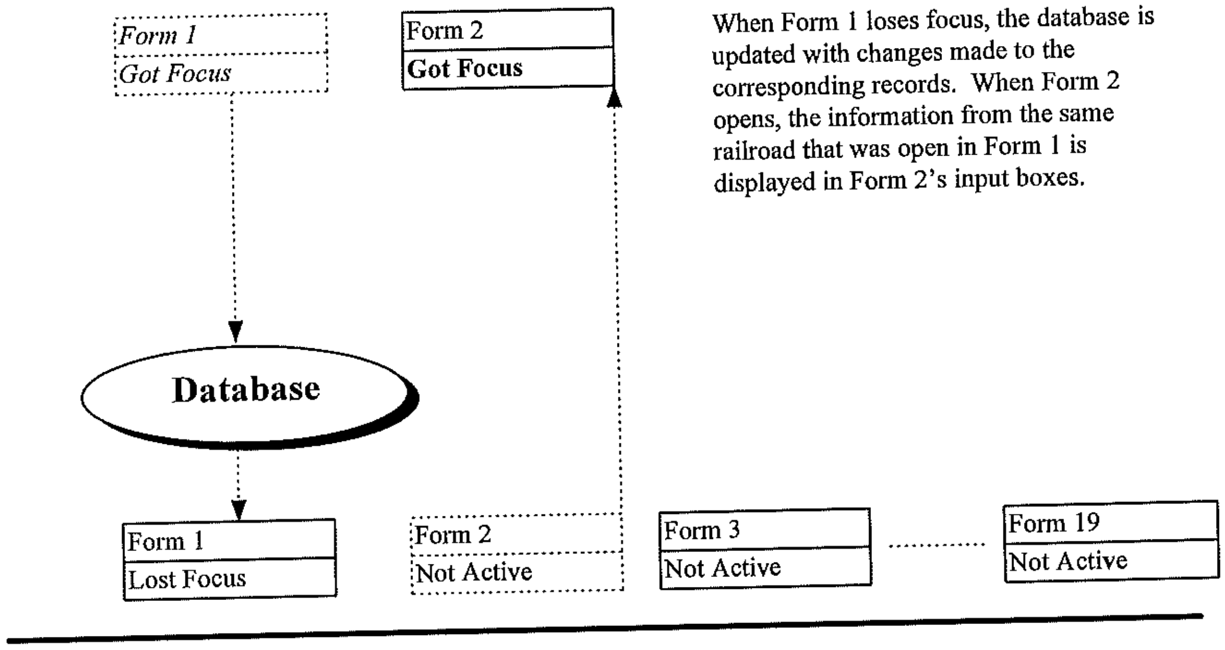
In the chart below, Form 1 is active and survey information for this form is entered. Changes to Form 1 are not saved to the database until Form 1 loses focus.

Figure 1.2: Active Form



When you press the command that opens up the next survey form, Form 1 becomes inactive, and Form 2 becomes active. All information entered into Form 1 is automatically saved to the corresponding database table.

Figure 1.3: Lost Focus Event



When Form 1 loses focus, the database is updated with changes made to the corresponding records. When Form 2 opens, the information from the same railroad that was open in Form 1 is displayed in Form 2's input boxes.

Microsoft Access basic language is utilized to help control the flow of the database entry interface. Each object placed on a form has different event procedures. Code associated with that event is called on whenever the event takes place. The command button that closes the current form and opens the next survey form is an example of a command object. When the user clicks on this command button, it's corresponding *on click* event procedure is envoked. The *on click* event procedure for the next form command button has the following code:

```
Dim DocName As String                'Declare local variable for the form to open
Dim LinkCriteria As String           'Declare local variable for the form link criteria

  DocName = "Customer Form"          'Assign the form to be opened to the DocName var.
                                  'Assigns current railroad ID number to the next form
  LinkCriteria = "[RAILROADIDNUMBER] = Forms![Railroad Names]![Field34]"
  DoCmd OpenForm DocName, , , LinkCriteria  'Opens new form with railroad data
                                            'matching the LinkCriteria ID
  DoCmd Close A_Form, "Railroad Names"      'Closes the current form
```

The LinkCriteria variable links the record number from the railroad names table to the record identification number in the customer information table. The DoCmd opens the customer form with the LinkCriteria record displayed. If there is no matching record in the customer information table, a blank record is displayed in the customer form. The final DoCmd closes the railroad names form. When this occurs, the focus is lost from the Railroad Names form, automatically saving any editing that was done in that form.

CHAPTER 2. DATA ANALYSIS METHODS

Types Of Statistical Analysis

Dealing With Non-response And Zero-response

The main purpose of an analysis is to estimate, with the greatest precision, the population parameters. The responses to some of the questions in the annual short line survey were not always straightforward. Railroads filling out the survey do not always provide appropriate responses to the survey questions. The front page of the survey requests railroads to respond with an "N/A" for information that is not available. Many of the respondents leave survey questions blank. When a question is left blank, it is not known whether they meant a 0 or N/A response. Guidelines were developed to interpret the three following types of responses, i.e. N/A, 0, and blank.

If the railroad responds with an N/A for a certain question, the database stores a -1 in the corresponding database field. This -1 value is a flag telling the analysis team to ignore that response. If the question is left blank, the question is reviewed and a conclusion is made on whether or not the respondent meant it to be 0 or N/A. For most of the questions left blank, it is obvious that a zero response is actually meant. Questions that are suspicious when left blank generally come from the Annual Line-haul Operating statistics section, Base Financial Information section, Projected Capital Investment section, and the Employee Data section. 1n each of these sections, a zero response for certain questions does not make sense. It generally appeared that this information was actually not available. During the analysis, blank responses were reviewed, and either a 0 or a -1 was entered into the database.

The analysis continues by considering each individual computed statistic. For some statistics, a zero response is valid, but for others, a zero response will make the statistic invalid. The following examples use the 1994 data and show how different analysis results can be, depending on the inclusion of zeros when computing the statistic.

Table 2.1: Statistics With And Without Zero

Statistic	Mean (excluding zeros)	Median (excluding zeros)	Mean (including zeros)	Median (including zeros)
New Ties Replaced per Railroad	8,180	1,812	6,162	950
Operating Revenue per Man Hour Worked	$78.32	$64.62	Invalid	Invalid

With the statistic, New Ties Replaced per Railroad, a value of zero is valid. There may be railroads that did not insert any new ties on their lines in a particular year. Therefore, when computing New Ties Replaced per Railroad, one should include zeros in the computation. When a ratio is computed, such as Operating Revenue per Man-Hour Worked, both parameters have to be considered. A zero value for the denominator would make that ratio invalid. That particular response would have to be dropped. A zero response in the numerator would need further analysis. The example ratio computed, Operating Revenue per Man-Hour Worked, has a numerator where a zero response generally would not make sense. If gross operating revenue is zero, the railroad could not function profitably. An observation like this also would be eliminated from the analysis.

Descriptive Statistics

There are several different types of descriptive statistics computed from the database. Industry totals, percentages, averages, medians, ratios, and outlier statistics all are computed. The main purpose for the outlier statistics is to assist in locating data entry errors and suspect data points. Ranges and extreme values also are reviewed for this purpose. Suspect values are compared with the actual paper survey response to check for data entry error. Also, values that do not appear to match a particular railroad's size and type are reviewed. If the value is still questionable after review, the railroad may be contacted for verification of the value.

There exists a wide variation in size among the railroads in the survey. To reduce the variation of the statistics, the railroads are grouped according to their size and region. They are separated into regional and local-line haul railroads, based on gross operating revenue, and miles of track owned. The switching and terminal railroads are self-designated. The railroads also are grouped into regions defined by the American Short Line Railroad Association. Statistics are computed for each of the three types of railroads and for each of the ASLRA regions.

Descriptive statistics also are prepared upon request. Railroads and organizations can request statistics to be computed from the central database. First, the American Short Line Railroad Association has to okay the release of the requested information. Once okayed, the development team runs analysis for the requested information. The computational results, along with comments on the accuracy of the results, are then sent to the requesting party. Listed below are several examples of information requested from the database:

Average Revenue Per Carload
Revenue Per Mile
Carloads Per Mile
Operating Ratio
Carloads Per Locomotive
Employee Compensation To Expense Ratio
Projected 5-Year Investment

Revenue Per Employee
Carloads Per Employee
MOW Expense Per Mile
Payroll Per Revenue
Carloads Per Transportation Employee
FRA Track Class Totals

Software Used In Analysis

Microsoft Excel

Currently, the Microsoft Access program is not setup to conduct any statistical analysis on the data. Microsoft Excel 5.0 has several straighforward statistical functions and graphing capabilities that can be applied to the data. Therefore, each table in the Access database is exported to a worksheet in Excel. All tables that have a one-to-one relationship are then combined into one worksheet for the purpose of cross-table analysis. Using Excel functions, basic descriptive statistics can be computed rather quickly. The Excel functions have the capability to compute means, medians, standard deviations, and sums of variables.

Table 2.2: Microsoft Excel Spreadsheet Calculations

	A	B
1	Average Length Of Haul	Total Number of Carloads
2	24	100
3	23	800
.	.	.
.	.	.
200	12	754
201	=AVERAGE(A1:A200)	=AVERAGE((B2:B200/A2:A200))

In Table 2.2 above, cell A201 would display the overall average length of haul, and cell B201 would display the average number of carloads per mile for the railroads. The database contains a -1 for all N/A values. To skip these, the formulas would be adjusted as follows:

```
        Cell A201:
=Average(If(A2:A200<>-1,A2:A200))

        Cell B201:
=Average(If(A2:A200<>-1,If(B2:B200<>-1, B2:B200/A2:A200)))
```

An error will occur in cell B201 if any zero values are in cells B2:B200. Also, a zero value for the average length of haul does not make sense. To correct for this, the following formula is used:

```
        Cell B201:
=Average(If(A2:A200<>-1,If(B2:B200<>-1,If(A2:A200<>0,
 If(B2:B200<>0,B2:B200/A2:A200))))
```

This formula will compute the ratio, average number of carloads per mile, only for responses that have both average length of haul and total carloads that do not equal -1 and do not equal 0.

SAS

The database tables can easily be converted to tab or comma delimited text files in Microsoft Access or in Microsoft Excel. These files are then uploaded to a UNIX environment that has a SAS installation. SAS procedures are utilized to obtain ranges, extreme values, and general descriptive statistics. These descriptive statistics can be computed quickly in SAS using the Univariate procedure. This procedure will give us the sum of the variable, overall mean, standard deviation, 25th, 50th, and 75th percentiles, kurtosis, skewness, and the five largest and five smallest data points. Proc Capability is utilized for creating histograms and visualizing the statistics. Appendices A and B show example code for Proc Univariate and Proc Capability, and output for Proc Univariate.

CHAPTER 3. THE DENTRY SOFTWARE PROGRAM

An electronic data entry program was created and used for 1993 and 1995 annual data profiles. Dentry version 1.0 was the first data entry software program created for the 1993 annual data profile. A previous version of Visual Basic, version 2.0, was used for the design of Dentry 1.0. Due in part to the limitations of this version of Visual Basic, and the availability of program add-ons, Dentry did not accommodate all the capabilities that were desired. No data entry software was developed for the 1994 data profile. For the 1995 data profile, the data entry software program was added to the project. This program, Dentry 2.0, employs new features included with Visual Basic 3.0 and added functionality by utilizing third party products.

Several major goals for developing the software program for the survey were identified. First, the survey is time consuming to complete due to its length and the amount of information needed. A user-friendly program with easy to learn functionality can increase the speed with which the survey is filled out and increase the response rate.

Second, is the need to reduce data entry errors. Entering information from paper surveys into a database will lead to typing errors. With the data entry software, railroads forward the data on disk, then the data is appended to the central database. No entering of data and deciphering of handwriting is needed to enter this railroad into the central database. Therefore, the data entry errors made by the database administrator are reduced, and this leads to fewer resources needed for checking the validity of the data. An overview of the techniques and functions used in developing the software is discussed in the next three sections.

The Dentry 2.0 Program Structure

The Visual Basic 3.0 language was used to create the Dentry 2.0 software package. Dentry 2.0 is made up of several types of files: forms, basic modules, and Visual Basic custom controls. To better understand how the program works, the main modules, forms, and vbx's that were utilized in Dentry 2.0 are discussed in this section.

Three basic modules are important to the functionality of Dentry. The module containing the most functionality is Module11.bas. Nearly all of the global declarations for Dentry 2.0 are included in this module. All the database field variables in the database are declared in this module and most of these data field variables are two-dimensional. The first dimension index shows the current survey year. The index 0 implies that the information is for the 1995 database. The index 1 implies that the information is for the 1994 database. The second dimension of the array signifies which database field is being addressed.

Table 3.1: Visual Basic Global Database Variables Sample

Array Variable	Contents of Array Variable
Custnum(0,0)	the total number of coal customers in 1995
Custnum(1,0)	the total number of coal customers in 1994
Custnum(0,1)	the total number of farm customers in 1995
Custnum(1,1)	the total number of farm customers in 1994
. . .	. . .

The snapshots and dynasets for each of the database tables also are declared in this module. A snapshot is a static set of records that can be used to find data and give the program read-only access to the database tables. The CreateSnapshot method creates a recordset of one or more tables. Once created, the snapshot's current record is the first record. Moving around the recordset is controlled by MoveNext, MoveLast, and MoveFirst methods. To find a specific record, the FindFirst, FindLast, and FindNext methods can be used. The snapshot variables ending in 94 signify snapshots for the 1994 database tables. All other snapshot variables are for the 1995 database tables.

A dynaset is a dynamic set of records that has updatable records. The database object represented by the dynaset is a dynamic set of records that can be used to add, change, and delete records from the underlying database tables. The CreateDynaset method is used to create a recordset of one or more updatable tables. Once created, the dynaset's current record is the first record. Moving around, and finding specific records in a recordset are controlled by the same methods as the snapshot. The AddNew, Delete, Edit, and Update methods all are used to save changes directly to the database. A dynaset differs from a snapshot in that snapshots cannot manipulate save changes to the database, whereas dynasets can.

Flag variables also are an important part of this module. Listed below are some of the important global flag variables and their descriptions.

Q = 1	tells the program that 1994 data is being viewed in a form - do not allow saving of information to the database
Q = 0	tells the program that 1995 data is being viewed or edited in a form - allow updating of information to the database
OpenFlag = 0	tells the program that no record is currently open
OpenFlag = 1	tells the program that a record is currently open
RR_ID	holds the current record number for a 1995 survey record
RR_ID94	holds the current record number for a 1994 survey record
SaveFlag = 0	implies a new record is created - use Addnew to update database
SaveFlag = 1	implies an existing record is created - use Edit to update database
DoneFlag	specifies whether a record is open or not
NoData	lets the program know when a text box on the first main form was left blank
Dat94	lets the program know if 1994 or 1995 data is being displayed
TheForm	global form variable
AnyForm	global form variable
OutputDestination	print to window or to the printer
No_of_Copies	number of copies to print

The module11 basic file also contains about a dozen function procedures that are utilized throughout the application. These functions control initialization of variables, opening, closing, and updating databases, and obtaining information. The following table lists each procedure and gives a brief description of its function.

Table 3.2: Module11.bas Procedures

Procedure	Description
ClearAll	Clears all of the database field variables
Close_Dynasets	Closes all of the Dynasets
Copy_Display_Information	Copies all the information for the railroad into the variables
GetRailDetails	Obtains the identifying information for the railroad
MainMenu	Opens up the Main Menu Form
Open_Dbase	Opens the slrd95.mdb Access database
Open_Dbase94	Opens the slrd94.mdb Access database
Open_Dynasets	Opens the dynasets
Take_Data	Checks to make sure all identifying information is filled out
Update_Dynasets	Updates an existing database record in slrd95.mdb
Update2_Dynasets	Adds a new database record to slrd95.mdb

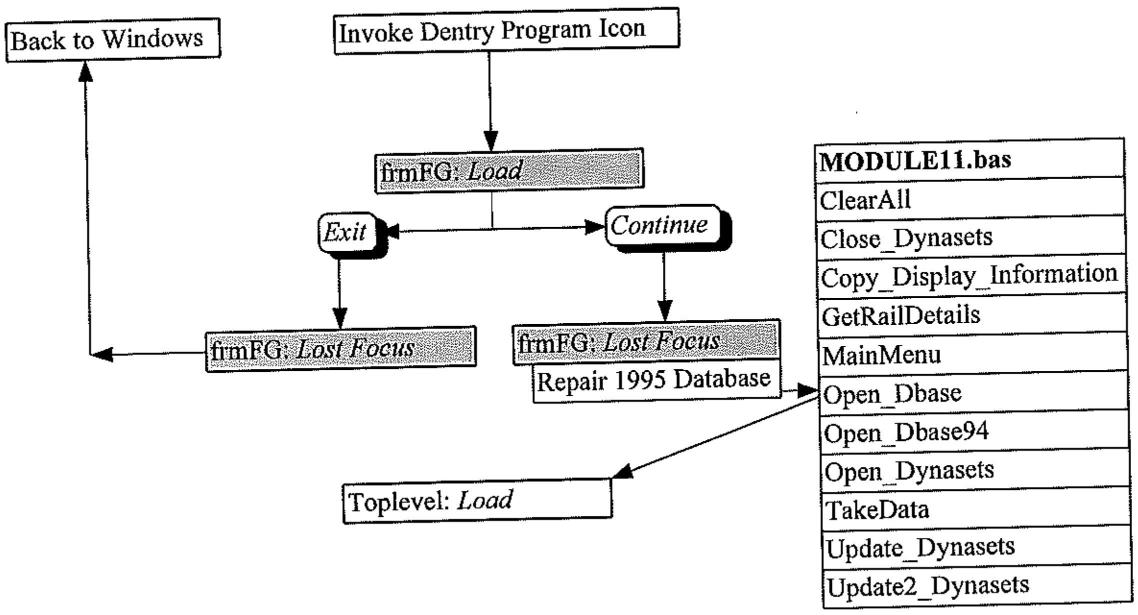
The other two important basic modules are RSDECL.bas and RSWVB.bas. RSDECL.bas contains several important VB functions for the rsreport.dll. The rsreport.dll functions are utilized for the display and printing of the survey reports. RSWVB.bas sets a global constant for how the reports are printed.

The Visual Basic interface design of Dentry 2.0 provides a graphical user interface to the database. Creating an interface that best represents the survey involves designing forms similar to the pages of the paper survey. The design of the database interface required designing multiple forms for the survey. The most appropriate approach to represent these multiple forms is to use the MDI (Multiple Document Interface) capabilities in Visual Basic. An MDI application contains only one MDI form. Dentry 2.0's MDI form is Toplevel. This form is the center of the application, containing all the application's child forms. The child forms of Toplevel consist of the pages of the survey. The menu structure and toolbar within Toplevel give the user control over filling out the survey pages.

Program Control Flow

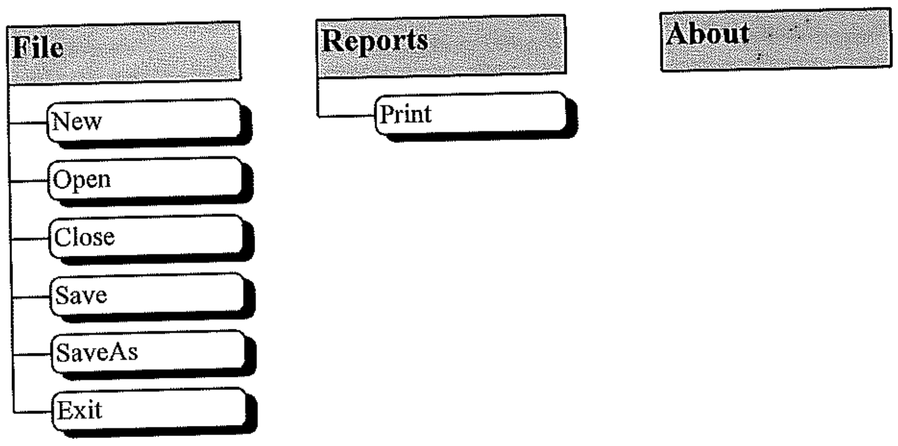
Dentry 2.0 begins by opening up its startup form, frmFG.frm (Figure 3.1). This form presents the user with two options. The user may exit the program or continue on with Dentry. If the continue command is pressed, the 1995 database is opened, frmFG is closed, and the Toplevel MDI form is opened.

Figure 3.1: Dentry's Program Startup Flow



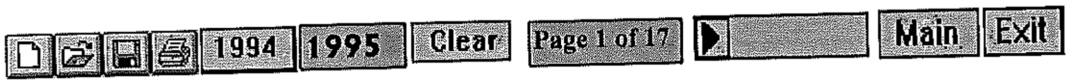
Toplevel is the MDI form controlling the flow of the entire application. The menu structure and the toolbar found in Toplevel give the user access to the majority of the functions included with Dentry 2.0. Figure 3.2 shows how the menu structure is setup.

Figure 3.2: Dentry's Menu Structure



Most of the toolbar commands also can be found in the menu. The commands that are in both the toolbar and the menu provide the exact same functionality as their counterpart. The toolbar commands that do not have any corresponding menu commands are 1994, 1995, clear, scrollbar, and main. Figure 3.3 shows the view of the toolbar with all the toolbuttons visible. From left to right they are: New, Open, Save, Print, 1994, 1995, clear, the current page label, Scrollbar, Main, and Exit.

Figure 3.3: Dentry's Toolbar



Flow charts describing the flow of the program when certain sequences of commands are invoked can be found in Appendix C.

Control

Each of the electronic survey pages contain many text boxes that collect the responses to the survey questions. The four main data types collected in the text boxes are integer, real, currency, and text. The input entered into these text boxes need to meet certain restrictions. The first restriction is the length of the value entered. Each text box has a TextLength property. When a user tries to enter more characters than the number in the TextLength property, nothing happens. No matter what is entered, no more characters are allowed in the text box. This property is useful for restricting the size of numeric values entered into a text field. For example, the TextLength property for the total number of customers field might be set to six. This will prevent a user from accidentally entering a numeric value that is in the millions, clearly an impossible amount for this particular text field.

Type Checking

Another limitation placed on the text box fields is type restriction. All values entered in the text boxes are of the text type. Therefore, a large number of the text boxes in Dentry will have to convert their contents to numeric values before saving them into the data storage structure. If a user enters a value that cannot be converted to a numeric value, a type mismatch occurs and the program crashes. To avoid this problem, Dentry checks the text box each time a value is entered or changed.

When a text box recognizes that an event has occurred, it automatically invokes the corresponding event procedure. The change event procedure is invoked whenever the user enters new information into the text box. Type checking code placed in the change event procedure can restrict the type of data input into a text box. If adequate type checking isn't coded, a type mismatch error will occur when a text value is attempted to be converted to an incompatible numeric type. For example, a type mismatch error will occur if a character such as, "123ab," is entered and attempted to be converted to an integer value. If a type mismatch error occurs, the program will crash and all unsaved changes to the database will be lost. If the type checking code is robust, a type mismatch error should never occur. The following code checks the information entered into the CStxt text box, which is the user input box for the total number of customers served by a railroad:

```
Sub CStxt_Change ()
If Not IsNumeric(Trim(CStxt.Text)) And CStxt.Text <> "" Or CStxt.Text = "." Or
CStxt.Text = "0." Then                    'Checks for unnacceptable non-numeric characters
      MsgBox " Enter Integer Values Only"
      CStxt.SetFocus
      CStxt.Text = ""
```

```
        Exit Sub
    End If
    If CStxt.Text = "" Then                'If input box is left blank, store a -1 in the database
        Total_All_Customers(0) = -1
    ElseIf CStxt.Text <> "" Then           'otherwise, convert the text value to the long integer type
        Total_All_Customers(0) = CLng(CStxt.Text)
    End If
    Exit Sub
```

The first If...Then loop checks to see if the value entered is a numeric value. It also checks if the value entered has a decimal point, since this question only takes integer responses. If the value entered is not the integer type, the text box CStxt is cleared and a message box appears prompting the user to enter an integer value.

The next If...Then loop checks to see if the text box was left blank. If it was, the code stores a -1 in the database field variable, Total_All_Customers(0), to signify an N/A response. Otherwise, if the value passes all the type checking and is not blank, the value is converted to a long integer and saved to the appropriate database variable.

Type checking the currency fields is a little more complicated. If the Visual Basic function Not IsNumeric is used to check for invalid type, problems may occur. When a value is entered into a currency field, it is changed to the formatted style, $###,###,###.00. This formatted value in the text box is a text value. With Visual Basic, a value in this format will always fail the IsNumeric check because of the '$' character or a "-$". The IsNumeric function is still needed to check for invalid text entered, but another check also is needed if the number is in the currency format shown previously. The code shown below for the change event for Mh3dText1 compensates for this by cutting off the $ character or the "-$" that might be out in front of the text before using the IsNumeric type check.

```
Sub Mh3dText1_Change (Index As Integer)
Dim Temp As String
Dim Temp1 As String
Dim Temp2 As String
Dim Temp3 As String
```

```
If Q = 0 Then                    'Checks for the current year, if it is not 1995, exit &
                                 'do not save changes

On Error GoTo Handler

If Mh3dText1(Index).Text = "" Then            'If field is blank, save a -1, and exit
  FDDATA(0, Index) = -1
  Exit Sub
Else                                          'If not blank, check type of data
  Temp = CStr(Mh3dText1(Index).Text)          'Converts Text Box info to a string
  Temp3 = CStr(Mh3dText1(Index).Text)         'A second copy of the Text Box
  Temp1 = Left(Temp, 1)                       'Stores leftmost character of input
  Temp2 = Left(Temp, 2)                       'Stores two leftmost characters
  Temp = Right$(Temp, Len(Temp) - 1)          'Stores all characters except leftmost
    If Len(Temp3) > 1 Then                    'If length of text input is > 1
      Temp3 = Right$(Temp3, Len(Temp3) - 2)        'Stores all but two leftmost
      Else                                    'If length of text input is not > 1
      Temp3 = Right$(Temp3, Len(Temp3) - 1)        'Store all but the leftmost
    End If

    If Temp1 = "-" Or Temp1 = "$" Then        'If leftmost character is "-", or "$"
      If Not IsNumeric((Temp)) And Temp <> "" Then         'Check rest of value
       MsgBox " Please enter Numeric Values"
       Mh3dText1(Index).SetFocus
       Mh3dText1(Index).Text = ""
            Exit Sub
      End If
    End If

    If Temp2 = "-$" Then                 'If two leftmost characters = "-$" then
      If Not IsNumeric(Temp3) And Temp3 <> "" Then         'Check rest of value
       MsgBox " Please enter Numeric Values"
       Mh3dText1(Index).SetFocus
       Mh3dText1(Index).Text = ""
       Exit Sub
      End If
    End If
  End If

 If Temp1 <> "-" And Temp1 <> "$" And Temp2 <> "-$" And Temp1 <> "(" And
          Mh3dText1(Index).Text <> "" Then     'Check value if it is not currency format yet
   If Not IsNumeric(CStr((Mh3dText1(Index).Text))) Then
          MsgBox "Please Enter a Numeric Value"
          Mh3dText1(Index).SetFocus
          Mh3dText1(Index).Text = ""
          Exit Sub
   End If
 End If
   If IsNumeric(Trim(Mh3dText1(Index).Text)) Then
```

```
                If Mh3dText1(Index).Text <> "" And Mh3dText1(Index).Text <> "." Then
                        'If IsNumeric, and if field is not blank or field is not just a "."
                FDDATA(0, Index) = CCur(Mh3dText1(Index).Text)
                End If
            End If
        End If
        Exit Sub
        Handler:
            Select Case Err
            Case 13
            MsgBox " A Type Mismatch Error Has Occurred - Clearing Field"
            Mh3dText1(Index).Text = ""
            Mh3dText1(Index).SetFocus
            Case Else
                MsgBox "Unexpected error #" + Str(Err) + " occurred: " + Error
            End Select
            Resume Next
            End Sub
```

A text value of "$300.00" should pass the IsNumeric test, while a text value of "$3ab5.00" should fail. The change event eliminates the leading text-like characters in Mh3dText1 that will fail the IsNumeric check. After the removal of "-", or "-$", or "$," a valid entry should pass the IsNumeric check, be converted to a currency value, then be saved to a database variable. A similar change event is written for the text boxes that will be converted to percentage values. This format, ###.##%, creates the need to drop the leftmost character if the value is negative, and drop the rightmost character, %, before checking whether the value is numeric or not.

Dentry's Database Activities

Opening The Database

The first process that occurs after the continue command is pressed on Dentry's startup form is the opening of the database. The Opendbase procedure from Module11.bas is invoked just before formFG is closed and the Toplevel form is opened. OpenDbase uses the following line of code to set the global variable Db to the 1995 database.

```
Set Db = OpenDatabase("C:\dentry2\SLRD95.mdb", False, False)
```

The OpenDatabase command opens the Microsoft Access database SLRD95.mdb and sets the exclusive and read-only properties to False. If the exclusive property is set to true, then only one user has access to the open database. If the exlusive property is set to False, then multiple users can have access to the database. A True value for the ReadOnly property makes the open database read-only. A False value for the ReadOnly property makes the open database read-write. The ReadOnly property is ignored for ODBC sources, which is the case with the underlying Microsoft Access database in Dentry 2.0. Once the database is set to the Db variable, it is in memory and the program will have access to its tables and data.

When a user opens an existing railroad from the database, snapshots of the database tables are utilized. First, the snapshot reads in the railroad names of each railroad in the database. Second, the names are sorted alphabetically and displayed in the RailroadNames form's listbox, and the snapshots are closed. Third, when a user selects a railroad from the listbox the snapshots are again opened and the record number matching the railroad selected is found. This ID number is used to copy the record's corresponding table information from the snapshots to the program variables in memory. Finally, once all the railroad's information is stored in memory, the snapshots are closed. All the data for that particular railroad is now available to be displayed in the survey page forms.

Displaying Data

Database values are displayed using the DisplayValues procedure found in each form's load event. This procedure converts the database field variable data to the text type by using the CStr function. The data type is formatted accordingly and the formatted data is displayed in each text box's text property. The following DisplayValues procedure is an example taken from the customer information form of Dentry.

```
Sub DisplayValues ()
  Dim i As Integer

  For i = 0 To 15
      If RTS_Other_Values(0, i) = -1 Then        'If a database field contains the value
       mh3dText1(i).Text = ""                    'corresponding to an N/A response, then
                                                 'display an empty textbox
                                                            'Otherwise
       Else
       mh3dText1(i).Text = CStr(RTS_Other_Values(0, I))     'Display the value stored in
                                                            'the database

       mh3dText1(i).Text = Format$(mh3dText1(i).Text, "###,##0")   'Then format the
                                                                   'value with commas

     End If
   Next
   If NewFlag(12) = 1 Then                  'If it is a new record in the database,
     For i = 0 To 15                        'then display all blank text boxes
       mh3dText1(i).Text = ""
     Next
   End If
   NewFlag(12) = 0                          'Set the NewFlag variable to let the
                                            'program know that this is no longer a new record

 End Sub
```

The procedure first checks for any -1 or N/A values in the database and displays a blank in the corresponding text box. If the value is not -1, the data is converted to text and formatted in a standard numeric format with commas; ###,##0. If a 0 is not placed in the last column of the format, a blank will be displayed when a 0 is encountered. Finally, if the NewFlag variable has been set to 1, signaling that this form was opened for a new survey record, all the text boxes are blanked out.

Saving Information To The Database

The next step after data is inputted and displayed in the electronic survey pages is saving this information. There are two procedures in the Module11.bas file that control the saving of information to the database, Update_Dynasets and Update2_Dynasets. Before either of these procedures are run, the dynaset variables are created with the CreateDynaset method. Both Update_Dynasets and Update2_Dynasets procedures save the information stored in each database variable to the dynasets. UpdateDynasets is used when an existing database record is being changed. The Edit method of the dynaset is used to allow the current record to be edited and copied to the copy buffer. The Update method of the dynaset is then used to save the contents of the copy buffer to the database. The following code comes from the Update_Dynasets procedure. This code saves changes made to the Annual Operating Statistics variables to the Annual Operating Statistics Table in the 1995 database. The code from Update_Dynasets that is not shown follows this same basic pattern and applies it to the other tables in the database.

```
Criteria = "[ANNUAL_OP_R_NO] = " & R_ID  'Sets the record matching criteria
                                          'Finds the first record with matching ID number
AnnOpStat_Dynaset.FindFirst Criteria

  For I = 0 To 13
    AnnOpStat_Dynaset.Edit   'Opens the current record for editing
       'Copies variable contents to the copy buffer
    AnnOpStat_Dynaset("CAR_ORG_TERM_ON_LINE") = AOSCars_Org_Term(0, I)
    AnnOpStat_Dynaset("INTRLND_CARS_ORG") = AOSCars_Org(0, I)
    AnnOpStat_Dynaset("INTRLND_CARS_TERM") = AOSCars_Term(0, I)
    AnnOpStat_Dynaset("BRIDGE_CARS") = AOSBridge_Cars(0, I)
    AnnOpStat_Dynaset("COMMODITY") = Commodity(I)
    AnnOpStat_Dynaset("STCC_CODE") = STCC(I)

       'Saves the contents of the copy buffer to the AnnOpStat Dynaset
    AnnOpStat_Dynaset.Update

       'Finds the next record that matches the record ID number - if there is any more
    AnnOpStat_Dynaset.FindNext Criteria
    Next
```

The second procedure for saving information, Update2_Dynasets, is used when a new railroad record is added to the database. The code is similar to the code used in the Update_Dynasets procedure. The main difference is that the AddNew method is used instead of the Edit method. AddNew clears the copy buffer to create a new record in the dynaset. When all the variables are copied into the copy buffer, the Update method is again used to save this information to the dynaset. The other major difference is that it is not necessary for this procedure to search through the database for a matching record ID, since a new ID is created for the new record.

1994 Functionality

Dentry 2.0 is designed to allow users to view their 1994 survey response and the code used when displaying 1994 data is similar to the code used to display the 1995 data. A command button named 1994 appears when a survey page is open. To view the 1994 response, the user clicks on the 1994 button. The background of the survey page will change and the fields of the 1994 survey will be displayed. The 1994 fields and 1994 data are read only; no copying, editing, or pasting is allowed with the 1994 data in Dentry 2.0.

After the 1994 button is clicked, the separate 1994 database, SLRD94.mdb, is opened with the Open_Dbase94 procedure in Module11.bas. Next, a railroad names list is generated, sorted, and placed on the frmRNames94 form. All of this code is nearly identical to the Open Record procedure found in the main menu of the Dentry program. The differences in code occur when a 1994 railroad is selected from the railroad names list. The double-click event on the list box runs the SelectRailroad procedure. The code in this procedure is identical to the SelectRailroad procedure used in the Open Record function except for one item. The SelectRailroad procedure used for displaying 1994 data calls Copy94 after the railroad has been located in the database instead of calling CopyDisplayInformation.

Copy94 uses the Scrollbar index value to determine what form is currently displayed in the electronic survey. After this has been located, all of the input boxes on that particular form are disabled, the background color is changed, and the year label in the caption is changed to 1994. Then a local Clear procedure for the current form is called to clear the corresponding 1995 variables. Finally, the variables from a snapshot of the related table in the 1994 database are displayed.

Once the 1994 data is displayed, a 1995 command button becomes visible on the toolbar and the 1994 button is disabled. When the 1995 button is pressed, the current form returns to its appearance just before the 1994 button was pressed. The form's input boxes are enabled, background color is returned to normal, input boxes are cleared, and the 1995 data for the railroad record originally opened is displayed in the input boxes. None of the 1995 database variables have been altered by the viewing of the 1994 data.

Dentry's Printing Activities

New functionality developed for Dentry 2.0 includes the print capabilities. Print functionality is enabled after the 1995 database is opened and can be accessed by the menu or by the toolbar. When invoked, the user is asked if any changes to the database need to be saved. After this response, the print form is opened. During the print form load event, the menu and toolbar functions are disabled to ensure that no changes can be saved to the 1994 database. Two options appear on the form: Select a Railroad and Exit. If the Select a Railroad command is pressed, a form with a list of 1995 railroads in the database appears using the same snapshot methods described for retrieving other railroad lists in Dentry. If a railroad in the list is selected by double clicking on it, the global RR_ID variable is set to the corresponding railroad's ID, and the Print form is reopened. Two new commands are enabled after a railroad has been selected: Print Preview a Section of the Survey, and Print the Entire Survey.

The Print Preview a Section of the Survey command opens the R&R Report file, survey.rp6. The destination is set to one, which will send an open report to a print preview window. Next an R&R form opens with a list of the different report sections. If a report section is selected and double clicked, the corresponding report section is printed to an R&R style print preview form to be viewed or printed.

The Print the Entire Survey command selects a railroad to print with the same functions used with the Print Preview a Section of the Survey command. If a railroad is selected from the list, the code sets the ReportName property in R&R to the first report, "Part I. RR & Customer Profile". The guage is set to visible with a value of zero. The Filter property of the report is set to the selected railroad's ID number. The R&R Action command to print the report is run. When the report is completed, the gauge value is incremented to signify that a certain percentage of the entire survey has been printed. Each remaining report is printed in succession following the same procedure as given above. When the entire survey has completed printing, the gauge will display 100 percent complete for a short period of time, then become invisible. The print form's display will show just the two command buttons, Print Preview a Section of the Survey and Print the Entire Survey. The following table lists each of the R&R reports that make up the entire 1995 survey.

Table 3.3: 1995 R&R Reports

1995 Annual Data Profile R&R Reports	
Part I. RR & Customer Profile	Part V. Financial Data
Part II. Roadway, Track, & Str	Part VI. Employees & Benefits
Part II (continued)	Part VII. Passenger Services
Part III. Equipment Inventory	Part VIII. Computers and Apps
Part IV. Annual Op. Statistics	Part IX. Comments
Part IV (continued)	

Dentry Installation Program Development

Once software development neared completion, the process of creating the distribution disks began. The Setup Toolkit shipped with Visual Basic 3.0 was not sufficient for developing the distribution disks. Editing existing files was necessary for a successful setup of Dentry 2.0 and is not possible with Visual Basic's Setup Wizard. Eschalon Setup Pro has this capability, plus other additional version checking features important to the success of distributing the Dentry 2.0 Software Package.

Several problems were encountered when creating the distribution disks. These issues are discussed in the following sections. The first issue considered deals with distributing an ODBC application such as Dentry 2.0. Many problems arose with attempting to get the correct ODBC Data Sources setup from the distribution disks. Another problem was determining what files needed to be distributed with Dentry 2.0. The final problem discussed deals with getting each individual 1994 database record saved as a separate but complete database for shipping with the program.

ODBC Setup

One of the major tasks in developing Dentry 2.0 is making the program portable across many different computer systems involving many different system configurations. One major aspect of a system configuration is Open Database Connectivity(ODBC) and Dentry 2.0 needs to use ODBC to access database information. This requires an ODBC Administrator to be installed, along with information about the ODBC drivers and the ODBC Data Sources. One difficulty with this is that a computer system configuration may already have the ODBC Administrator installed, while other systems may not. If a computer has the ODBC Administrator installed, it will need to have Dentry's ODBC Data Source and ODBC driver set up without interfearing with the current settings. If a computer doesn't have an ODBC manager, Dentry 2.0's setup program must be able to install the ODBC Administrator with the correct driver and Data Source.

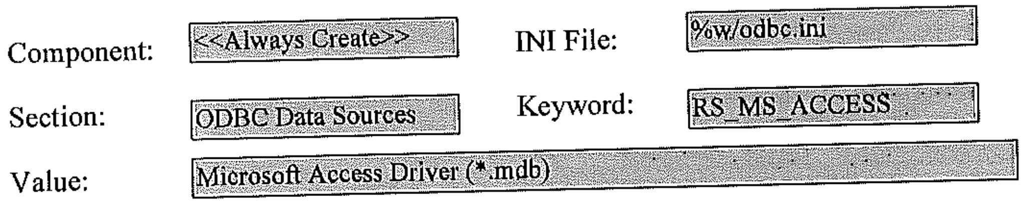
Eschalon Setup Pro is the distribution disk utility program utilized to help create the setup disks for Dentry 2.0, and address the ODBC issue. The distribution disks must contain the necessary database driver, initialization files, and ODBC Administrator files. An option, included with Eschalon Setup Pro, for each distribution file is Copy If Does Not Exist. This option should be set for the initialization files for ODBC; ODBC.ini, and ODBCINST.ini. If the ODBC Administrator program already exists on the current system, the installation of Dentry will not overwrite the existing initialization files. However, Dentry will still add its data source and driver to allow Dentry to access the Access database for printing purposes.

Adding the data source and driver requires functionality that comes with Eschalon Setup Pro. Eschalon contains a section that allows the creation of .ini file entries that can be added to an existing .ini file. This utility allows the specification of lines that will be added to an .ini file, or create the initialization file if it does not exist. The items required to fill in a line are the .ini File, Section, Keyword, and Value. The Section is the part in the square brackets in an .ini file, and the Keyword is the part to the left of the equal sign (=) in an .ini file. The Value is what is assigned to the keyword. For example, the Dentry installation needs to add the following line to the ODBC Data Sources section of the ODBC.ini file in the Windows subdirectory.

RS_MS_ACCESS = Microsoft Access Driver (*.mdb)

Eschalon Setup Pro can take care of this if the following information is entered in the input boxes on the Create .ini File Entries section of Eschalon Setup program:

Figure 3.4: Eschaelon INI File Entries



Component:	<<Always Create>>	INI File:	%w/odbc.ini
Section:	ODBC Data Sources	Keyword:	RS_MS_ACCESS
Value:	Microsoft Access Driver (*.mdb)		

After each of these text boxes are filled out properly, the insert item command is used to store the entry. Correct information has to be entered into the text boxes for each line added to the .ini files.

The .ini files that are required to be edited or created by Dentry are ODBC.ini and ODBCINST.ini. The ODBC.ini file is a Windows initialization file containing information about specific Data Sources. The ODBCINST.ini file is an initialization file that contains information about installed ODBC drivers. The Data Sources that are set up in the ODBC Administrator are located in the ODBC.ini file, while the ODBCINST.ini file contains the driver information used by the ODBC Administrator in setting up the Data Sources.

The minimum code needed in the ODBC.ini file to setup the RS_MS_ACCESS Data Source needed with Dentry 2.0 is:

```
[ODBC Data Sources]
RS_MS_Access=Microsoft Access Driver (*.mdb)

[Microsoft Access Database]
Driver=c:\WINDOWS\SYSTEM\odbcjt16.dll
DriverId=25
FIL=MS Access;
JetIniPath=odbcddp.ini
```

The minimum code needed in the ODBCINST.ini file to install the Microsoft Access Driver in the ODBC manager is:

```
[ODBC Drivers]
Microsoft Access Driver (*.mdb)=Installed

[ODBC Translators]
MS Code Page Translator=Installed

[Microsoft Access Driver (*.mdb)]
Driver=c:\WINDOWS\SYSTEM\odbcjt16.dll
Setup=c:\WINDOWS\SYSTEM\odbcjt16.dll
```

Once the ODBC Administrator is installed and the data source and driver are added to the correct .ini files, that particular data source will be ready for ODBC data connectivity, and the R&R reports can access the database for printing purposes.

Distribution Files

A number of files are required to be distributed with Dentry 2.0. Many of the systems files on the distribution disks will already be installed on many computer systems. These files are on the distribution disks only to cover the few cases where they are not installed on a computer system. The Dentry setup program automatically runs a check on each of these files and copies them to the system only if they are newer than the existing files, or if the files do not exist. There is a table in Appendix F that lists each file and a brief description of each file.

Individual Railroad Database File Creation

The 1994 database files for each participating railroad were sent out in a separate mailing on a single floppy disk. The separate mailing was done as a security measure against sending private information to the wrong railroad. The current form of the 1994 database contains all 229 railroads in one central database. The Dentry 2.0 1994 functionality required each individual railroad to be extracted from the database into one single record database. One method of accomplishing this task is to copy each individual record from the database and paste them to a blank database. However, this process is too time consuming. A better way was to create a program in Visual Basic to automate this copying process.

The program, Record94, begins by asking from which database file should the program extract a record. Next, Record94 takes as input the record number that matches the railroad record being extracted. Once the record number is determined, the Update Record command can be invoked.

Update Record first repairs the database using the RepairDatabase command. Next, the database is opened using the OpenDbase94 procedure, which is similar to the OpenDatabase procedure used in Dentry 2.0. The DelRecord function is run searching each table in the database and all the records not matching the record to be saved are deleted. The following code shows how DelRecord deletes all the records in the GenProf table except the one selected, *Record*.

```
If GenProf_Dynaset.EOF = False Then
   GenProf_Dynaset.MoveFirst
End If
Do While GenProf_Dynaset.EOF = False
  If Not (GenProf_Dynaset("R_NUMBER") = Record) Then
     GenProf_Dynaset.Delete
     GenProf_Dynaset.MoveNext
  Else
     GenProf_Dynaset.MoveNext
  End If
Loop
```

When the DelRecord function is complete, the database only has one record left, the one that matches the record number selected earlier. This database is again repaired using the RepairDatabase function and the program is completed. The repaired database is now the 1994 database file that is sent to the railroad corresponding to the selected record number.

The Append Database Program

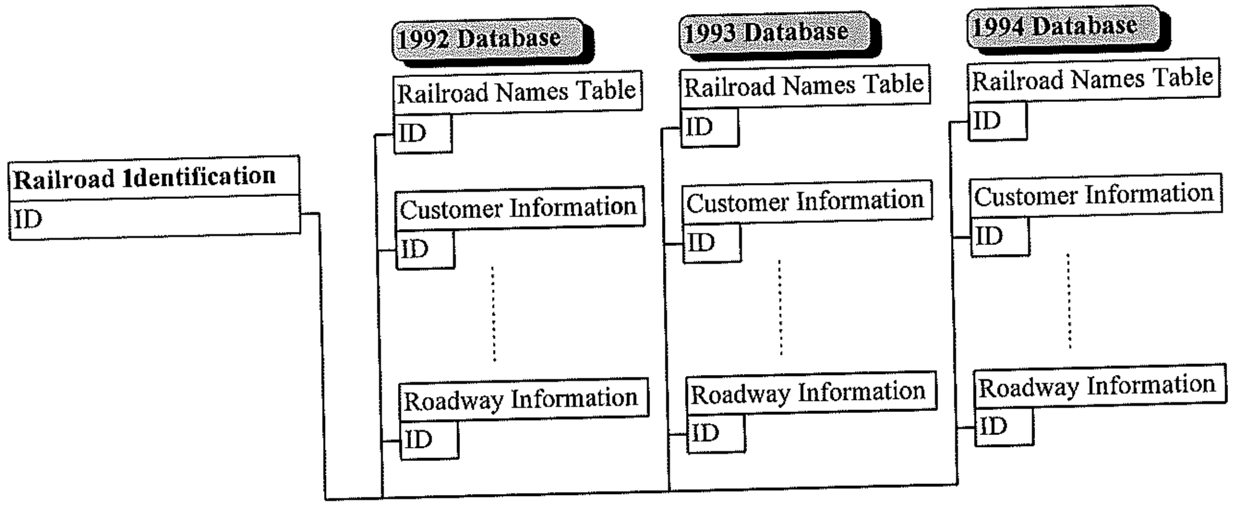
A Visual Basic program also was developed for inputting electronic data received for the 1995 data. The program, Append, was created to copy survey records from the database file returned to us on floppy disk to the main database. Once the program is run, the startup form is displayed with two options, Append and Exit. When Append is clicked, a form is displayed listing the railroad records on the floppy disk. To add a record to the main database, simply double click on the corresponding record from the list. This increments the last database record ID number by 1 and assigns this number to the new record to be added. The tables are then saved to the database with this ID number using the Addnew record Visual Basic function. After several seconds, a message appears stating that the record has been appended to the main database. It takes roughly 20 seconds for Append to add a new record to the database, compared to about five minutes for typing a survey in manually.

CHAPTER 4. FUTURE OF THE DATABASE

Future Database Design Techniques

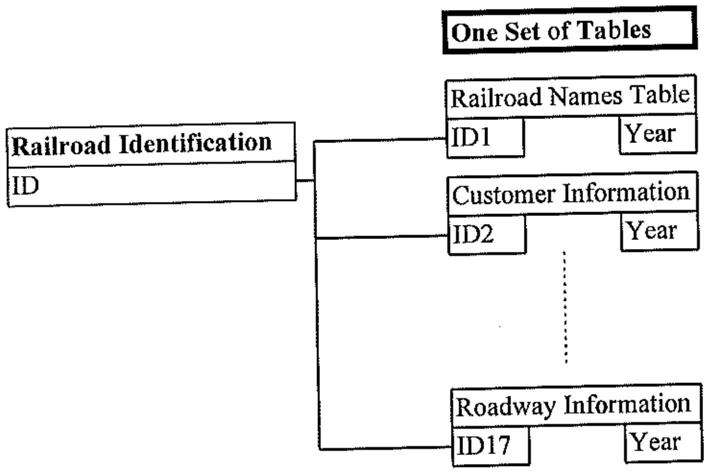
Storage and data entry for the database are currently in the Microsoft Access 2.0 format. There exists a separate database for each year of the survey, but one idea under consideration is the creation of a single, central database. This would involve appending data from each yearly database into one database. A new table would be created containing the railroad names and a permanent, unique railroad ID number, and each record ID number in all the tables would need to be replaced with the corresponding permanent, unique number. All tables in the central database would be linked to this central railroad ID number. There are several options for the design of the yearly tables and their relationship to the central railroad ID table. The following figures show several different scenarios on how the tables could be related.

Figure 4.1: Central Database Scenario 1. Each year contains 17 tables of data. In this scenario, each year will have a separate set of 17 tables.



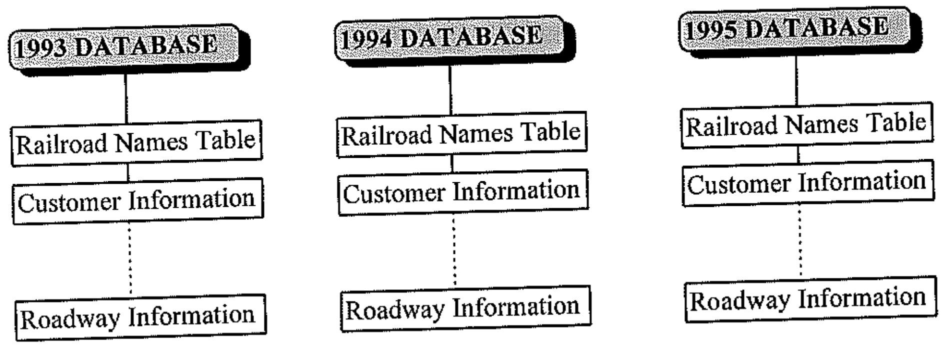
The main problem with this method is the large number of tables that will slow down database sorting, querying, and access. Data normalization is the process of eliminating redundant data within the database. To normalize the database design, tables common to each year should be combined into one table. Microsoft Access makes this process of combining tables relatively easy. A new field that will be necessary is a year field, which associates each record with the proper year. Figure 4.2 in scenario 2 represents these adjustments to the database.

Figure 4.2: Central Database Scenario 2. Each year contains 17 tables of data. In this scenario, each set of 17 tables will be combined into one set of 17 tables. The year of the survey will be distinguished by the new year field common to each table.



The railroad identification table has a one-to-many relationship with each table. This implies that each railroad will have only one record in the railroad identification table, but each railroad will have a different record in the 17 tables for each year they have responded to the survey. As data is added to the database each year, problems with the number of records in each table might arise and may cause problems with the number of records in each table. If the number of records get too large for a table and performance slows down dramatically, that particular table can be separated into several tables.

Fignre 4.3: Central Database Scenario 3. The final option is keeping a separate database for each year of the survey.



There are several advantages for keeping a separate database for each year of the survey. A central database containing all years of the survey data in one database, like in scenarios 1 and 2, may eventually cause problems. The number of tables will increase in size each year for scenario 1, and the size of the tables will increase in size each year for scenario 2. This will create a large database file whose size will eventually affect the speed of searching and querying in the database. Keeping the years as separate databases can be an effective method of storage and retrieval. Each year's database will be roughly the same size, and there will be no need to append each year's database tables to the central database. The data is arranged and organized for analyzing each year separately. Even trend analysis across yearly data can be done easily with separate databases. Features in Access and SAS allow users to easily merge variables from different years into one file to use for analysis.

This method requires each railroad to be assigned a permanent ID number. The permanent ID number would assist the database managers in keeping track of individual railroad data. This also would assist in querying among the different yearly databases. Each new survey year the ID number corresponding to a particular railroad would have to be manually added to the database and new railroads would be assigned new ID numbers.

Another item to track in the database is parent companies. There are several companies in the short line industry that own more than one railroad. A parent table containing the common information and a child table containing the member's railroad names could be formed. This would assist in tracking the number of railroads in the database.

The design of the tables should organize data so information can be retrieved most efficiently. When designing the tables, it is important to make sure that all information in that table relates to one topic. The current table setup follows this technique. If information happens to be repeated in a number of records, that common information should be moved to another table. In the current design, the annual operating statistics table can become large in size degrading search and query performance on the table. One method to address this problem would be to group similar information and create several new tables with these groups. However, too many tables also can hurt performance. Each time a table is opened, a file handle is used, which takes up memory. If the amount of information in the annual operating statistics table remains about the same, there should be no need to split up the table. In 1994, there were roughly 1,500 records in this table.

CHAPTER 5. FUTURE DATABASE ANALYSIS AND INTERFACE

The information collected from 1993, 1994, and 1995 data profile surveys are available, creating a need for the organization and analysis of multi-year information. The optimum resolution is to have a single interface integrating the functionality required for the entry, maintenance, analysis, and reporting of the data. To create an interface with these capabilities will require a review of several developmental software packages. The majority of this chapter consists of reviews of software products and their development and data analysis capabilities. Also included in this chapter is a review of several statistical techniques that may be utilized in the future analyses.

Visual Basic 3.0 vs. 4.0

To create an interface that utilizes several different software packages will require some of the OLE capabilities included with the Visual Basic development environment. If the Visual Basic development environment is going to be used for the central database, one question that needs to be answered is whether to use Visual Basic 3.0 or Visual Basic 4.0 as the interface's development software. Some of the new features included with Visual Basic 4.0 are:

- *Create DLLs.* DLL files can be created entirely within VB 4. Although they are not "true" DLLs, they do meet most of the needs of "true" DLLs. These DLL files are in-process servers. They share the same memory address as the application allowing for an increase in the speed of these functions.
- *Data Access Object Functionality,* formerly available only in Access.
- *Distributed OLE Objects.* Developers have the ability to create OLE objects that can be shared with other OLE capable applications.
- *Three Tier Client/Server Applications*
- *Database Improvements.* Improved Jet Database Engine and improved programmatic control over the engine.
- VB4 allows database operations in multiple workspaces and multiple database transactions running in parallel. This could be very beneficial if each year's survey is stored in a separate database.
- *Distribution Tools.* Conditional compilation enables programs to be deployed on both 16-bit and 32-bit Windows environments.
- *Windows 95 Controls.* Creation of official Windows 95 applications is possible.

- *Update VB3 Apps.* Automatically convert VB3 apps to VB4 apps. Their might be some VBXs that will not convert to VB4 OCXs, and some slight changes in coding.

VB 4.0's OLE Automation allows applications to use functions and manipulate other OLE objects like Microsoft Excel, and Microsoft Access. While variants in VB3 were only wrappers to primitive data types, they can now contain OLE object references. This implies that large, complex objects, such as an entire range on a spreadsheet, can be passed in a single variant instead of iterating across all rows and columns to determine the cell contents. Object-oriented classes can expose an application's functionality to other applications through OLE Automation. Object-Oriented Programming(OOP) need not be used in VB 4.0, but, according to Microsoft, software development should be easier to design, implement, test, document, and maintain with OOP.

One major problem with VB4 is that its runtime files are massive for 16-bit applications - 300k to 1.8MB. For 16-bit applications, VB3 still has a smaller runtime and generates a smaller executable file. The 32-bit VB4 runtime files are smaller because much of the OLE component is already built into the operating system. Also, VB applications created for 16-bit operation cannot utilize all the new features of the 32-bit Jet 3.0 database engine. If the database administrators ran the interface on a 32-bit machine, the full features of VB4 and the new 32-bit jet database engine could be utilized for the central database interface.

Windows API function calls can be slightly different with 32-bit vs. 16-bit DLLs. For applications that are to be developed for use only in Windows 3.1, API functions used are declared in WIN31API.TXT. For applications that are to be developed for use only in 32-bit environments, API functions used are declared in WIN32API.TXT. For applications that are developed to work on both 16-bit and 32-bit platforms, the conditional compilation feature for making API callscan be used. The conditional compilation feature is just a logical check on what system is being used. If the 32-bit system is being used, then use the 32-bit API function call, otherwise the program uses the 16-bit

API function call. The following sample code shows how to implement the conditional compilation feature of Visual Basic 4.0.

```
# If Win32 Then          '32-bit Windows.
        Declare Function OSGetClassName Lib "User32" Alias _
        "GetClassNameA" (ByVal hWnd As Long, ByVal lpClassName As _
        String, ByVal nMaxCount As Long ) As Long   '32-bit Window handles are passed
                                                    'as Long Integer values
# Else                   '16-bit Windows.
        Declare Function OSGetClassName Lib "User" Alias _
        "GetClassName" (ByVal hWnd As Integer, ByVal lpClassName _
        As String, ByVal nMaxCount As Integer) As Integer   '16-bit Window handles are
                                                            'passed as Integer
# End If
```

Another problem with using VB4 is the learning curve. The developers would need to learn about the new processes and features for coding in VB4. If OOP techniques are going to be utilized, developers would need to attain a working knowledge of object oriented programming. For a succesful central database interface, the developers also will need to acquire a good understanding of the functionality and speed of the OLE Automation technologies provided with VB4.

Visual FoxPro

Visual FoxPro is another software package providing a front-end development environment for a central database interface. This program has general database functions similar to Microsoft Access and can read MS Access tables. Defining tables, queries, and reports follows the general standards used by database management software such as MS Access and Paradox. The main advantage to Visual FoxPro is the OLE capabilities and the ability to create distributable database interface programs. The following is a list, retrieved from Microsoft's Web page, of features included with Visual FoxPro 3.0.

- Offers a visual development environment for rapid application development
- Allows developers to create classes visually, as well as program them in the Command window
- Provides object-oriented programming capabilities, including inheritance, encapsulation, subclassing and polymorphism
- Provides full-featured debugging with watch variables, Trace window and break points

- Leverages the capabilities of the Windows 3.1, Windows NT™ and Windows 95 operating systems
- Ships with a rich set of standard controls, including the following: Grid, Label, Check Box, Combo Box, Command Button, List Box, Shape, and PageFrame
- Allows developers to distribute solutions they have created to take advantage of the capabilities of the Windows version 3.x, Windows NT, and Windows 95 operating systems
- Provides a Setup Wizard, which creates a set of disk image directories containing all the files needed to install an application
- Contains a built-in Class Browser for creating, manipulating and reusing classes within an application
- Allows the creation of standalone .EXEs that are royalty-free and unlimited in distribution
- Contains additional OLE Controls component software, enabling developers to build more robust, extendible solutions, including the following: Outline Control and Messaging Application Programming Interface (MAPI) controls
- Contains the full online Win32® API programmers' reference to provide the information that developers need to build powerful 32-bit solutions (CD-ROM version only)
- Data Access and Client-Server Support
- Integrates client-server support to popular back-end databases, such as Microsoft SQL Server™, ORACLE® 6 and ORACLE7™, using 32-bit Open Database Connectivity (ODBC) version 2.0 drivers
- Includes ODBC desktop drivers to Btrieve®, dBASE®, Microsoft Excel, Paradox™, Microsoft Access, and other FoxPro data
- Features visual design of database connections, local and remote views
- Includes Upsizing Wizard to adapt data from Visual FoxPro to Microsoft SQL Server back ends
- Rich OLE Support (Formerly Called OLE 2.0 Support)
- Includes OLE Container Control for using OLE components in applications dynamically at run time
- Includes OLE Automation support to "drive" other applications correctly from Visual FoxPro, such as sending FoxPro data to a Microsoft Excel spreadsheet or to control other OLE components such as Microsoft Word documents
- Features OLE Documents object control on the toolbar for quick drag-and-drop access to pre-built components such as the Microsoft Excel charting object
- Allows full OLE in-place editing of components from other applications embedded in Visual FoxPro solutions, such as editing a Microsoft Excel chart without leaving Visual FoxPro
- Contains additional OLE Controls that help add and extend functionality in Visual FoxPro
- Offers ability to open external API libraries or a procedure file
- Contains international tools for developing applications for international use
- Contains a powerful Help compiler for authoring Help files for use in applications
- Includes a GENDBC.PRG for generating a program that can re-create a database
- Contains hundreds of additional graphics files for use in Visual FoxPro solutions
- Includes FOXTOOLS, a Visual FoxPro API library that exposes Windows Dynamic Link Libraries (DLLs) for use in Visual FoxPro
- Includes ImagEdit, a tool that allows developers to view and edit bitmaps, cursors and icons
- Contains a Win32 programmers' reference to provide developers the information they need to build robust 32-bit solutions
- Contains the following printed documentation: Developer's Guide, Installation Guide, Quick Reference Guide and User's Guide

- Contains the following printed documentation: Language Reference and Professional Features Guide

Visual FoxPro is designed specifically to build database applications, whereas Visual Basic is designed for general purpose applications. Since Visual FoxPro is optimized specifically for database applications, it would seem like an appropriate tool to use for the central database and quite possibly for the data collection.

Before Visual FoxPro is considered as a possible central database development tool, the developer has to be assured this product will continue to be supported by Microsoft. With Microsoft Access taking up such a large chunk of the database development market, many developers think that Visual FoxPro will not see further development. According to an article in Microsoft's Web page, both Access and Visual FoxPro databases are important to Microsoft. Neither Visual FoxPro's Xbase language, or Access's Basic language are going to be abandoned by Microsoft.

Visual FoxPro 5.0 for Windows is scheduled to be released in fall 1996. This version is supposed to add new connectivity features, the ability to create OLE automation servers, full support for ActiveX controls, an enhanced development environment, a smaller footprint, and greater performance. Then work will begin on the next version of Visual FoxPro. First, Visual FoxPro will be developed as a separately maintained tool and later as part of a unified development environment which will serve as a common front-end for FoxPro, Visual Basic, Visual C++, and other Microsoft tools. Developer Studio, the development environment that is already a part of Visual C++, is expected to serve as the unified front-end for the tools. The following is a quote from Microsoft responding to the rumors that 5.0 will be the last version of Visual FoxPro developed. "After version 5.0 ships, Microsoft will start working on the next release of Visual FoxPro. Visual FoxPro will remain a great tool for high-end database development today and in the future" (Nielson & Muglia, 1996).

MS Excel's Capabilities

Microsoft Excel 5.0 has been used in past analyses of information from the central database. The main reasons Excel is used are for its graphing capabilities, and ability to quickly obtain means, medians, and counts across tables. Graphs of variables can be easily created and ported to a wordprocessing document, and excel formulas are utilized for quickly computing general descriptive statistics. To provide further statistical functionality, several statistical add-ons are available for Excel.

The statistical analysis add-on that comes with Microsoft Office has minimal performance capabilities. General descriptive statistics can be computed fairly easily, but the output is not presented in a clean, user controlled fashion. For example, histograms can be created using the data analysis add-on, but there is a lack of control over the appearance of the histogram. There are, however, many add-on statistical packages for Microsoft Excel that could be utilized. These add-ons include functionality for data mining, data visualization, model building, and forecasting.

The querying functionality across Microsoft Excel worksheets is not adequate to run a thorough analysis on the survey data. Microsoft Excel will still be used for analyzing the 1995 data profile. However, if an adequate statistical add-on is not found, or if one of the following statistical packages is brought into use, Microsoft Excel will have a diminished role in the analysis of the survey.

Analysis Of Statistical Software Packages

SAS JMP 3.1 For Windows 3.1 Or Windows95

SAS JMP 3.1 for Windows is a statistical discovery software package and is designed as a statistical visualization tool. Graphical visualization of the data is the main operation of JMP software. JMP allows the user to manipulate the raw data in a spreadsheet presentation similar to Microsoft Excel and graphs are generated easily by choosing the needed variable or variables. General descriptive statistics and sophisticated model fitting options for intuitive exploration also are included in JMP 3.1. Additional features included with JMP software that would be of use to this project are, logistic regression, data management, graphing, goodness of fit tests, and the ease of use of the program. Two features not included in the program that would be beneficial are time series analysis and OLE capabilities.

STATISTICA

STATISTICA for Windows integrates statistical data analysis, graphics, and database management activities into one program. There are hundreds of types of two- and three- dimensional graphical displays, including special four-dimensional graphs, multidimensional graphs, categorized multigraphs, icons, layered graphs, and many other specialized procedures supported in STATISTICA. Detailed customization facilities and tools also assist the user in tailoring graphs to specific needs. Support is included for server and client mode OLE. Due to the fact that STATISTICA supports MS Windows DDE/API interface standard, one can build their own STATISTICA commands or complete extensive STATISTICA scripts into macros created within and run from other applications such as MS Excel and MS Word. An Excel macro may include SCL commands which call STATISTICA from within Excel, perform a specified set of analyses, and then transfer and integrate specific parts of the output and/or graphs created by STATISTICA with the current Excel worksheet or workbook.

In OLE server mode, STATISTICA graphs copied to other applications maintain their links to STATISTICA, and can be customized and edited using all STATISTICA tools by double-clicking on the graphs in the client application. A second option involves the use of Paste Link. When Paste Link is used, the graphs will automatically be updated in the client application when the source STATISTICA graphs change. In OLE client mode, STATISTICA can serve as a client for other application objects. Spreadsheets, wordprocessor documents, drawings, and graphs can be embedded or linked into STATISTICA's graphics documents. OLE objects also can be embedded up to the fourth order, implying objects can be embedded that already contain other OLE objects.

STATISTICA contains an in-process editor. The statistical output is placed in this editor in the rich text format that can be edited directly within STATISTICA or any major wordprocessor. The STATISTICA ODBC interface allows you to integrate data from multiple tables (from source databases such as MS Access) into a single STATISTICA data set. This option would be useful for merging and analyzing data from several years of the survey.

Relevant statistics supported by Statistica:

- descriptive statistics
- exploratory data analysis
- correlations
- t-tests
- distribution fitting
- reliability analysis
- cluster analysis
- log-linear analysis
- nonlinear estimation
- time series analysis and forecasting
- regression based forecasting techniques
- sampling plans

S-Plus

S-Plus is based on the S language, the only modern, object-oriented language developed specifically for data analysis at AT&T Bell Labs. S-Plus contains over 1,650 functions for performing and managing data analysis. S-Plus statistical functions include basic statistics, multivariate functions and graphing, regression and ANOVA, survival analysis, time series, and interactive graphics and data visualization to help determine which analysis methods to apply. There is a program called S-PLUS for ARC/INFO that integrates the statistical and data analysis tools of S-PLUS with the GIS capabilities of ARC/INFO.

StatGraphics Plus

Statgraphics Plus for Windows has many graphics and analysis capabilities, but it does not all come in one package. Separate statistical functions such as time series and quality control have to be purchased as separate, add-on modules. This program is not recommended due to the requirement to purchase separate software packages for each different statistical function used.

Overview Of Software Reviewed

Of all the software packages reviewed, SAS JMP, STATISTICA, and S-PLUS are the three most impressive. All three are formidable Windows-based programs with a wealth of statistical functionality. Several questions remain as to which software is faster, allows the largest data sets, is the most user-friendly, is OLE capable, and is easy to use. Since SAS JMP is not OLE capable, the best package for our situation is between S-PLUS and STATISTICA. STATISTICA's OLE, database, and wordprocessing features are more impressive than the features of S_PLUS. If STATISTICA is relatively the same speed as, or faster than S-PLUS, STATISTICA would be the best choice of analytical software for this project.

Statistical Analysis Techniques Utilized

There exists a multitude of statistical analysis functions and techniques, of which many could be applied to the information in the central database. Data mining and data visualization techniques, estimation techniques, forecasting models, time series analysis, nonparametrics, and different descriptive statistics can all be utilized with this data. If the results from these analysis are found to be reliable and informative, they could be possible additions to future database reports. The next few sub-sections give overviews of the possible analysis that can be done with the data in the central database.

General Descriptive Statistics

All of the statistical software packages described above have extensive data exploration capabilities. The goal of any statistical analysis is to extract meaningful and interesting information from the data. Data visualization techniques are generally the first analysis completed on any data set. Visualizing the data through graphs gives a statistician an idea of the structure of the data. Data smoothing, categorizing of data, plotting confidence intervals and confidence areas, spectral planes, distribution fitting, and 3-D graphs all are functions of exploratory data analysis that can be used on this data.

General descriptive statistics and industry total estimations should be made for each variable in the survey. Different ratios and correlations should be analyzed for detecting important information. After an in-depth look at each variable, it may be discovered that a variable provides information that has limited value and the importance of each variable could be ranked. Several questions that do not provide the industry with pertinent information could be considered for exclusion from future surveys.

Jackknife And Bootstrap Estimators

For the 1994 data profile, industry totals were not actual and complete industry totals. They were only the totals for the 233 railroads that responded to the survey. For a more accurate view of the contributions of the short line industry, estimated totals and confidence intervals should be computed to present a more complete view of the entire industry. Two methods of estimation that might provide better insight into the industry as a whole are called bootstrap and jackknife estimators.

The jackknife estimator is a resampling method whose main goal is reduction of the bias or variance of the estimate. This procedure was introduced by Tukey in 1958 and involves computing the average of a statistic when one value is removed from the sample. If this procedure is repeated for each value in the sample and the average of these statistics is computed, the resulting statistic will have a reduced bias. This process is described in mathematical notation as follows. If we let

$$\hat{p}(i) = \hat{p}(x_1, x_2, \mathrm{K}\ , x_{i-1}, x_{i+1}\mathrm{K}\ , x_n$$

be the value of the statistic when x_i is deleted from the data set, and let

$$\hat{p}(\bullet) = \left(\frac{1}{n}\right)\sum_{i-1}^{n}\hat{p}(i)$$

The Jackknife formula for the standard deviation is

$$\hat{\sigma}_j = \left[\left(\frac{(n-1)}{n}\right)\sum_{i-1}^{n}\left(\hat{p}(i) - \hat{p}(\bullet)\right)^2\right]^{12}$$

The jackknife can be applied to any statistic that is a function of n independent and identically distributed variables.(Efron and Gong, 1983)

The bootstrap procedure introduced by Efron in 1979 is a resampling technique similar to the jackknife technique. This method generates the sampling distribution through successive sampling from a data set. For example, to estimate a population's sampling distribution from a sample of size n = 40, first select the 40 data points, with replacement, randomly from the database. Next, calculate the average, and repeat this process, several thousand times. The resulting bootstrap distribution gives a good representation of the underlying population distribution. The reason that this occurs is because the sample contains information about the underlying population. A more detailed review of the bootstrap method follows:

There are $x_1, x_2, \mathrm{K}\ , x_n$ independent observations from an unknown, underlying distribution, call it F. F can be estimated by the empirical probability distribution $\hat{F}$, by assigning a probability of $1/n$ to each observed data point x_i. Draw a bootstrap sample X_1^*, X_2^*, K, X_n^* by independent random sampling from the empirical distribution, $\hat{F}$. In other words, take n random draws, with replacement, from the sample $\{x_1, x_2, K, x_n$. For the second step, compute the bootstrap replication, $\hat{p}^* = \hat{p}\left(X_1^*, X_2^*, K, X_n^*\right.$. In other words, compute the average of the bootstrap sample to get $\hat{p}^*$. Repeat this process a large number of times, say b times, to obtain independent bootstrap replications $\hat{p}^{*1}, \hat{p}^{*2}, K, \hat{p}^{*b}$. Then approximate the standard deviation of the bootstrap replicates by

$$\hat{\sigma}_b = \sum_{t=1}^{b}\left[\left(\hat{p}^{*t} - \hat{p}^{*\bullet}\right)^2 \Big/ (b-1)\right]^{1/2}, \quad \hat{p}^{*\bullet} \equiv \frac{\sum_{t=1}^{b}\hat{p}^{*t}}{b}$$

(Efron and Gong, 1983)

Like the jackknife, the bootstrap can be applied to any statistic that is a function of n independent and identically distributed variables. In most cases investigated by Efron and Gong, the bootstrap performed better than the jackknife. The main reason for using the jackknife estimator is that it is less complicated computationally; however, SAS has a program for computing the bootstrap estimate of variable bias that can easily be applied to the variables in the short line database. Once a bootstrap estimate of the bias for a statistic is computed, confidence intervals for industry totals can be derived with a minimal amount of bias.

Modeling

There exist many different modeling techniques that can easily be applied with the use of current statistical software packages. Simple linear regression, logistic regression, time series analysis, factor analysis, principal components, cluster analysis, and reliability analysis could be beneficial tools in the analysis of the database. Reliability analysis would likely have the most relevant functionality for the analysis of the survey. Reliability analysis includes many procedures for the development and evaluation of surveys and questionnaires.

The other multivariate modeling techniques can be used to set up a classification model. A classification model could tell what type of railroad is being looked at. Distinct terms differentiate regional and local-line haul railroads, but a switching and terminal railroad is self-designated. An adequate classification model could be setup to these types of railroads. Switching and terminal railroads generally will not own much track, have short average lengths of haul, and will not own many cars. Several classification models could be tried based on this information.

Recommendations For The Central Database Interface

The usefulness of an interface to the central database is evident to the database administrators. The grouping of all the central database functionality into one interface would truly enhance the system. The information going into the database would be consistent and organized, and the timeliness of retrieving information from the database also would improve.

Currently, it is possible to write a program that could automate preset tasks accomplished through the integration of several software packages. One of the major functions for such a program is the linking of graphs, statistics, and report template fields. Anytime the database is changed, the graphs would be updated, the statistics recomputed, and the report template fields updated. In other words, the information in one file would remain consistent with that same information in another file. There would no longer be a need to manually update each SAS data set, Excel spreadsheet, graph, and Access database when a change occurs.

This program would require significant resources to develop. Through the use of OLE Automation, the administrators could conduct more application specific functionality with the database. The interface would have the option to go directly to one of the software components to utilize its functionality while not leaving the interface. If an administrator needed additional functionality, it could be programmed into the interface. Development of this program would be an evolutionary process.

The creation of a central database interface with Visual Basic incorporating functionality for database editing, graphing, statistical analysis, and reporting would involve extensive development. The following is a quote taken from an electronic book on the Internet called <u>Special Edition Using Visual Basic 4</u>:

"Eventually you will need to build an application that uses more than one Microsoft Office application. This project will probably involve a large programming team and will require weeks (or even months) to complete." (Webb et. al. 1996)

If an interface was to be developed with Visual Basic, it would be viewed as a long-term project, adding functionality one step at a time. If the features of other development platforms, such as Visual FoxPro, are easy to implement, they may require less development time than Visual Basic for creating an interface to the datbase.

CHAPTER 6. FUTURE ELECTRONIC DATA COLLECTION METHODS

Visual Basic Program Development

Dentry has been developed in Visual Basic 3.0 and continued development in Visual Basic requires a decision on whether to switch to the new version, 4.0. The major differences between 3.0 and 4.0 have already been discussed. This section is devoted to addressing the concerns with switching to VB4 for the 1996 Dentry software package.

A major concern is the amount of resources it would take to switch to VB4. The development team has valuable experience in developing software in VB3 and the switch to VB4 will create new challenges. While it may be adequate to just fine tune the VB3 version of Dentry for the 1996 survey, the following list presents several reasons to develop Dentry in the VB4 environment.

- VB4 allows 32-bit programming and applications that take advantage of Windows95 features
- Runtime speed of the program will increase on 32-bit machines
- Creating a help system module is made easier with added features in VB4
- Porting the program to VB4 will better prepare the program for future VB upgrades

The new underlying VB4 language engine, Visual Basic for Applications, is designed to be fully backward-compatible with VB3 code. When a VB3 project is loaded into the VB4 development environment, the project automatically gets converted to VB4. The Visual Basic control files (vbx) also get converted to their .ocx equivalent, if one exists. However, new features in VB4 might create conflicts with existing VB3 code. Several key coding differences between VB3 and VB4 are listed below:

- Boolean properties used to be expressed as integer type. In VB4 a new data type was created, Boolean, whose values are True or False.

- Database transactions were always global in VB3. In VB4, transactions occur within Workspace objects. VB4 now contains the IsolateODBCTrans property to isolate multiple transactions involving the same database.
- In VB4 the default property name must be used when refering to a database object. In VB3 each data access object had a property that could be used for storing and retrieving values just by refering to the object, without using the property name. The following VB3 code

  ```
  s = Db.TableDefs(i)
  ```

  needs to be changed to

  ```
  s = Db.TableDefs(i).Name
  ```

- Arrays are reallocated differently in VB4. VB now temporarily locks an array when any element of that array is passed by reference to another procedure. Some code involving arrays that worked in VB3 may no longer work in VB4.

Additional concerns with porting Dentry 2.0 to VB4 deal with space and system resources. The runtime files for the 16-bit version went from 300k to more than 1 MB, a relatively large jump in size. Dentry's distribution files already consume three disks, and adding a help system to Dentry for 1996 will also add to the program's size. If switching to VB4 adds enough size to the program to add another disk or two to the setup program, the switch may not be justified.

One of the main reasons for switching to the VB4 development environment is that it is 32-bit capable. The conditional compilation feature of VB4 allows single program code that can be deployed on both 16-bit and 32-bit Windows systems. The 32-bit system users could take advantage of the increase in speed with a 32-bit application, while the 16-bit system users could still run the program on their system. This option would be nice, but might not have that much impact on the running of Dentry unless some 32-bit capable features are added to the existing Dentry program. Also, it would not be necessary to develop a 32-bit capable program since Windows95 is backward compatable with 16-bit software programs.

File Management

The file management capabilities with Dentry 2.0 need improvement. One function that the design team will review is allowing the user to be able to specify what directory to use for software installation. Improved file management would include an option for saving the database to the hard drive, or renaming the database. Also, instead of sending a blank Access database with the distribution disks, Dentry should be able to create its own database. This would assist in reducing the number of distribution files required to be sent out with Dentry.

Previous Year's Data

Several data viewing options can be added to the program to improve the transfer of unchanged survey information from the previous year's survey. Split screen view of previous year's data, allow copying and pasting of data, viewing of each year's survey data, viewing of industry totals from previous years, graphing individual railroad variables across time, and viewing the previous year's Data Profile are all database functions that would be useful. The problem again lies in how much larger this makes the program. These features will require more database files, more visual basic control files, and possibly more .dll files. Many consequences exist if Dentry 3.0 requires six or seven distribution disks disks instead of two or three disks. Railroads may be inhibited by the size of the software, and the cost of distributing the software will go up. The design of the 1996 version of Dentry requires optimizing the number and the size of files distributed. There may be some files on the Dentry 2.0 distribution disks that are not necessary to distribute, and possible alternative files should be considered.

Printing Issues

The print features new to the Dentry software package need some refinement. The R&R Report Writer VB control did an adequate job of printing out the reports. The output was nearly identical in appearance to the paper survey. However, one of the printing issues is the requirement to create so many different reports to represent the entire survey. Printing the survey would be much faster if R&R could create one single report for the survey. Each time a report is printed, the R&R runtime is loaded into memory, then unloaded from memory. A report generator that can contain the entire survey on one report would provide the best solution to Dentry's printing process.

The user friendliness of the print process also can be improved. A totally reformatted form is recommended for the 1996 version of Dentry. The print menu design and print functionality need refinement, and a new layout of the print commands and print menu will be considered. Also for 1996, an improved gauge measuring the print process will be utilized.

Another downfall to the R&R Report Writer control is the large number of files required to run the reports. All of the R&R associated report writing files take up around 1.8 MB of space, and the required ODBC administrator sent out takes an additional 1.3 MB of space. A better option for printing the survey would be one that does not require the conditional installation of an ODBC manager. If the next version of Dentry is created with Visual Basic, the design team will need to look at alternative report writer software.

Help System

A help system is an important feature scheduled to be added to the 1996 electronic survey. A help system for Dentry would basically contain the contents of the survey glossary, a search mechanism, and technical support informaiton. With the add-ons available to VB4, this should not require a significant programming effort. The only issue will be the added size to the distribution disks as help files generally make up a large portion of a software package. The time consuming part in developing the help system will be the organization of the help information.

Visual FoxPro

Another option for developing the 1996 electronic survey is the use of Visual FoxPro. Visual FoxPro is designed specifically for developing database applications and Visual FoxPro might be more useful for developing the electronic survey than Visual Basic. Development features of Visual FoxPro that could prove useful in creating a stand-alone survey include: database control, report creation, interface design, help system creation, setup wizard, rapid application development, object-oriented programming capabilities, full-featured debugging, and leveraging the capabilities of the Windows 3.1, Windows NT™ and Windows 95 operating systems. Consideration will be given to switching to the Visual FoxPro development environment.

Internet Possibilities

The internet could also be used for retrieving survey information. With the use of HTML code, a web page can be created for the survey. The VBScript language, which is similar to Visual Basic, can be used to control the survey text boxes and command buttons. Currently, there is a shareware program called ActiveX that can make the creation and management of a web page easier.

Placing the survey on the internet would be fairly easy with the use of ActiveX controls as ActiveX allows an individual to develop interactive web pages very quickly. A program called the Microsoft ActiveX Software Development Kit comes with many ActiveX tools to assist in developing web pages and included with the SDK is the Microsoft ActiveX Control Pad. This tool allows the building of ActiveX web pages using ActiveX controls in a development environment similar to Visual Basic. The ActiveX Controls are the interactive objects in a web page that provide interactive and user-controllable functions. Thousands of ActiveX controls, which are actually OLE objects, already exist, and many more are being developed. Another tool, ActiveX Documents, enable users to view non-HTML documents, such as Microsoft Excel or Word files, through a web browser. This tool would be very useful in allowing a railroad to view graphs, statistics, and documents related to the Annual Data Profile. Finally, the ActiveX Server Framework tool provides a number of web server-based functions such as security and database access. Security would be one of the most important features in developing a web page for the annual survey, considering the highly confidential material in the ASLRA central database. The Server Framework tool provides some security measures for accessing information in the database. One stipulation with using the ActiveX SDK is that Windows95 is required to develop and maintain a web site.

Developing the web page through the use of ActiveX controls and VBScript would take some effort. Another program may make the development of a web page simpler. A program called Microsoft Front Page provides WYSIWSG editing and wizards to assist in creating web pages. This application provides everything needed to get a web site up and running with very little programming. Specifications for this software require at least a 486 with 16 MB of RAM and Windows95 or Windows NT operating system.

The development of a web survey site depends on the railroads that will use the survey. First, are there enough railroads that have access to the internet to warrant placing the survey on a web page? According to preliminary results from the 1995 Data profile, only 26 out of 137 railroads had access to the internet in 1995. Currently, there are not enough railroads with access to the Internet to justify deploying resources to the development of a survey web site.

Another issue to consider is whether the railroads are comfortable with the security of information placed on the internet. Many railroads might not place private information on the internet. They would be placing a lot of trust in the security measures utilized. Several new questions addressing the issue of adding the survey to the internet could be added to a future survey. The response would give a better idea of how many railroads would support an internet-based survey.

EDI Possibilities

This option depends on how many railroads have access to a modem, and how much functionality EDI will provide. According to preliminary results of the 1995 survey, about 80 percent of the railroads stated they have 486 computers or more, 55 percent stated they have a Fax/Modem, and 74 percent stated they currently use EDI. Since the majority of the short line railroads have 486 or more computers and are currently using EDI, most of them have access to a modem. However, Matt Reilly of the ASLRA estimated that only about 80 short line railroads would actually utilize an EDI based survey.

EDI would most likely give the railroads a better sense of security than the internet. Future versions of Dentry could add an EDI communications control that would allow them to FTP the completed database directly to a server at UGPTI. Different Visual Basic controls do exist for electronic data transmission and will be considered for future implementation into the project.

The internet web page and the EDI functionality added on to Dentry would both be useful additions to the project. The minimum amount of software needed to develop a web page is all royalty-free shareware. Current Visual Basic EDI controls are available and relatively inexpensive, and the next version of Visual FoxPro may support EDI functionality. A web site would have many advantages over the EDI functionality. It would allow the development team to add features like graphics, display of yearly analysis documents, efficient collection of comments and questions from short lines, and links to other related railroad sites. A more extensive EDI software package and much more development would be required to achieve EDI functionality similar to that capable via a web site.

The security of the information and the lack of short line railroad access to the internet are the only two real advantages that an EDI program has over a web site. If EDI was used to just transfer the completed database to UGPTI, there would be virtually no security risks. On the other hand, access to a web site can be attempted by anyone on the Internet. The encryption and decryption options accompanying the web site development software will have to be significant to assure respondents that hackers will not be able to gain access to private database information. Since security is an extremely important issue with this information, EDI is currently the most reliable method for collecting the survey data.

BIBLIOGRAPHY

Efron, B. Gong, G. 1983. "A Leisurely Look at the Bootstrap , the Jackknife, and Cross-Validation". The American Statistician, Vol. 37, No. 1, pp 36-45

Webb, J, McKelvy, M., Martinson, R., Maxwell, T. 1996. Special Edition Using Visual Basic 4. HTML version, QUE corporation, Indianapolis, IN, http://www.mcp.com/que/developer_expert/sevb4

PC Week Online. 1996. "Visual FoxPro 5.0 Slated To Ship In Late October". Ziff-Davis Publishing Company, http://www.pcweek.com/news/0819/23efox.html

Microsoft Corporation. 1996. "Microsoft Visual FoxPro 5.0 Overview". Microsoft Corporation, Redmond, WA, http://www.microsoft.com/VFOXPRO/vfinfo/fox5.htm

Nielsen, T., Muglia, B. 1996. "An Open Letter to the FoxPro Community". MicroSoft Corporation, Redmond, WA, http://www.microsoft.com/VFOXPRO/tovfpcom.htm

APPENDIX A: SAS CODE FOR THE UNIVARIATE PROCEDURE

```
options ls=79;
data customer;

/***     Imports the customer table with fields delimited by commas
infile 'customer.txt' delimiter = ',' firstobs=7;

/***     assigns a varable name for each table field
input ID Total coal farm chem food nonmtmin trans lumber pulp petro
stone met pmet waste oth list$;

/***     Sorts the data by ID
proc sort;
by ID;

/***     Creates a new data set that contains railroad type and region information
data two;
infile 'info.csv' delimiter = ',' firstobs = 2;
input ID type$ region$;
proc sort;
by ID;

/***     Creates a new data set that combines the customer data with the type & region /***
         information
data combo;
merge customer two;
by ID;

/***     runs the univariate procedure on the Total field for all regional railroads
Proc Univariate;
where type = 'Regional';
var Total;

/***     runs the univariate procedure on the Total field for all West/Southwest railroads
Proc Univariate;
where region = 'West' or region = 'Southwest';
var Total;
```

APPENDIX B: SAS OUTPUT FOR THE UNIVARIATE PROCEDURE

```
The SAS System                 1


               Univariate Procedure

Variable=TOTAL

                 Moments

        N               17         Sum Wgts      17
        Mean            230.1765   Sum           3913
        Std Dev         343.5723   Variance      118041.9
        Skewness        3.716487   Kurtosis      14.52232
        USS             2789351    CSS           1888670
        CV              149.2647   Std Mean      83.32852
        T:Mean=0        2.762277   Pr>|T|        0.0139
        Num ^= 0        17         Num > 0       17
        M(Sign)         8.5        Pr>=|M|       0.0001
        Sgn Rank        76.5       Pr>=|S|       0.0001


               Quantiles(Def=5)

        100% Max    1521     99%    1521
         75% Q3      195     95%    1521
         50% Med     133     90%     400
         25% Q1       10     10%      61
          0% Min      30      5%      30
                              1%      30

        Range       1491
        Q3-Q1         85
        Mode          30


                 Extremes

          Lowest   Obs    Highest   Obs
             30(    8)     195(      4)
             61(   16)     229(      6)
             70(    9)     250(      2)
             98(   15)     400(     10)
            110(   17)    1521(      5)
```

```
                 Univariate Procedure

Variable=FARM

                    Moments

        N            17          Sum Wgts      17
        Mean         44.64706    Sum           759
        Std Dev      48.78132    Variance      2379.618
        Skewness     0.921172    Kurtosis      -0.22394
        USS          71961       CSS           38073.88
        CV           109.2599    Std Mean      11.83121
        T:Mean=0     3.773668    Pr>|T|        0.0017
        Num ^= 0     16          Num > 0       14
        M(Sign)      6           Pr>=|M|       0.0042
        Sgn Rank     65          Pr>=|S|       0.0002


                  Quantiles(Def=5)

         100% Max    150     99%      150
          75% Q3      69     95%      150
          50% Med     34     90%      125
          25% Q1       4     10%       -1
           0% Min     -1      5%       -1
                               1%      -1

         Range       151
         Q3-Q1        65
         Mode         -1


                    Extremes

          Lowest   Obs    Highest   Obs
            -1(    17)      69(      5)
            -1(    12)      75(     11)
             0(     8)     114(     14)
             2(    16)     125(      1)
             4(     7)     150(     10)
```

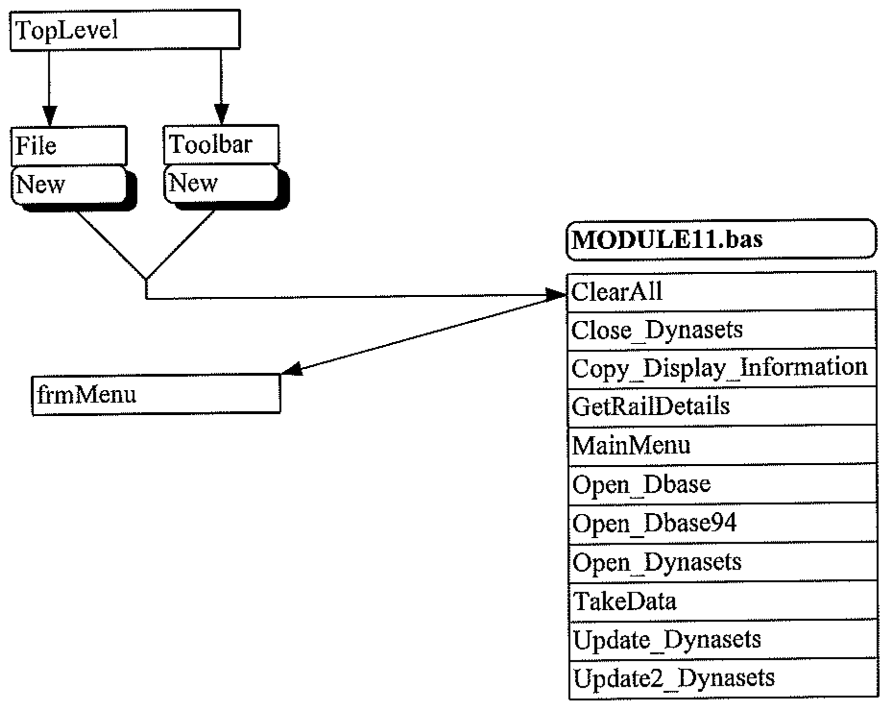
```
Univariate Procedure

Variable=TOTAL

                 Moments

N              72          Sum Wgts       72
Mean           59.52778    Sum          4286
Std Dev       185.6767     Variance    34475.83
Skewness        7.160637   Kurtosis       55.89389
USS       2702920  CSS      2447784
CV            311.916      Std Mean       21.88221
T:Mean=0       2.720374  Pr>|T|          0.0082
Num ^= 0      72    Num > 0       69
M(Sign)       33    Pr>=|M|          0.0001
Sgn Rank    1305    Pr>=|S|          0.0001


            Quantiles(Def=5)

 100% Max   1521      99%     1521
  75% Q3      47      95%      150
  50% Med     14.5    90%      120
  25% Q1       6      10%        2
   0% Min     -1       5%        1
                       1%       -1
 Range      1522
 Q3-Q1        41
 Mode          5


              Extremes

   Lowest   Obs    Highest   Obs
     -1(     57)     144(     16)
     -1(     56)     150(      5)
     -1(     34)     250(      6)
      1(     54)     400(     39)
      1(      3)    1521(     24)
```

```
            Univariate Procedure

Variable=FARM

              Moments

    N            72          Sum Wgts       72
    Mean         12.80556    Sum            922
    Std Dev      29.52661    Variance       871.8208
    Skewness     3.153631    Kurtosis       10.08595
    USS          73706       CSS            61899.28
    CV           230.5766    Std Mean       3.479745
    T:Mean=0     3.680027    Pr>|T|         0.0005
    Num ^= 0     43          Num > 0        42
    M(Sign)      20.5   Pr>=|M|        0.0001
    Sgn Rank     467         Pr>=|S|        0.0001


          Quantiles(Def=5)

    100% Max     150     99%     150
     75% Q3        8.5   95%      75
     50% Med       1     90%      34
     25% Q1        0     10%       0
      0% Min      -1      5%       0
                              1%      -1

    Range       151
    Q3-Q1       8.5
    Mode        0


              Extremes

     Lowest   Obs    Highest   Obs
        -1(   34)       69(    24)
         0(   69)       75(    40)
         0(   68)      114(    59)
         0(   64)      125(     5)
         0(   63)      150(    39)
```

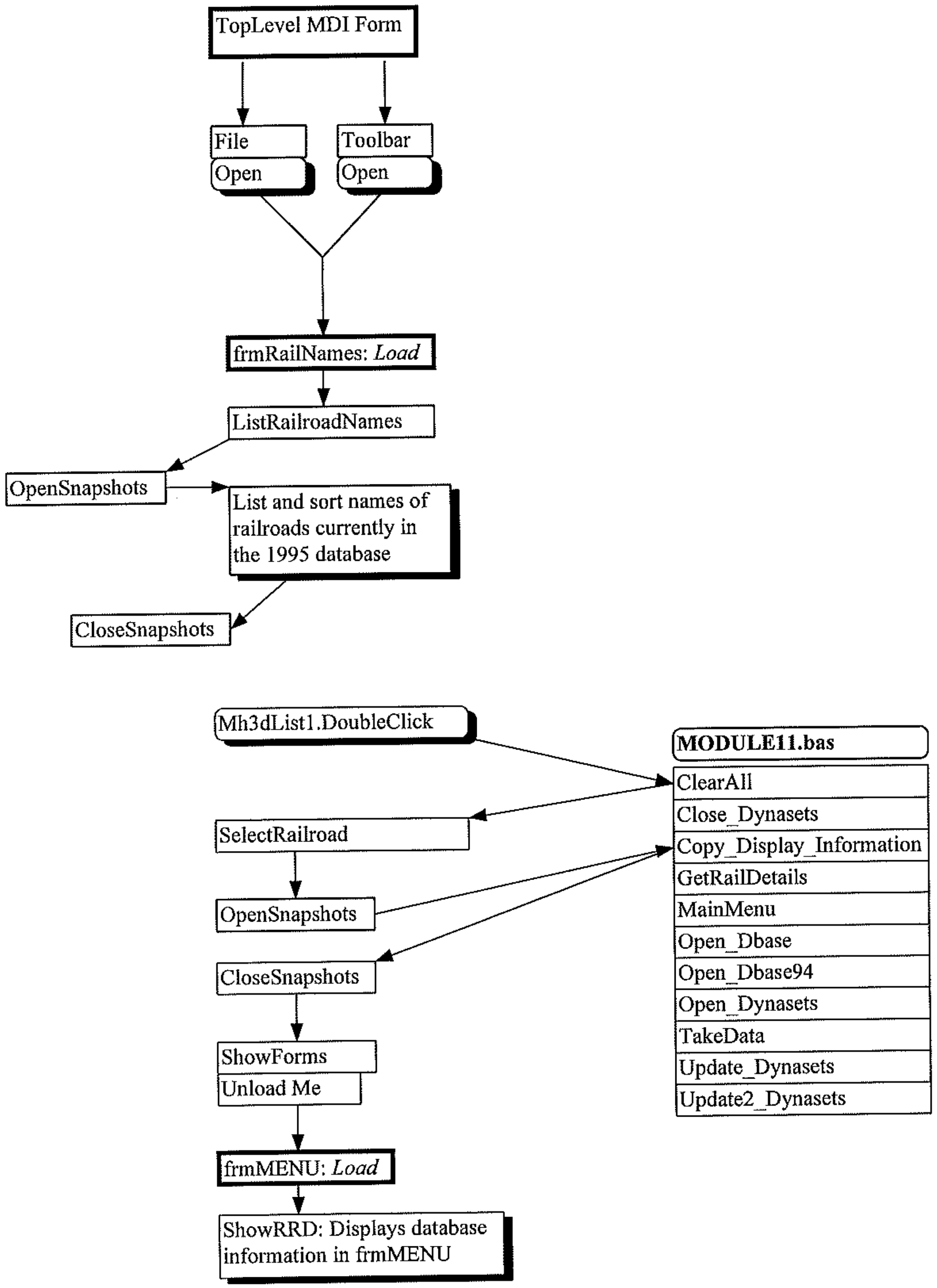
APPENDIX C: PROGRAM FLOWCHARTS

Opening A New Record In Dentry 2.0

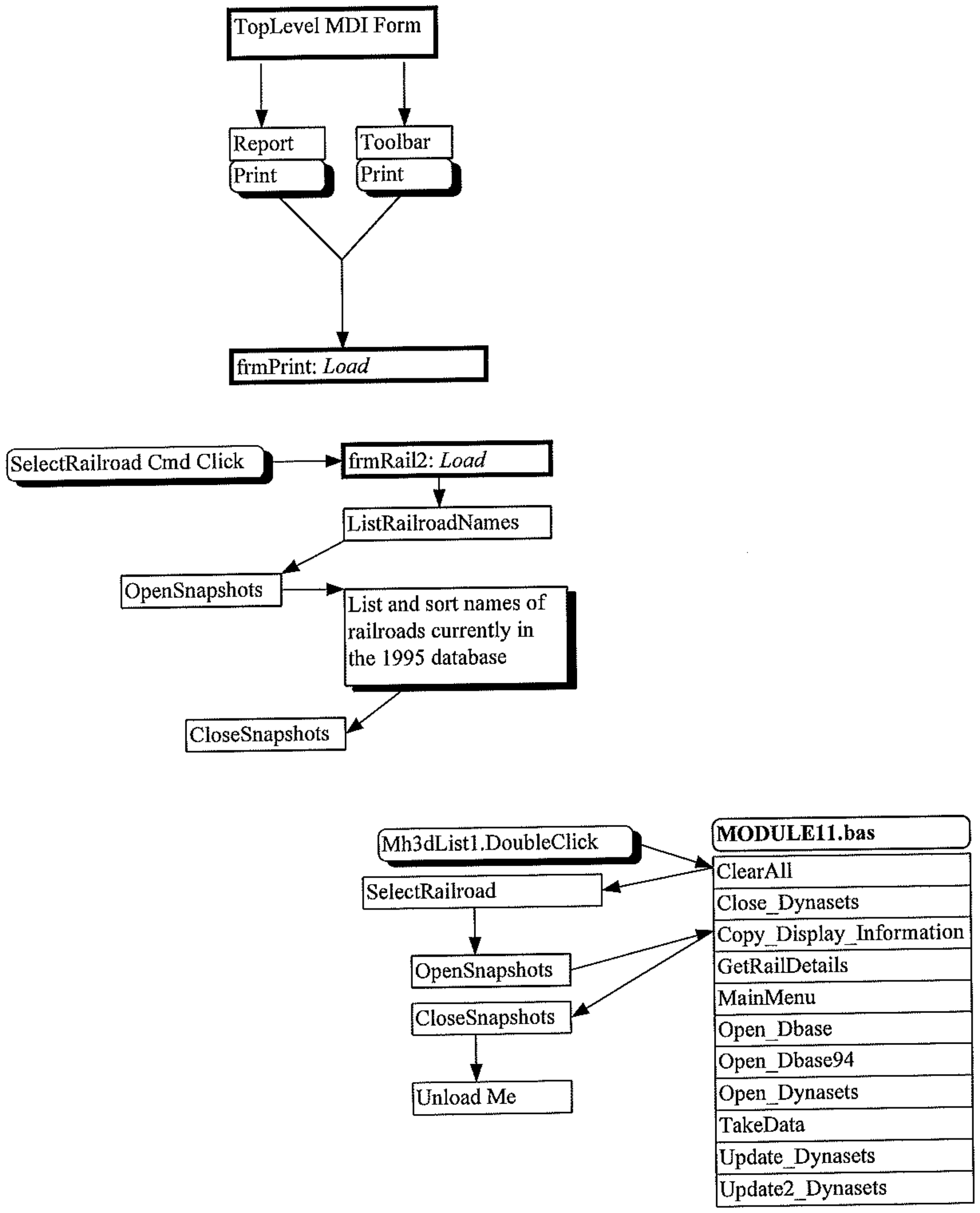
The sequence of events that occurs when the toolbar's new record button or the menu's file new command is invoked.



Opening An Existing Record In Dentry 2.0

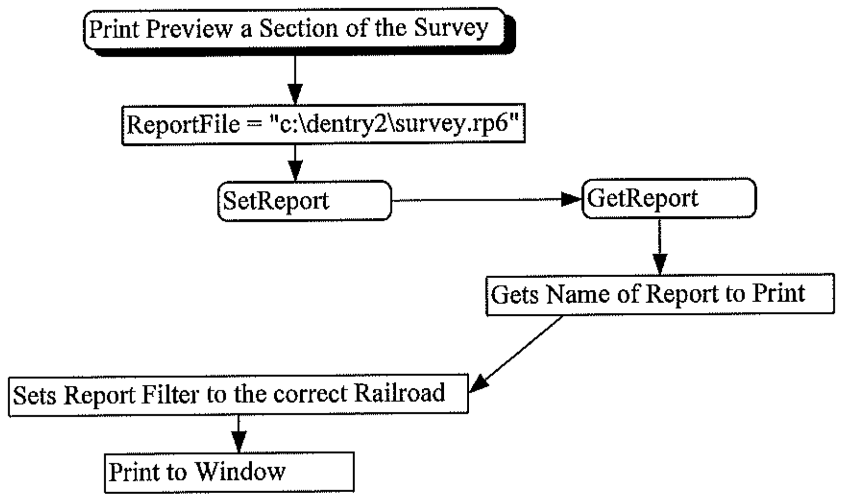


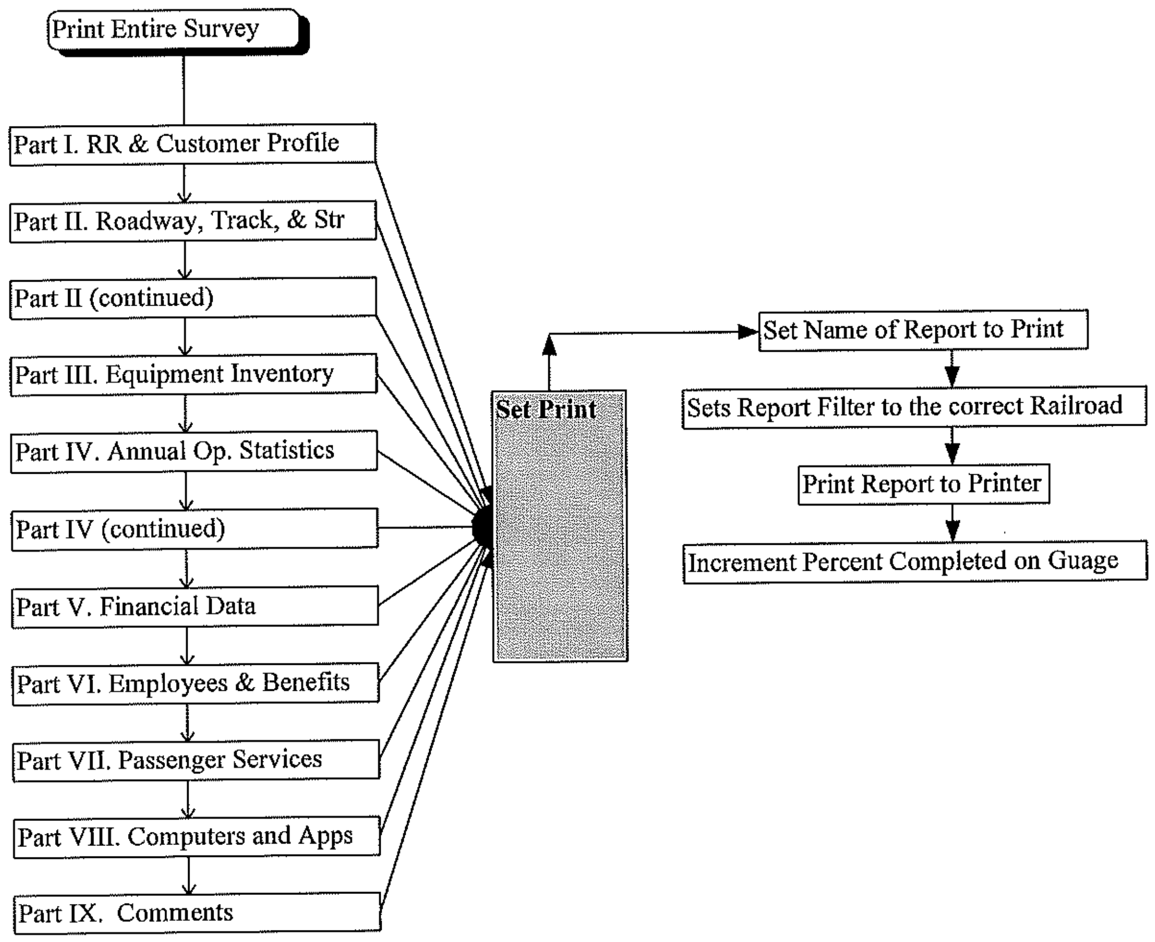
Selecting A Survey Record To Print In Dentry 2.0



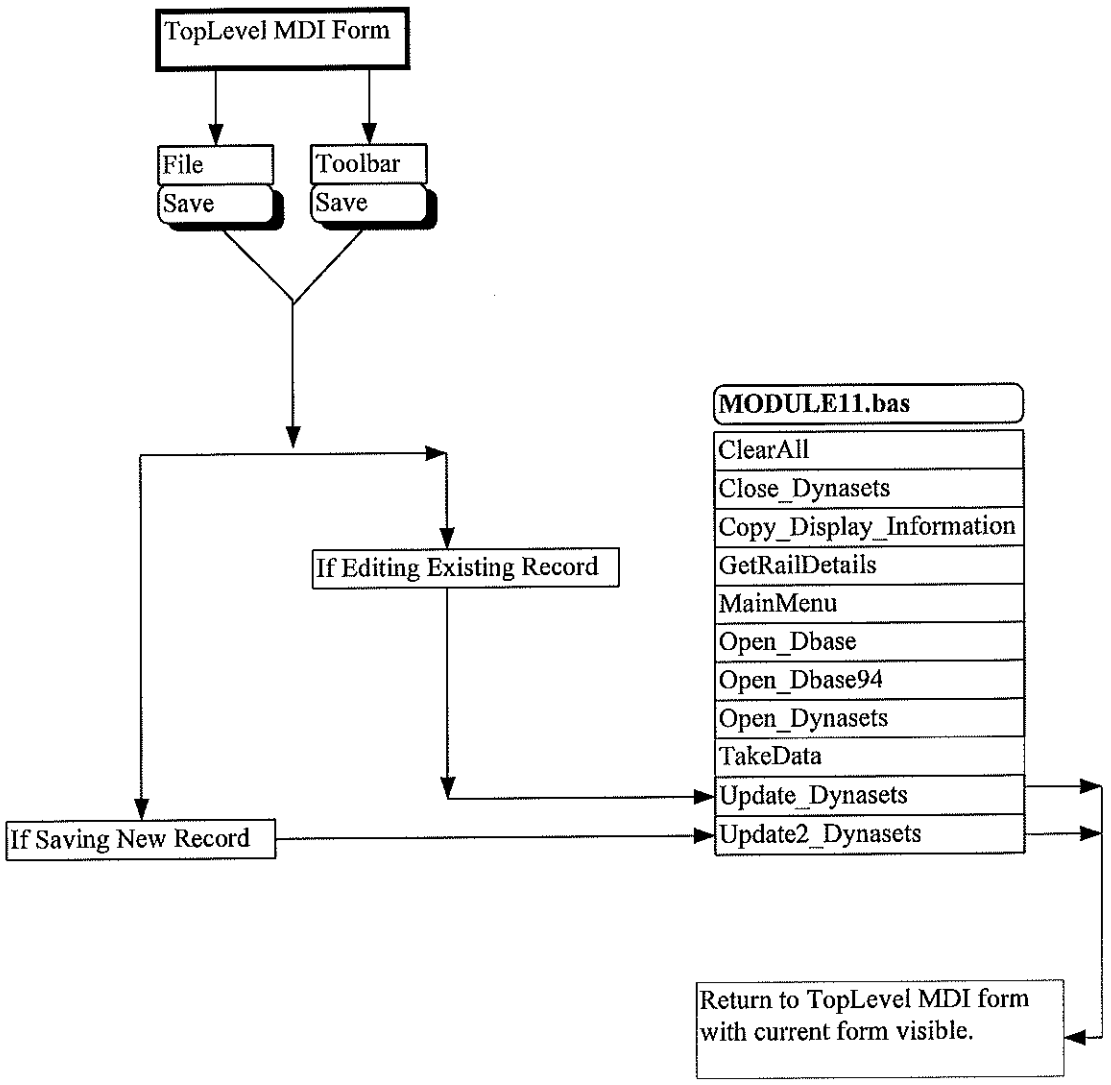
Print Preview & Print Entire Survey Commands

(continued from the previous page)


Print Preview a Section of the Survey
ReportFile = "c:\dentry2\survey.rp6"
SetReport
GetReport
Gets Name of Report to Print
Sets Report Filter to the correct Railroad
Print to Window


Print Entire Survey
Part I. RR & Customer Profile
Part II. Roadway, Track, & Str
Part II (continued)
Part III. Equipment Inventory
Part IV. Annual Op. Statistics
Part IV (continued)
Part V. Financial Data
Part VI. Employees & Benefits
Part VII. Passenger Services
Part VIII. Computers and Apps
Part IX. Comments
Set Print
Set Name of Report to Print
Sets Report Filter to the correct Railroad
Print Report to Printer
Increment Percent Completed on Guage

Save or SaveAs Command In Dentry 2.0



APPENDIX D: DATABASE TABLES AND THEIR FIELDS

Table: Annop2

FieldName	Type	Primary Key
ID	Counter	
RailId	Number	✓
HAZ_MAT_LOAD	Number	
INTERMOD_TRAILERS	Number	
INTERMOD_CONTAINERS	Number	

Table: Annual Operating Statistics

FieldName	Type	Primary Key
ID	Counter	✓
ANNUAL_OP_R_NO	Number	
STCC_CODE	Text	
COMMODITY	Text	
CAR_ORG_TERM_ON_LINE	Number	
INTERLND_CARS_ORG	Number	
INTERLND_CARS_TERM	Number	
BRIDGE_CARS	Number	

Table: Auxilliary Financial Information

FieldName	Type	Primary Key
ID	Counter	
AUX_R_NO	Number	✓
FREIGHT	Currency	
OTHER_OP_REV	Currency	
TOT_GROSS_OPER_REV	Currency	
WAY	Currency	
BRIDGES	Currency	
FACILITIES	Currency	
OTHER	Currency	
WAYS_AND_STRCT_TOT	Currency	
FREIGHT_CARS	Currency	
LOCOMOTIVES	Currency	
MOW_EXP	Currency	
OTHER_EQUIP_EXP	Currency	
TOT_EQUIP_EXP	Currency	
TRANSPORTATION	Currency	
GEN_AND_ADMIN	Currency	
OTH_OPER_EXP	Currency	
TOT_OPER_EXP	Currency	

FieldName	Type	Primary Key
NET_OPER_INC	Currency	

Table: Base Financial Information

Field Name	Type	Primary Key
ID	Counter	
FIN_R_NO	Number	✓
CURRENT_ASSETS	Currency	
CURRENT_LIABILITIES	Currency	
NET_WORK_CAPITAL	Currency	
CONTRIB_TO_CAPITAL	Currency	
TOTAL_ASSETS	Currency	
LONG_TERM_DEBT	Currency	
STOCKHOLDERS_EQUITY	Currency	
TOTAL_CAP_EXP	Currency	
DEP_AMORT_AND_RET	Currency	
OTHER_EXPENSE	Currency	
CAP_ROAD_EXP	Currency	
CAP_EQUIP_EXP	Currency	
CAP_OTHER_EXP	Currency	

Table: Benefit Plans Information

Field Name	Type	Primary Key
ID	Counter	
BENEFIT_R_NO	Number	✓
TOT_EXEMPT_EMP	Number	
TOT_NON_EXEMPT_EMP	Number	
Combined	Number	
TOT_ANNUAL_EXEMPT_COMP	Currency	
TOT_ANNUAL_NON_EXEMPT_COMP	Currency	
TOTAL_COMBINED	Currency	
NO_EMP_LABOR_AGREEMENT	Number	
TOT_MAN-HOURS_WORKED	Number	
BENEFIT_PLAN	Yes/No	
%_EMP_CONTRIB_SINGLE_MED	Number	
%_EMP_CONTRIB_FAMILY_MED	Number	
%_EMP_CONTRIB_SINGLE_DENTAL	Number	
%_EMP_CONTRIB_FAMILY_DENTAL	Number	
%_EMP_CONTRIB_SINGLE_LIFE	Number	
%_EMP_CONTRIB_FAMILY_LIFE	Number	
PENSION_PLAN	Yes/No	
401K_EMPLOYER_MATCHING	Yes/No	

Field Name	Type	Primary Key
DEF_BEN	Yes/No	
OTHER_PENSION_PLAN	Text	
TOTAL_ANNUAL_COST	Currency	

Table: Capital Investment Information

Field Name	Type	Primary Key
ID	Counter	
CAP_INV_R_NO	Number	✓
EQUIP_LOC_INVEST	Number	
EQUIP_LOC_FUND_INTERNALLY	Number	
EQUIP_ROLL_STOCK_INVEST	Number	
EQUIP_ROLL_STOCK_FUND_INTERNALLY	Number	
ROAD_TRACK_INVEST	Number	
ROAD_TRACK_FUND_INTERNALLY	Number	
ROAD_STRUCT_INVEST	Number	
ROAD_STRUCT_FUND_INTERNALLY	Number	
OTHER_INVEST	Number	
OTHER_FUND_INTERNALLY	Number	

Table: Comments

Field Name	Type	Primary Key
ID	Counter	✓
CommID	Number	
Comments	Memo	

Table: Computer Systems and Applications

Field Name	Type	Primary Key
ID	Counter	
COMP_R_NO	Yes/No	✓
286_COMPUTERS	Yes/No	
386_COMPUTERS	Yes/No	
486_COMPUTERS	Yes/No	
PENTIUM_COMPUTERS	Yes/No	
MACINTOSH	Yes/No	
MINICOMPUTERS	Yes/No	
MAINFRAME	Yes/No	
OTHER_COMPUTERS	Text	
DO_NOT_USE_COMPUTERS	Yes/No	
DOS_OPER_SYS	Yes/No	
WINDOWS	Yes/No	
WINDOWS_95	Yes/No	

Field Name	Type	Primary Key
OS/2	Yes/No	
UNIX	Yes/No	
APPLE	Yes/No	
OTHER_OPER_SYS	Text	
FAX/MODEM	Yes/No	
CD-ROM	Yes/No	
LAN	Yes/No	
ONLINE	Yes/No	
COMPUSERVE	Yes/No	
INTERNET	Yes/No	
PRODIGY	Yes/No	
OTHER_ONLINE_SERVICE	Text	
EDI	Yes/No	
INVENTORY_CONTROL	Yes/No	
ACCT/PAYROLL	Yes/No	
WAYBILLS_APP	Yes/No	
CAR_REPAIR_BILLING	Yes/No	
PERSONNEL_RECORDS	Yes/No	
FRA_SAFETY	Yes/No	
CAR_ORDERS	Yes/No	
CAR_HIRE	Yes/No	
YARD_OPERATIONS	Yes/No	
TRAFFIC_STATISTICS	Yes/No	
BUDGET_CONTROL	Yes/No	
OTHER_APPLICATIONS	Text	
BILLS_OF_LADING	Yes/No	
FREIGHT_BILLS	Yes/No	
WAYBILLS_EDI	Yes/No	
CAR_CONSIST	Yes/No	
ISS	Yes/No	
CONNECT_WITH_SHIPPERS	Yes/No	
REN	Yes/No	
OTHER_EDI	Text	
DO_NOT_USE_EDI	Yes/No	
DEVELOP_OWN_COMPUTER_APP	Yes/No	
EXPANSION	Text	

Table: Customer Information

Field Name	Type	Primary Key
ID	Counter	
RAILROADIDNUMBER	Number	✓
TOTAL_CUSTOMERS_SERVED	Number	
COAL	Number	

Field Name	Type	Primary Key
Farm_Products	Number	
CHEMICAL_AND_ALLIED_PRODUCTS	Number	
FOOD_AND_KINDRED_PRODUCTS	Number	
NON-METALLIC_MINERALS	Number	
TRANSPORTATION_EQUIPMENT	Number	
LUMBER_AND_WOOD_PRODUCTS	Number	
PULP_PAPER_AND_ALLIED_PRODUCTS	Number	
PETROLEUM/COKE_PRODUCTS	Number	
STONE_CLAY_AND_GLASS_PRODUCTS	Number	
METALLIC_ORES	Number	
PRIMARY_METAL_PRODUCTS	Number	
WASTE_AND_SCRAP_MATERIAL	Number	
OTHERS	Number	
OthComm1	Text	
OthComm2	Text	
OthComm3	Text	

Table: Equipment Information

Field Name	Type	Primary Key
ID	Counter	
EQUIP_R_NO	Number	✓
CAR_LOCO_NAME	Number	
TOT_OWN_BXCR	Number	
TOT_LEAS_BXCR	Number	
LSS_10_BXCR	Number	
10_TO_20_BXCR	Number	
GTR_20_YRS_BXCR	Number	
TOT_OWN_GON	Number	
TOT_LEAS_GON	Number	
LSS_10_GON	Number	
10_TO_20_GON	Number	
GTR_20_YRS_GON	Number	
TOT_OWN_COVH	Number	
TOT_LEAS_COVH	Number	
LSS_10_COVH	Number	
10_TO_20_COVH	Number	
GTR_20_YRS_COVH	Number	
TOT_OWN_OPNH	Number	
TOT_LEAS_OPNH	Number	
LSS_10_OPNH	Number	
10_TO_20_OPNH	Number	
GTR_20_YRS_OPNH	Number	
TOT_OWN_FLAT	Number	

Field Name	Type	Primary Key
TOT_LEAS_FLAT	Number	
LSS_10_FLAT	Number	
10_TO_20_FLAT	Number	
GTR_20_YRS_FLAT	Number	
TOT_OWN_OTHER	Number	
TOT_LEAS_OTHER	Number	
LSS_10_OTHER	Number	
10_TO_20_OTHER	Number	
GTR_20_YRS_OTHER	Number	

Table: Equipment table 2

Field Name	Type	Primary Key
ID	Counter	
EQ2_R_NO	Number	✓
TOT_OWN_LT15	Number	
TOT_LEAS_LT15	Number	
LSS_10_LT15	Number	
10_TO_20_LT15	Number	
GTR_20_YRS_LT15	Number	
TOT_OWN_15TO30	Number	
TOT_LEAS_15TO30	Number	
LSS_10_15TO30	Number	
10_TO_20_15TO30	Number	
GTR_20_YRS_15TO30	Number	
TOT_OWN_GT30	Number	
TOT_LEAS_GT30	Number	
LSS_10_GT30	Number	
10_TO_20_GT30	Number	
GTR_20_YRS_GT30	Number	

Table: FRA-Track Class Information

Field Name	Type	Primary Key
ID	Counter	
FRA_TRACK_R_NO	Number	✓
MILES_CLASS1_ROAD	Number	
MILES_CLASS2_ROAD	Number	
MILES_CLASS3_ROAD	Number	
MILES_CLASS4_ROAD	Number	
EXCEPTED_MILES	Number	
CONCRETE	Number	
STEEL	Number	
WOOD	Number	

Field Name	Type	Primary Key
COMBINATION	Number	
USEDTIES	Number	
NEWTIES	Number	
90POUNDSGREATER	Number	
90POUNDSLESSER	Number	
GREATERADDITIONAL	Number	
LESSERADDITIONAL	Number	
PUBLIC	Number	
PRIVATE	Number	
EQUIP_AUTO_WARN	Number	
AUTO_WARN_INSTALLED	Number	
IMPROVE_AUTO_WARN_DEVICES	Number	
NO_RESURF_CROSS	Number	

Table: FRA 2

Field Name	Type	Primary Key
ID	Counter	
RailId	Number	✓
NO_INTERM_FAC	Number	
NO_CIRCUS_RAMPS	Number	
NO_TOP_PICKUP	Number	
NO_BOTTOM_PICKUP	Number	
NO_TRANS_FAC	Number	
NO_RAIL_TO_TRUCK	Number	
NO_TRUCK_TO_RAIL	Number	
NO_RAIL_TO_WATER	Number	
NO_WATER_TO_RAIL	Number	

Table: GENPROF

Field Name	Type	Primary Key
ID	Counter	
R_NUMBER	Number	✓
TRAIN_MILES	Number	
YARD_MILES	Number	
LOCOMOTIVE_MILES	Number	
AVG_LENGTH_OF_HAUL	Number	
AVG_REV_PER_CARLOAD	Currency	
AVG_WGT_PER_CARLOAD	Number	
REVENUE_TON_MILES	Number	
CABOOSE	Yes/No	
One_Way_EOT	Yes/No	
Two_Way_EOT	Yes/No	

Field Name	Type	Primary Key
OTHER	Text	
TOTAL_GALS_FUEL	Number	
AVG_COST_PER_GAL	Currency	

Table: Passenger Services Information

Field Name	Type	Primary Key
ID	Counter	
PASS_R_NO	Number	✓
PASS_SERVICES	Yes/No	
SEASONAL_EX	Yes/No	
DINNER_TRAIN	Yes/No	
OTHER	Text	
ANN_REV_FROM_SERV	Currency	
TOT_REV_PASS	Number	

Table: Railroad Names

Field Name	Type	Primary Key
ID	Counter	✓
RIGHTS	Yes/No	
Rail_Name	Text	
RailType	Text	
R_Owner	Text	
ContactPerson	Text	
R_Year	Number	
ShortLineYear	Number	
Region	Text	

Table: Roadway Information

Field Name	Type	Primary Key
ID	Counter	✓
ROADWAY_R_NO	Number	
STATE_ID	Text	
RT_MILES_OWNED	Number	
PER_OWN_GTR_90_LBS	Number	
TOT_RT_MILES_OPERATED	Number	

APPENDIX E: FILES COMPILED IN THE DENTRY 2.0 EXECUTABLE

FileName	Function of the file
frmabout.frm	Dentry 2.0 program information form
frmaos.frm	Annual operating statistics form
frmaos1.frm	Annual operating statistics form 2
frmben1.frm	Employee benefits form
frmcomm.frm	Comments form
frmcomp.frm	Computer systems form
frmcomp2.frm	Computer applications form
frmcs.frm	Customer information form
frmdes2.frm	Railroad information & main menu form
frmebd.frm	Employee information form
frmequip.frm	Equipment form
frmfd.frm	Income statement financial data form
frmfd2.frm	Balance sheet financial data form
frmfd3.frm	Future investment form
frmfg.frm	Dentry 2.0 startup form
frminter.frm	Intermodal and transloading facilities form
frmirts.frm	Road and track form
frmnmes2.frm	Form listing 1995 records available to print
frmops4.frm	General profile information
frmps.frm	Passenger services form
frmstruc.frm	Structures information form
print.frm	Form for printing the 1995 survey records
rnames.frm	Form listing 1995 records available to edit
rnames94.frm	Form listing 1994 records available to view
saveas.frm	Form for saving 1995 database file to floppy
toplevel.frm	MDI form for Dentry 2.0
constant.txt	Visual Basic global constants
datacons.txt	Global data access constants
module1.bas	Windows global function declarations
module11.bas	Global program declarations and procedures
rsdecl.bas	VB function declarations for rsreport.dll
rswvb.bas	Rsreport action constants

APPENDIX F: LIST OF DISTRIBUTION FILES FOR DENTRY 2.0

Group FileName	Function of the file
Program Directory Files	
dentry2.exe	Dentry 2.0 executable file
bltrain.bmp	Bitmap of train
rrctl.dll	Dialog box control file
rswpd.dll	Runtime printer DLL
rswreng.dll	Runtime English resource file
rswrun.exe	Runtime program
rswsql.ini	Translation parameters for R&R functions
slrd95.mdb	1995 Microsoft Access database
slrd95.ldb	1995 database record locks
survey.ico	Dentry program icon
Windows Files	
odbcddp.ini	Datasource setup initialization file
Windows/System Files	
cmdialog.vbx	Common dialog control
compobj.dll	OLE 2.0 library
ctl3d.dll	The 3D control DLL
ctl3dv2.dll	Newer version of 3D control DLL
guage.vbx	Visual Basic guage control
keystat.vbx	Key status Visual Basic control
mhglbx.vbx	3D label Visual Basic control
mhgtxt.vbx	3D text box Visual Basic control
mhrun500.dll	VBTools 5.0 control DLL
msajt112.dll	Jet 2.x comp. loader
msajt200.dll	Jet 2.5 engine
mscpxlt.dll	ODBC translators
msjeterr.dll	Error services
msjetint.dll	International
odbc.dll	ODBC driver manager
odbcadm.exe	ODBC administrator program
odbcinst.dll	Driver setup manager
odbcinst.hlp	ODBC administrator help file
odbcjet.hlp	16-bit database driver help file
odbcjt16.dll	16-bit database driver
odbctl16.dll	16-bit database driver
ole2.dll	OLE 2.0 library
ole2.reg	OLE Windows registration file

Group FileName	Function of the file
ole2disp.dll	OLE 2.0 automation library
ole2nls.dll	OLE 2.0 international character set translator
querr.lic	Q+E library and utility files
rrbas06.dll	Q+E library and utility files
rrgui03.dll	Q+E library and utility files
rrlib.dll	Q+E library and utility files
rrmds03.dll	Q+E library and utility files
rrqry03.dll	Q+E library and utility files
rrsql03.dll	Q+E library and utility files
rrutl03.dll	Q+E library and utility files
rrutl06.dll	Q+E library and utility files
rsreport.dll	Runtime DLL file
rsw.vbx	R&R custom control
scp.dll	Code page translation library
stdole.tlb	OLE type library
storage.dll	OLE 2.0 library
threed.vbx	3D command button control file
typelib.dll	OLE automation type information interfaces
vaen2.dll	16-bit jet file
vaen2.olb	16-bit jet file
vbajet.dll	Microsoft Access driver file
vbalink.dll	Microsoft Access driver file
vbar2.dll	16-bit jet file
vbdb300.dll	Visual Basic / Jet support
vbrun300.dll	Visual Basic runtime file