University of Tennessee, Knoxville

TRACE: Tennessee Research and Creative Exchange

Masters Theses

Graduate School

8-1998

Using geographic information systems for business logistics analysis

Kenneth M. Bennett

Follow this and additional works at: https://trace.tennessee.edu/utk_gradthes

Recommended Citation

Bennett, Kenneth M., "Using geographic information systems for business logistics analysis. " Master's Thesis, University of Tennessee, 1998.
https://trace.tennessee.edu/utk_gradthes/10171

This Thesis is brought to you for free and open access by the Graduate School at TRACE: Tennessee Research and Creative Exchange. It has been accepted for inclusion in Masters Theses by an authorized administrator of TRACE: Tennessee Research and Creative Exchange. For more information, please contact trace@utk.edu.

To the Graduate Council:

I am submitting herewith a thesis written by Kenneth M. Bennett entitled "Using geographic information systems for business logistics analysis." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Geography.

Bruce Ralston, Major Professor

We have read this thesis and recommend its acceptance:

Tom Bell, Chen Liu

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Kenneth M. Bennett entitled "Using Geographic Information Systems for Business Logistics Analysis." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Geography.

[illegible]
Bruce Ralston, Major Professor

We have read this thesis
and recommend its acceptance:

Thomas L Bell
Tom Bell

Cheng Liu
Chen Liu

Accepted for the Council:

[illegible]
Associate Vice Chancellor and
Dean of The Graduate School

USING GEOGRAPHIC INFORMATION SYSTEMS FOR BUSINESS LOGISTICS ANALYSIS

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Kenneth M. Bennett
August 1998

Acknowledgments

During the nearly two years I have spent at the University of Tennessee, and throughout my life, there have been many colleagues, friends, and family members whose influence on me has led, in one way or another, to my arrival at this moment. While I cannot acknowledge all of them here, there are several whose names must not go unmentioned.

First and foremost, I would like to express my deepest gratitude to Dr. Bruce Ralston for being my teacher, mentor, and friend. For the entirety of my stay here at the University of Tennessee, he has responded to my concerns and ambitions – both as a student, and as a first-time father – with the utmost respect, understanding, and generosity. I would like to thank the other members of my Thesis Committee – Drs. Bell and Liu – for their cooperation and guidance, and Bill Dewitt, from the College of Business, for his inspiration and support. I must also offer my appreciation to the faculty and my fellow graduate students in the Department of Geography for their friendship and camaraderie, and for giving me a new perspective on the world.

Perhaps the greatest debt I owe is to my family. My parents, Errol and Sharon Bennett, long ago planted the seeds of intellectual and professional pursuit that have born fruit with this thesis. My grandfather Al Hughes has always inspired me to think scientifically. My brother Eric has comforted me with his solid personality and common sense. My brother Daniel has encouraged me with his challenges, good humor, and alacrity. Most importantly, my wife Rosario and our son Diego have given me the love and support I have needed to achieve this goal.

Abstract

Although geographic information system (GIS) technology has been used by government agencies and academic institutions since the early 1960s, the adoption of GIS technology by the private sector has only begun to occur on any significant scale in the past few years. Obstacles to its diffusion have been the high cost of necessary computer hardware, lack of readily available spatial data, and misconceptions about what GIS is and who can benefit from it. Since the late 1980s, however, certain trends have cleared the way for growth in private sector applications of GIS. One of these trends has been the dramatic decline in the cost of computer hardware coupled with greatly improved performance. Another trend has been the phenomenal growth in spatial data available from government agencies and professional spatial data providers. Lastly, corporations have begun to find themselves overloaded with data, and are seeking innovative ways to leverage their data resources, much of it geographically referenced, in order to gain an information-based competitive advantage. While most private sector GIS applications have focused on sales territory management, niche marketing, retail location analysis, and fleet management, its potential as a tool for logistics analysis has gone relatively unnoticed. This thesis explains the advantages of using GIS for logistics, and discusses in detail its application to the distribution network optimization of a major U.S. drugstore chain. Emphasis is placed on the ability of GIS to provide a better understanding of business logistics processes through the power of visualization. It is concluded that the inherent advantages of GIS technology for data processing, combined with its devolution into libraries of functions and objects that may incorporated piecemeal into mainstream information systems, will fuel the rapid diffusion of GIS throughout the private sector.

Preface

When we reason about quantitative evidence, certain methods for displaying and analyzing data are better than others. Superior methods are more likely to produce truthful, credible, and precise findings. The difference between an excellent analysis and a faulty one can sometimes have momentous consequences.

– Edward R. Tufte

Table of Contents

List of Tables

List of Figures

Figure Page

Figure Page

Chapter 1

An Introduction to GIS and Logistics

Introduction

An examination of the application of geographic information system (GIS) technology to the field of business logistics is difficult to conduct for two reasons. The first reason is that very few profiled cases exist where a GIS has been implemented for the expressed purposes of business logistics management. The second reason is that GIS technology, and especially its application to mainstream business activities, is a relatively recent phenomenon. Therefore, it will take some time for such GIS applications to be addressed in depth in the literature (Dewitt and Ralston, 1996). Nevertheless, the promise GIS technology holds for business logistics remains strong, and it is my conviction that GIS technology will soon become a commonplace tool within that industry. The goal of this thesis is to explain why GIS has not been, but is now very much ready to be, embraced by the business logistics industry and to illustrate a practical example of how GIS technology can be used to develop a powerful, yet simple-to-use tool for logistics analysis. To understand the current status of GIS in the logistics industry, it is first necessary to explore the historical and theoretical framework underlying the diffusion of GIS technology over the past several decades.

The Diffusion of GIS Technology

Coppock and Rhind have shown that the pioneering of GIS technology began in North America in the late 1950s and the early 1960s with attempts to use emerging computer technology to automate cartography and the processing of geographically referenced data, such as those derived from census, cadastral, and land use surveys. Innovations in automated mapping were made by national agencies such as the U.S. National Ocean Survey, by military establishments, and by universities, notably the Harvard Laboratory for Computer Graphics and Spatial Analysis. The early developers of geographically referenced data processing were often large national government agencies, such as the U.S. Bureau of the Census and the Canadian Department of Agriculture. It was through this latter organization that Roger Tomlinson developed what is considered to be the first true GIS, the Canadian Geographic Information System (Coppock and Rhind, 1991). What is noteworthy here is that the origin of GIS technology was in the public sector and academia.

For reasons which will be discussed, the employment of GIS technology continues to be dominated by the public sector and academia, and has experienced a rather slow adoption rate by private sector business. A 1993 survey revealed that roughly 65% of GIS users are educational institutions and government agencies at federal, state, and local levels, while approximately 14% are from other land and resource intensive industries such civil, environmental, and transportation engineering and consulting, utilities, forestry, and real estate. Only 6% of the users were businesses in the commercial sense, and they were categorized as retail marketing and sales. A 1995 survey of GIS software sales showed only 9% going to business. Of these, a vast majority are retail marketing and sales applications, while business logistics is not even mentioned (Korte, 1997). Grimshaw has also noted that while GIS is already widespread throughout the public sector, private sector business has only just started to realize its practical applications (Grimshaw, 1994).

One of the factors inhibiting the diffusion of GIS technology into the private sector has been its cost. Early GIS applications, like other early software applications, were designed to run on the only computer systems then available – mainframes. The expense of acquiring mainframe computer hardware and software, and of training staff to use them, was prohibitively high for all but the largest of organizations (Korte, 1997). Goodchild has argued, moreover, that GIS applications have been considered less central to business processes than accounting, spreadsheet, and word processing applications. And so, any organization considering the use of GIS will subject the technology to a strict cost-benefit analysis (Goodchild, 1991). Before the dramatic decline in the cost of computing technology in the last decade or so (Ibid.), it would have been difficult to convince many organizations of the value in investing in the technology. Thus, it is no surprise to find that public agencies and academic institutions formed the market stronghold for early GIS applications. Such organizations traditionally have had less stringent return-on-investment policies than private sector, market-driven, organizations. Also, the research role of academic institutions leads many of them to accept "cutting edge" technologies long before private sector accounting can justify them.

A second factor limiting the business world's acceptance of GIS relates to the accessibility of the technology and the data upon which it depends. As already noted, early GIS applications were mainframe-based. Like other mainframe-based applications, GIS was a centralized, "backroom" activity that was highly specialized, capital intensive, and often slow and laborious (Dewitt et al., 1997; Szajgin, 1997). Indeed, early GIS technology was relatively more cumbersome than currently, since the storage, manipulation, and integration of spatial and non-spatial data is, in general, more computationally intensive than most other data processing applications (Goodchild, 1991). Early GIS technology also suffered from a lack of readily available spatial data (e.g. coordinatized base maps of political boundaries, road and rail networks), so organizations using GIS often had to produce their own maps "in house" before they could take advantage of the technology (Hamilton, 1996). These technology and data constraints meant that only organizations whose time horizons for decision-making were relatively long, such as government planning agencies and certain land-intensive industries such as forestry, were willing to commit the human and capital resources necessary to exploit the unique capability of GIS to process information spatially. Time horizons for decision-making in the traditional business environment, on the other hand, are much shorter (Dewitt et al., 1997). The turnaround time for information requested of early GIS applications often would have exceeded business expectations, thus discouraging the adoption of the technology.

A final factor inhibiting the diffusion of GIS technology in the business arena involves various elements of social and organizational theory. Rogers has argued that the complexity of a technology innovation will slow its rate of adoption among potential users, and suggests that GIS diffusion suffers from such complexity, due to its lack of user-friendliness and its rapidly expanding and advancing functionality. He also cites studies showing that the decision to adopt a new technology often hinges primarily on the observation and recommendations of industry peers who use the new technology, rather that on media and corporate publicity about the new technology. A technology like GIS, whose user base is dominated by the public sector, will therefore have difficulty, at least initially, in bridging the personal communication gap that exists between public and private sector employees (Rogers, 1993). Sherwood blames the U.S. business world's resistance to GIS technology on the lack of geographical awareness of the baby boomer generation (i.e., 1946 - 1964) and their immediate forebears. These people, who are the ones primarily in control of today's businesses, had little exposure to geography in their primary, secondary, and even post-secondary levels of schooling, because geography had fallen into decline within educational curricula during these times, and subsequently lost its reputation as an applied professional discipline in the non-academic world. With this in mind, she argues that current GIS applications are associated too much with the unrecognized discipline of geography and are too generic in their functionality. She suggests that business users will more readily accept GIS when the functional capabilities and the vocabulary of GIS applications are tailored to specific business tasks (Sherwood, 1995). Grimshaw has also cited the generic and complex nature of GIS applications as a constraint to its adoption, as well as its stigma as a tool limited to geographers. He also argues that GIS technology, like other information technology innovations, suffers from corporate information strategies that view new technologies as merely a substitute for existing data processing methods, rather than as a complement to them. Thus, organizations that do not already use maps extensively will not see the need for GIS technology, and will fail to appreciate the ability of GIS to leverage the wealth of geographically referenced data existing in their current systems (Grimshaw, 1994).

In spite of these obstacles to the diffusion of GIS technology, the 1980s and early 1990s saw dramatic changes in the computer industry in general, and in GIS technology specifically. These changes are aiding the adoption of GIS technology. Perhaps the most significant was the dramatic order-of-magnitude declines in the cost of computing (Coppock and Rhind, 1991). In tandem with cheaper computer technology came the shift away from centralized, mainframe-based computing toward distributed computing based on networks of powerful workstations (Goodchild, 1991). This time period also saw the rapid dissemination of the desktop computer and an exponential growth in their computational and graphical capabilities. Although desktop computers were originally scorned as a novelty of the home consumer, information systems of the largest public and private sector organizations are today being built around the relatively inexpensive desktop computer, using client-server networks, relational database management systems, and user-friendly, windows-and-mouse-based graphical user interfaces. At the same time, vast amounts of spatial data have been produced, standardized, and enhanced with increasing levels of accuracy, and have been made available to the public at decreasing cost by a host of commercial data vendors and government agencies (Fung and Remsen, 1997; Johnson, 1993; Zwart, 1993). The familiar adage that 80% of all data may be geographically referenced has also made its way into corporate information strategy. Corporations once struggled with having too little data about their operations, but as information systems have improved, many now find themselves inundated by a sea of data they cannot easily interpret. Increasingly, corporations are looking to the unique ability of GIS technology to process and display complex sets of data spatially in an effort to more quickly and efficiently generate useful information (Dewitt, 1997; Rao, 1995). Finally, geography as a discipline has been experiencing a renaissance since the early 1990s in the national education policy of the U.S., as well as at the grassroots level, which should help to make future business managers in the U.S. more disposed to spatial analytical techniques and GIS technology (Sherwood, 1995).

The result of GIS developers adapting their products to the new desktop regime, and the increasing availability of vast amounts of commercial and corporate data, has been an explosion in the use of GIS technology (Huxhold and Levinsohn, 1995; Korte, 1997). As noted earlier, a small but increasing portion of this growth has been due to private sector businesses investing in the technology, and it is projected that the business market will continue to fuel GIS sales, especially as the business world becomes more aware of the opportunities to leverage their spatial data resources to remain competitive (Tetzeli, 1993; Jacobs, 1996; Swenson, 1996). Unfortunately, only certain functional areas of business are currently pursuing GIS technology with any vigor, while other areas, such as business logistics, have not been as accepting. Grimshaw's very thorough discussion of applying GIS technology to private sector business targets marketing, sales, and retail location as the most promising business application of GIS (Grimshaw, 1994).

My review of existing literature about companies that have applied GIS to business seems, in general, to corroborate Grimshaw's prediction. Moreover, the only periodical devoted primarily to business applications of GIS – GIS World, Inc.'s *Business Geographics* – also devotes most of its coverage to retail marketing and sales applications.

Nevertheless, GIS technology is beginning to make some significant inroads into one area of logistics, namely fleet management. Sears has used GIS as a tool for scheduling and routing home deliveries (Jacobs, 1996), and Federal Express uses GIS in its Operations Research and Spatial Applications Department to manage and optimize its vast transportation operations (Gates, 1997). Yellow Freight Systems is another transportation company using GIS to manage its fleet of 3,700 trucks (Tetzeli, 1993). The U.S. Postal Service has combined GIS and global positioning system (GPS) technology as part of an automatic vehicle tracking system (Harder, 1997). Fleet management, however, is only a small part of what constitutes business logistics.

A Brief Overview of Business Logistics

Business logistics is defined by the Council of Logistics Management (CLM), a professional organization of logistics managers and educators, as

> the process of planning, implementing, and controlling the efficient, cost-effective flow and storage of raw materials, in-process inventory, finished goods and related information from point of origin to point of consumption for the purpose of conforming to customer requirements.

Implied in this definition are a number of major logistics planning areas. Specifically, these are 1) customer service level decisions, 2) location decisions, 3) inventory decisions, and 4) transportation decisions. The first of these areas encompasses the latter three, because the target level of customer service that is established will affect the other three decision-making areas. Therefore, deciding on the level of service a customer should receive is an important task that will impact the overall design of the logistics system. Location decisions involve determining the number, size, and geographic placement of manufacturing, warehouse, and retail facilities, and allocating the market areas these facilities will serve. Inventory decisions consider different strategies for managing inventory flow, the levels of inventory (both the cycle stock for anticipated sales, and the safety stock for unanticipated sales) to maintain, and the deployment of raw materials and products throughout the logistics network. Transportation decisions include mode selection, shipment size, and routing and scheduling. All of these major planning areas, moreover, are interrelated, and often pose trade-off situations in which a cost decrease in one area results in a cost increase in another (Ballou, 1992). One example is the trade-off between inventory and transportation. While shipping by rail may be cheaper than shipping by truck, the increase in transit time and the decrease in reliability associated with rail may necessitate larger inventories at warehouse or retail locations, such that the inventory carrying cost may exceed the savings in transportation cost.

Logistics, GIS, and Visualization

Logistics involves the movement of materials and goods across space in a timely manner. Since logistics is concerned fundamentally with place as well as time, any technology that helps to depict spatial relationships will be very useful to logistics (Barone, 1997). As it turns out, logistics problems lend themselves well to GIS-based analysis. The geographic distribution of facilities in a logistics network, the lines of transportation that connect them, and the service areas resulting from the two, are easily represented by the geometry of points, lines, and polygons a GIS employs for thematic mapping. However, these maps are more than just a graphical representation of a logistics network, for associated with each graphical feature are tabular, alphanumeric data describing the real world objects or processes those features are designed to represent. Thus, a point on a GIS map representing a warehouse can contain information about the inventory contents and levels, the rate of materials handling cost, delivery windows, the fixed costs, or the capacity of the facility. Likewise, a line representing a shipment route can contain information about its mode, its capacity, and its shipping rate and time. Moreover, this tabular information can be used to graphically enhance the map display, by sizing the points representing warehouses according to capacity, or by color-coding the shipment lines according to their mode, for example. Since there is practically no limit to the amount of data an associated table may hold, the many different ways a feature can be displayed is limited only by what is known about the object or event it represents.

This ability of GIS to use maps to display information has become what has been coined the new "map paradigm" of the information age. The premise of this bold statement rests on the special angle GIS offers for displaying data.

> One of the great insights of GIS is that there is a vast difference between seeing data in a table of rows and columns...and seeing it presented in the form of a map. The difference is not simply aesthetic, it's conceptual – it turns out that the way you see your data has a profound effect on the connections you make and the conclusions you draw from (sic) them (Harder, 1997).

The traditional tools of business, such as the spreadsheet or database, can often obscure or misrepresent data that is linked to location. GIS, on the other hand, can link such data to places and processes, thus making the data more easily and intuitively understood (Ibid). In a similar vein, Buttenfield and Mackaness have argued that GIS is a technology that is well suited to meet the rising demand for what has been called data "visualization." Visualization is a method of data exploration that has grown from the needs of our information age

> to access pertinent information from an overwhelming volume of collected data; to communicate complex patterns effectively; to formalize sound principles for presentation of data that optimize visual processing skills; and to steer analytical computations for data modeling and interpretation (Buttenfield and Mackaness, 1991).

The key to successful data visualization lies in the graphical user interface (GUI). As its name implies, the GUI is the working environment graphically displayed on the computer screen that enables the user to interact with the data. This interaction involves more than just the viewing of data – it also involves data manipulation. Therefore, the degree to which a GUI helps or hinders the interaction with data depends less on how artistic the display is, and more on how logical and intuitive it is. If the GUI is not "user-friendly", as it is said, the user becomes distracted by the interface and loses sight of the data. In other words, the tools for interacting with data that the GUI provides should be transparent to the user, thus allowing the user to concentrate on the data themselves (Ibid.). The ability of GIS technology to embed data in map features, and to render the appearance of those features according to the data they contain, creates just such a logical and intuitive GUI. The GIS GUI enables the user to view data as the actual objects the data are describing, and to query, add, delete, and analyze the data embedded in them simply by pointing and clicking on the mouse. GIS takes these analytical capabilities a step further by allowing the features themselves to be analyzed. In addition to the full set of tools available to most relational database management systems, such as logical operators, math and string functions, and a scripting language, GIS offers the unique ability to conduct spatial analysis on graphical features, which is a powerful tool beyond the scope of traditional database management systems (Harder, 1997). Thus, for logistics, a GIS can be used to determine the percentage of a company's retail market that lies within the designated service area of a warehouse, or it can find the number of raw materials suppliers within a day's transit time from a manufacturing facility, to name just a few spatial operations.

According to Dewitt *et al.*, GIS can be an eminently useful tool of logistics practitioners for visualizing weaknesses or problems in a logistics network, for creating mutual understanding among the various functional areas of a business, and for formulating solutions quickly.

> Perhaps most important to logistics is the ability of GIS to integrate all components of the logistics chain and to display their relationships in an intuitive and understandable manner. Before companies can re-think their logistics operations...they must first understand their current situation...Looking at alternative configurations displayed in several forms (maps, charts, and tables) provides those who must construct a logistics system with a common frame of reference. GIS gives logistics an effective decision support capability (Dewitt et al., 1997).

The most noteworthy example of a GIS application being applied in this way to a logistics network was Proctor and Gamble's global supply chain restructuring project that involved the integration of GIS with integer programming, and network optimization models (Camm *et al.*, 1997). Although much of the spatial analysis capabilities of current GIS technology were not employed in the project, Camm *et al.* cited the need to be able to "drill down" into the data underlying the network optimization model, as well as to provide a quick, interactive interface, as the main reasons for using GIS technology as the front end for the modeling system.

> We needed a simple interactive tool that would allow product-strategy teams to quickly evaluate options (choices of plant locations and capacities), make revisions, evaluate the new options, and so on. If possible, we wanted a system that would guide users to better options in an evolutionary fashion (Ibid.).

They credited the use of GIS with having increased user acceptance of the project's analytical techniques by making the modeling system's solution algorithm transparent to the user, while emphasizing the important spatial relationships inherent in logistics networks of suppliers, plants, warehouses, and customers. Another surprising byproduct of using GIS for visualization was its ability to highlight database errors that might not have been detected otherwise. In short, the integration of GIS with the network optimization model proved to be a powerful and flexible decision support system (DSS) for the supply chain project (Ibid.).

The CVS-Revco Merger as a Sample Logistics Problem

In 1996, the drugstore chain Consumer Value Stores (CVS) acquired a much larger competitor chain, Revco. Prior to the acquisition, CVS's market area was limited primarily to the northeast United States, although it enjoyed a strong presence in the mid-Atlantic states (Figure 1). Revco, on the other hand, was distributed throughout most of the states east of the Mississippi River, excluding the New England region (Figure 2). The acquisition was approved by the shareholders of both companies and, contingent on certain provisions, by the U.S. Federal Trade Commission. The resulting merger formed the second-largest drugstore chain in the U.S. market. While both companies hailed the merger as a profitable venture, merging them into one company actually posed several problems that could have undermined its success. One of these problems dealt with integrating the two different distribution networks that existed prior to the merger.

CVS had three distribution centers (DCs) serving approximately 1,400 stores, while Revco had six DCs serving approximately 2,500 stores. Table 1 lists each company's respective DC locations. Even before the merger was finalized, it was evident to CVS management that a restructuring of the resulting distribution network would have to occur. A brief look at DC service territories revealed various overlapping areas in the two networks, particularly around the mid-Atlantic states (Figure 3). To

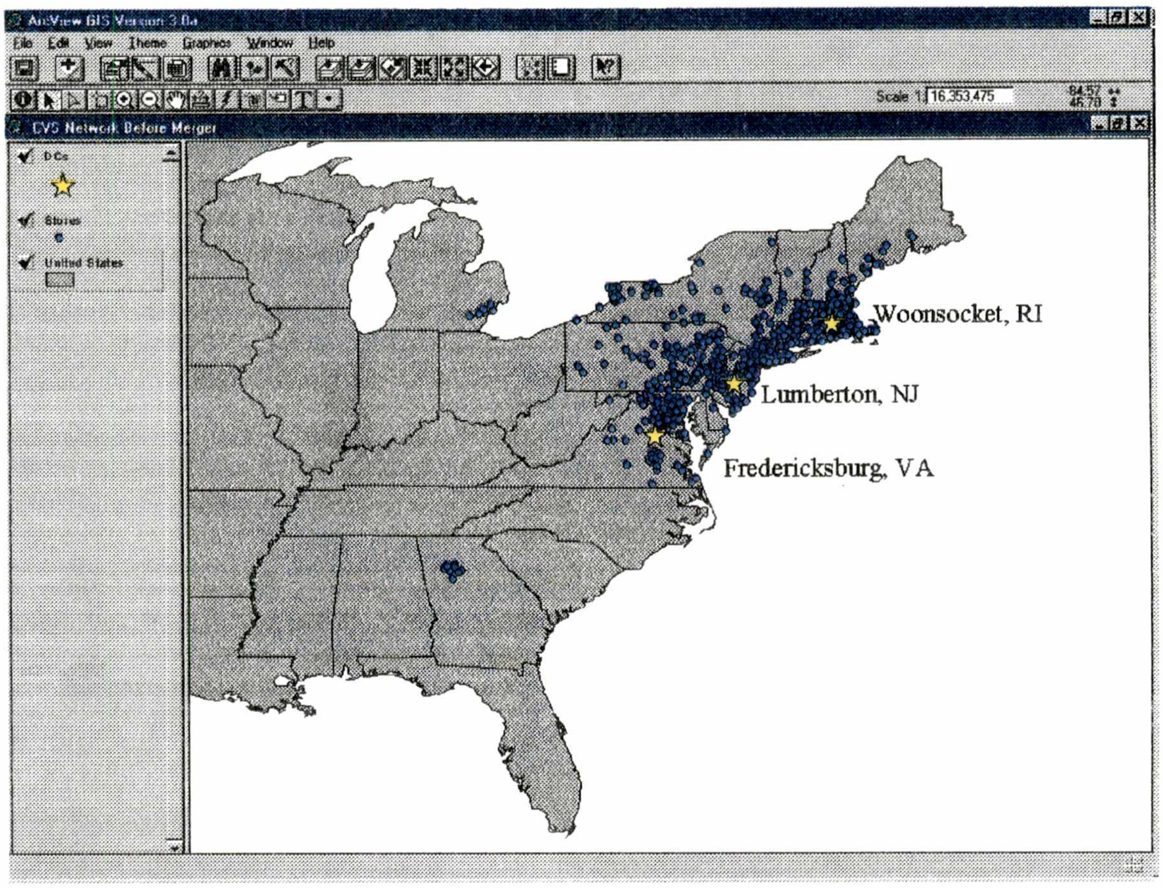


Figure 1. CVS stores and DCs before the merger with Revco.

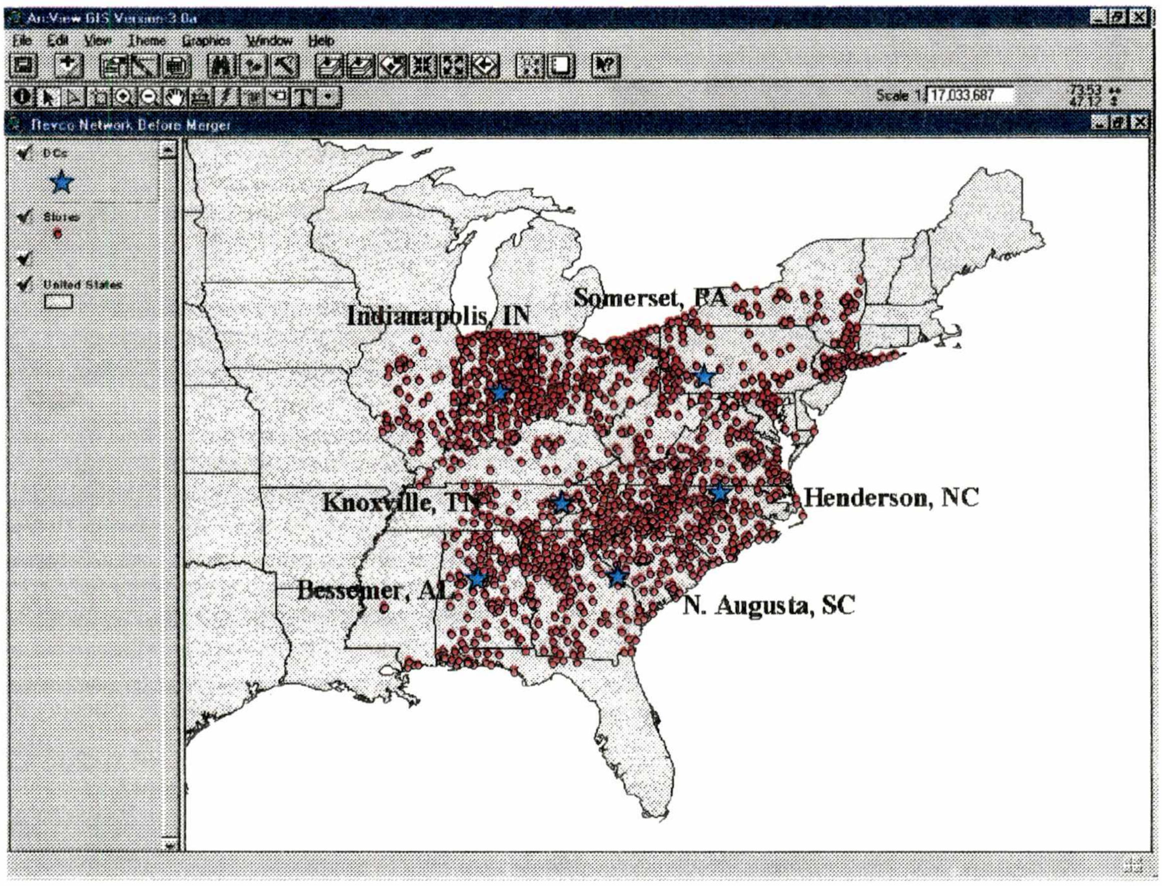


Figure 2. Revco stores and DCs before the merger with CVS.

Table 1. CVS and Revco DC locations

CVS	Revco
Woonsocket, Rhode Island Lumberton, New Jersey Fredericksburg, Virginia	Indianapolis, Indiana Somerset, Pennsylvania Knoxville, Tennessee North Augusta, South Carolina Henderson, North Carolina Bessemer, Alabama

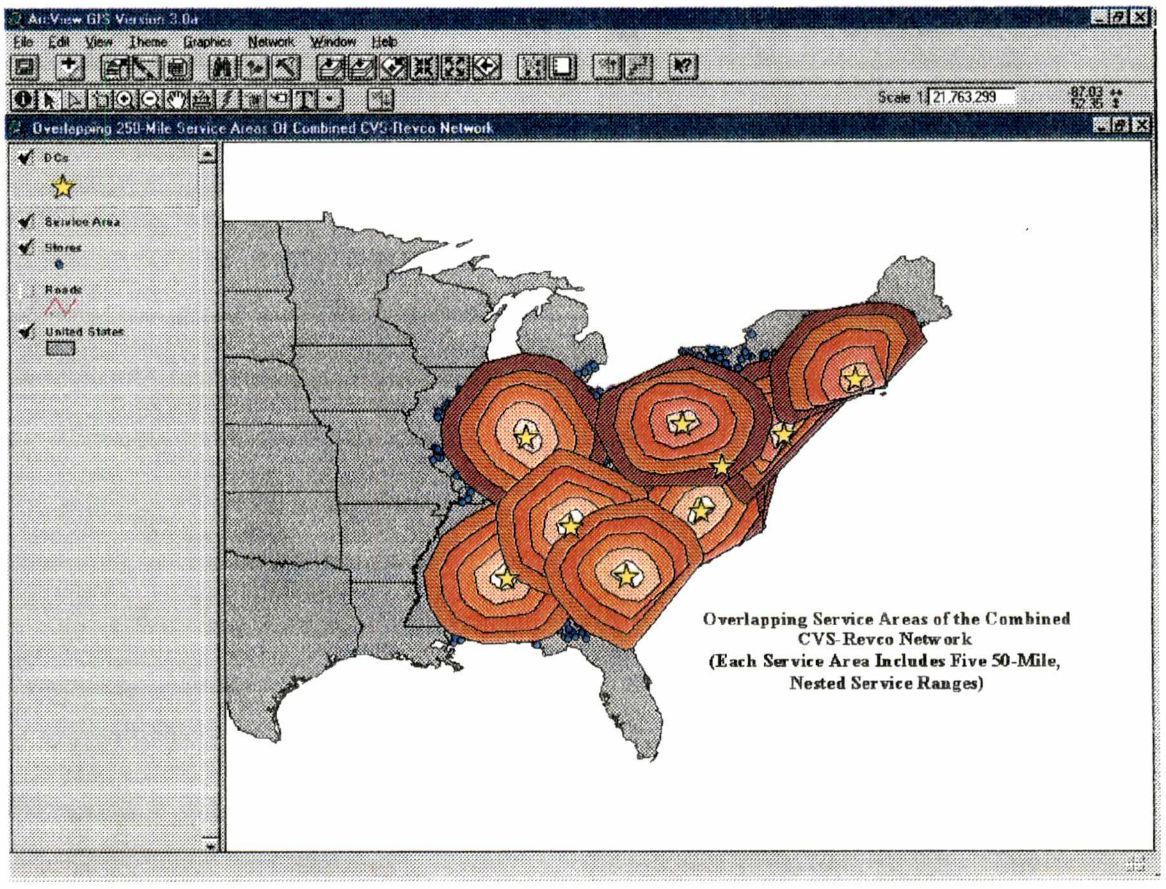

ArcView GIS Version 3.0a
File Edit View Theme Graphics Network Window Help
Scale 1: 21,763,299
87.03
52.35
Overlapping 250-Mile Service Areas Of Combined CVS-Revco Network
DCs
Service Area
Stores
Roads
United States
Overlapping Service Areas of the Combined CVS-Revco Network
(Each Service Area Includes Five 50-Mile, Nested Service Ranges)

Figure 3. Overlapping service areas of the combined CVS-Revco network.

make the new network more efficient, it was concluded that a reallocation of stores to DCs was required, at the very least. Further, some DCs might have to be closed, others might have to be expanded and improved, and perhaps new DCs would have to be constructed, in order for CVS to meet the challenges of its new market share and to accommodate future growth. In short, revamping CVS's new logistics system was a classic network optimization problem of the location/allocation type.

The CVS Distribution Network Optimization Project

During the summer of 1997, Dr. Charles Noon of the Department of Management Science, and Dr. Bruce Ralston of the Department of Geography, at the University of Tennessee, Knoxville, were contracted by CVS to work on this network optimization project using GIS as their primary analytical tool. As part of the contract, I was hired by CVS to help Drs. Noon and Ralston in the development of a GIS application using ArcView 3.0a with the Network Analyst extension, a product of Environmental Systems Research Institute (ESRI). The bulk of the work was to be done using ArcView's scripting language, Avenue.

In addition to our team, CVS had also contracted with Anderson Consulting to work on this project, since Anderson had worked previously with Revco on an earlier merger Revco had done with Alabama's Big B drugstore chain, and were therefore quite familiar with Revco's operations. The Anderson consultants were responsible for gathering the necessary data from CVS and Revco and would run the data through an optimization program tailored to the logistics industry called Supply Chain Strategist, a product of InterTrans Logistics Solutions, Inc. Our team was charged with importing the optimized network scenarios into ArcView and graphically displaying and analyzing them, and verifying the validity of the underlying data.

Thesis Outline and Disclaimer

Having placed the project in the context of the history of diffusion of GIS into the business world, and the utility of GIS in the field of business logistics, the remainder of this thesis will focus on the ArcView application we developed for CVS. The next chapter will provide an overview of the custom GIS functions we developed for the CVS project. Chapters 3; 4, and 5 will go into the detail of how these functions were programmed. Chapter 3 will address the scripts written for importing and visualizing the output of the logistics network optimization application. Chapter 4 will review the scripts written to analyze the total logistics cost for the retail facilities using the graphical user interface. Chapter 5 will discuss the scripts written for comparing the DC location strategies of CVS to those of its major competitors. Chapter 6 will conclude the thesis by reemphasizing the advantages of using the visualization capabilities of GIS to analyze and solve logistics system challenges, and by making some suggestions on how the project might have been improved. It will also make some final comments on the future of using GIS for logistics analysis, and on some emerging trends that foreshadow the convergence of GIS technology with mainstream information systems.

Before continuing, it should also be noted that the underlying data of the sample network solution used throughout this thesis have been altered and disguised in order to protect the sensitive and proprietary nature of this project and CVS corporate operations, and to comply with the non-disclosure clause of our team's contract with CVS. Apart from publicly available information, such as the location of the DCs and the stores, in no way does this thesis reveal any proprietary information about the company, its operations, or the project.

Chapter 2

Overview of the CVS Project's Custom GIS Functions

Importation and Display of the Optimization Scenario Data

Once the project had been outlined and our basic responsibilities defined, our team set about defining the tasks our portion of the overall project would require. Since many different optimized network scenarios would be generated during the project, one of the first tasks would be to automate and streamline, as much as possible, the data importation process. Accomplishing this task required more than just writing scripts. Since the data would be coming from several different systems (CVS, Revco, and Big B, the last of which still had not completed its merger with Revco), it would also require establishing with the Anderson consultants a consistent policy for data content and formatting. The details of how this data importation was automated will be presented in Chapter 3. Once the data had been imported, we could then begin to display and analyze them.

One of the foremost questions regarding the display of data was: what kind of information would a logistics manager want to see rendered by the GIS software? Obviously, the first aspect of the network one would expect to see on a map would be the stores and the DCs. Since the network optimization performed by Anderson Consulting aggregated the 4,000 stores into a more manageable set of 384 demand regions based on 3-digit zip codes, a view of the demand regions would also be desired. Figure 4 is an example of a map from the sample network solution showing the demand regions and the

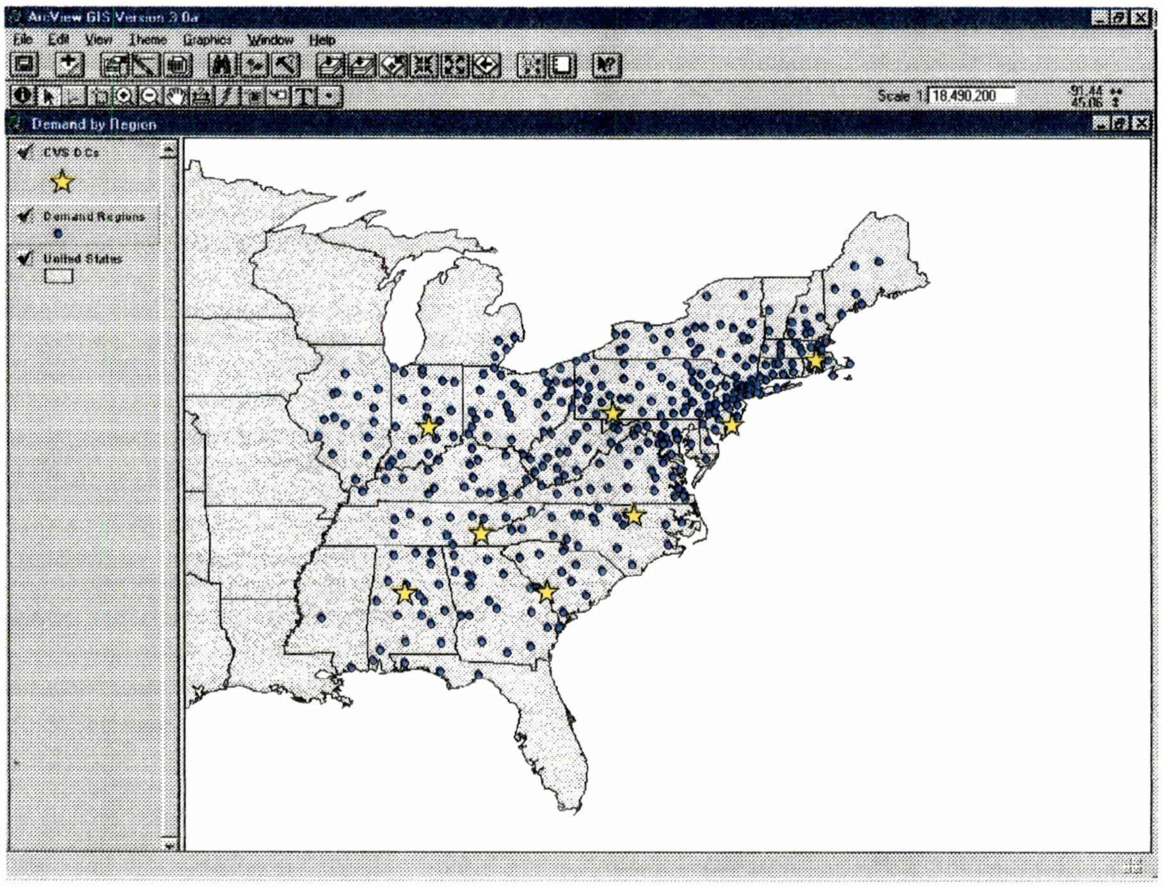


Figure 4. CVS DCs and demand regions for sample optimized network. The gold stars represent the DCs. The demand regions are represented by the blue dots.

DCs. From this starting point, it seemed logical that the next view of the network should include information about the flow of product from the DCs to the stores based on the allocation of demand regions to DCs determined by the optimization software. These flowlines could be simple lines connecting the DCs to the stores, or they could be lines displayed with graduated thicknesses that change according to the total flow volume occurring on each one.

In addition to total product flow volumes, Anderson Consulting's optimization of the new CVS network also broke down product flow into three major categories: prescription drugs and products (Rx), non-prescription products available "over-the-counter" (OTC), and slow-selling, bulky, or seasonal products that are centrally warehoused (CW). Given the availability of these data, a logistics manager would want to have the choice of seeing how product volumes flow throughout the network, or might wish to have these flows displayed by product type.

Figure 5 shows an example of how the flow of Rx products over the sample optimized network can be displayed. Gold stars indicate the locations of the CVS DCs, while the optimized flows of Rx product are displayed as red lines. Notice that the volume of flow is indicated by the thickness of the line. This type of display provides a quick and efficient overview of how the DCs are allocated to serve the demand regions and where the largest flow volumes are occurring in the network. For example, it is readily apparent that several of the lines in the Kentucky area cross each other, which is a signal to the logistics manager that the network is not completely optimal. Displaying the flow volumes with graduated line thickness offers useful insight about how transportation fleets should be assigned. It is also important to keep in mind that the user can zoom in

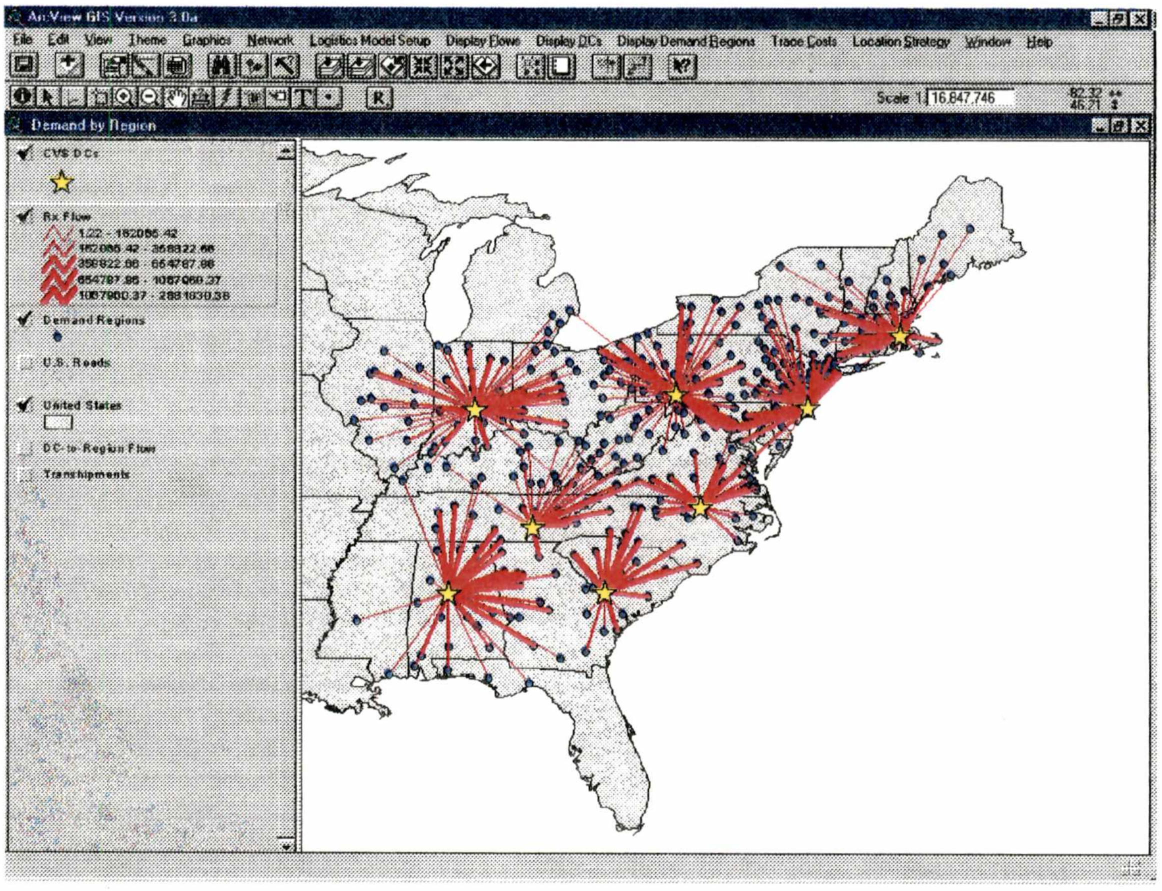


Figure 5. Prescription product flows over sample optimized network. Red lines whose thickness indicates volume represent flows. The gold stars represent DC locations.

and out on any portion of the view, and can query information about individual instances of the flow line and DC point features.

Next, it would be desirable to see where any transshipments between DCs are taking place and what are their volumes of flow. Once again, these transshipments should be sized to indicate flow volume, and a choice should be available of viewing total product flow, or the flow of specific product types. Moreover, it would be helpful to view the DCs according to their total demand, by the total handling costs incurred at the DC, by the fixed cost of the facility, or any other data that might be available.

Figure 6 shows a view of the Rx product transshipments and the DCs sized according to handling cost. Green lines whose thicknesses represent flow volume indicate the transshipments. In this scenario, the Indianapolis, Indiana, DC stocks and supplies Rx products for the Somerset, Pennsylvania, DC. Knoxville supplies Bessemer, Alabama, N. Augusta, South Carolina, and Henderson, North Carolina. Since the Lumberton, New Jersey, DC and the Woonsocket, Rhode Island, DC both stock Rx products, they do not have any Rx transshipments. If this is the case, however, then this display immediately begs the question of why the Indianapolis DC, and not the much closer Lumberton DC, provides the transshipments to the Somerset DC. Of course, the answer to that question may be more complicated than just a matter of distance, but the point has been made as to how such visual displays of a logistics network lead immediately to questions that aid in the understanding and analysis of the logistics network.

The only component of the network that remains unmentioned is the set of demand regions. The optimization program also generates data on each demand region's

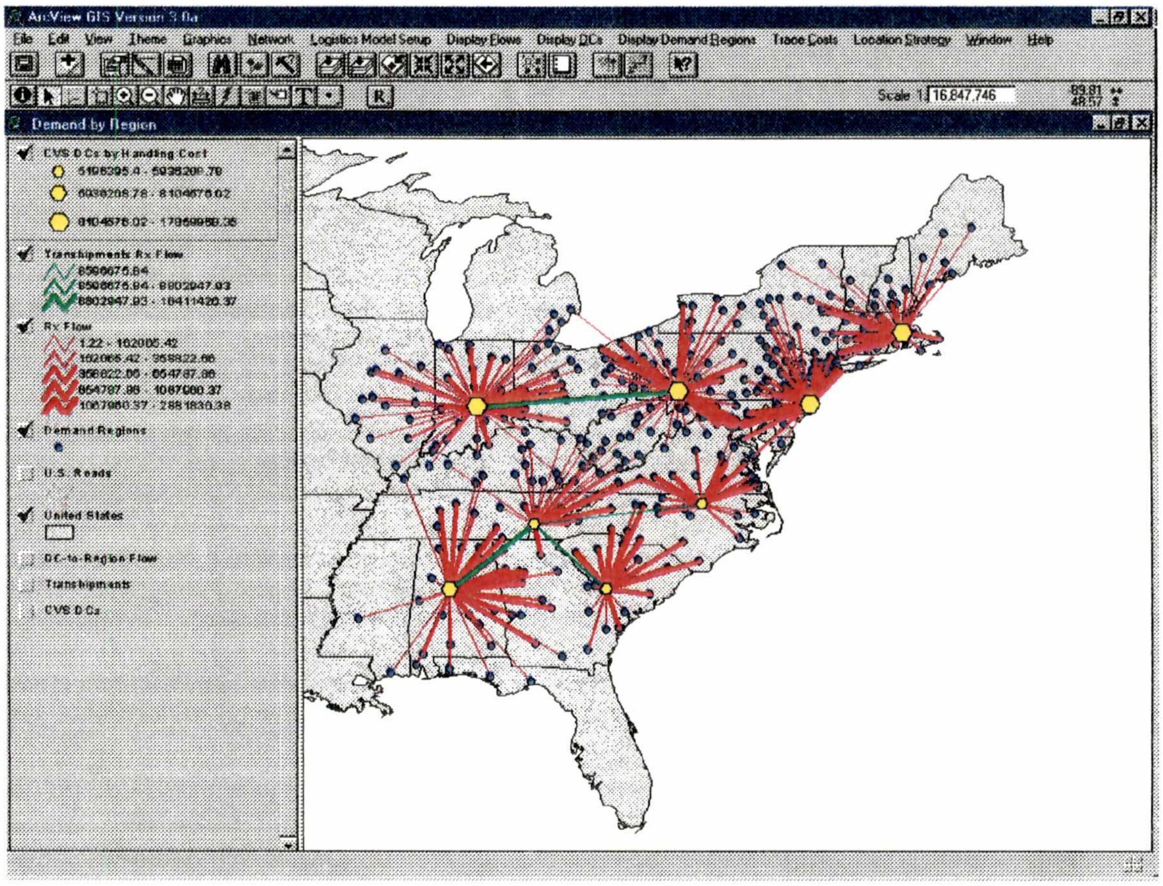


Figure 6. Prescription product transshipments and DC handling costs on sample network. Green lines whose thickness indicates volume represent transshipments. The size of the DC (hexagons) indicates total handling cost.

total demand for product, and on the demand for the different product categories. One informative way to display all of these data simultaneously is to portray each demand region as a pie chart in which the slices represent the composite demand for the three product types – Rx, OTC, and CW – while the size of the pie itself represents the total demand.

Figure 7 shows how this looks on a map of the sample network that zooms in and centers on the Bessemer, Alabama, distribution center. OTC product volume is displayed as a blue pie slice, while Rx product and CW product are displayed as red and green respectively. This map also provides good evidence of one reason why Rx and CW products are centrally warehoused. Most of the total product volume for the demand regions displayed is composed of OTC product. The demand for Rx and CW product is sufficiently small to warrant centralized stocking.

Tracing Demand Region Logistics Costs

Displaying data about a logistics network using graduated symbols and colors on graphical features is helpful when the goal is to get a general sense of the relative volumes or costs of an individual component of the network, such as the lines of flow between DCs and demand regions, or the points representing the origin and destination facilities of a flow. Calculating the total logistics cost for an individual demand region, however, requires tracing the product flow from the demand region back up through the network to the originating facility. Such a calculation involves summing the several components of the network, such as the shipping costs on the flow line and the picking cost at the DC. If transshipments between DCs are involved, then a crossdocking cost at

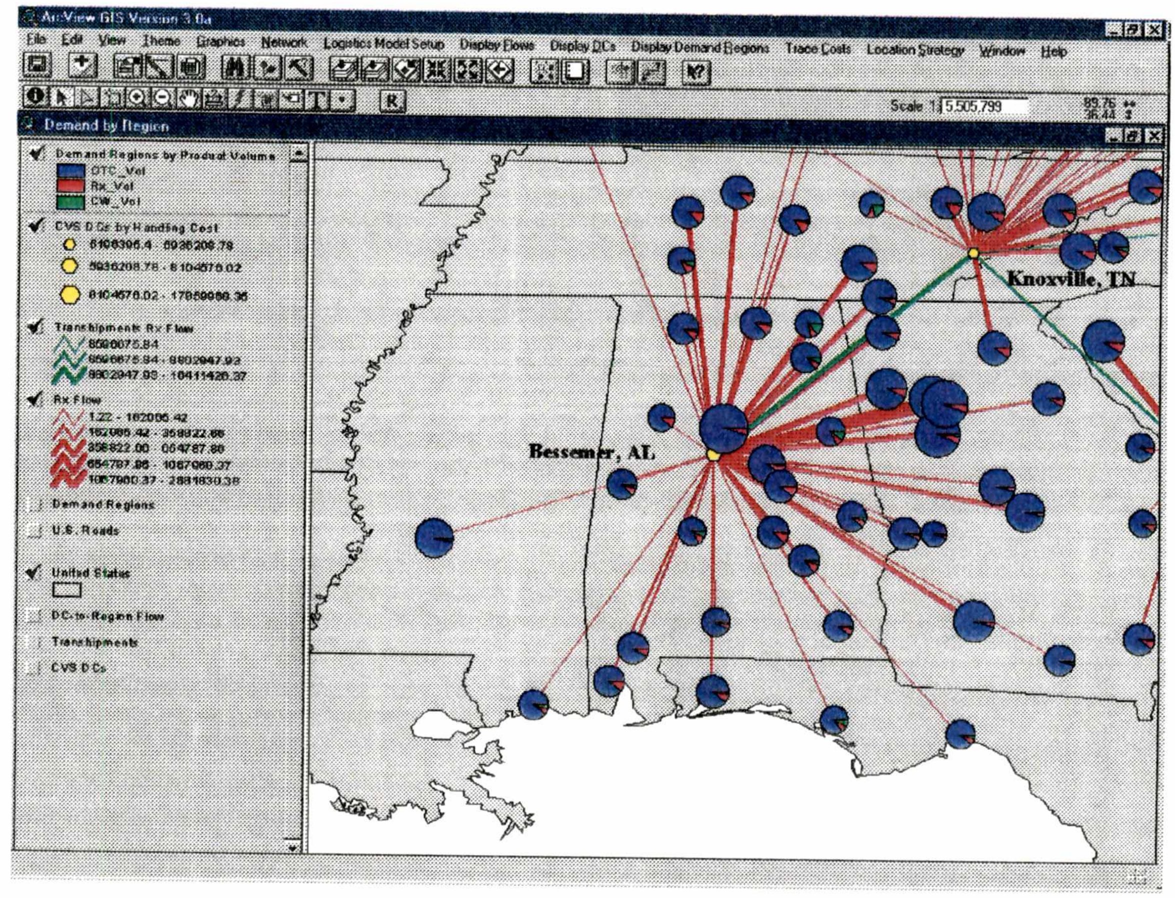
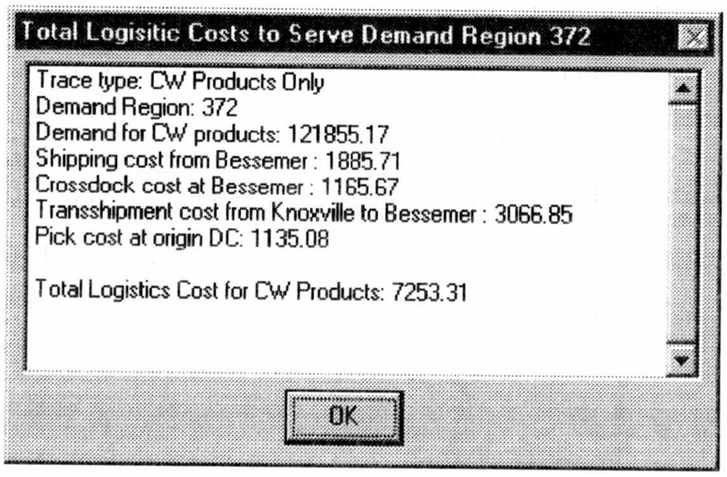


Figure 7. Demand Regions displayed as pie charts. Pie slices represent OTC (blue), Rx (red), and CW (green) product volumes, while the size of the pie indicates total product volume.

the distributing DC and a transshipment cost on the inter-DC flow line, must be added to the shipping and picking costs. In this situation, the value of GIS comes not so much from its graphical capabilities, but from its relational database.

Using the DCs as the common key, shipment tables and transshipment tables can be linked in a GIS, and information about all the related components of a particular branch of the network can be accessed by performing a single query on the branch's endpoint, the demand region. Employing the unique ability of GIS to query data embedded in an object by clicking on it, we developed a tool that would allow a logistics manager to click on a demand region and receive an itemized report of the demand region's logistics costs. Once again, this type of query would be based on total products or specific product types. Figure 8 provides an example of the report's pop-up window after clicking on one of the demand regions to trace total CW logistics costs.

Of course, if this logistics cost tracing could be done for a single demand region, it could also been done for the entire set of demand regions. We decided, therefore, to



```
Trace type: CW Products Only
Demand Region: 372
Demand for CW products: 121855.17
Shipping cost from Bessemer : 1885.71
Crossdock cost at Bessemer : 1165.67
Transshipment cost from Knoxville to Bessemer : 3066.85
Pick cost at origin DC: 1135.08

Total Logistics Cost for CW Products: 7253.31
```

Figure 8. Sample report of total CW product logistics cost for an individual demand region.

build in the capability of generating a table containing the component and total logistics cost for all of the demand regions, not only for total products, but also for specific product types. These data would enable the application user to view the entire network of demand regions according to their logistics costs, and would shed light on which regions are incurring the most costs. As with the total demand data for the demand regions, this data would be displayed using graduated pie chart symbols that would simultaneously represent the total logistics costs and the component logistics costs for each demand region. Figure 9 shows a view of the demand regions portraying the total and component logistics costs for Rx products. The view is zoomed in and centered on the Somerset, Pennsylvania, and Lumberton, New Jersey distribution centers. Notice that the Lumberton DC stocks Rx products, therefore the demand regions it serves do not incur crossdock or transshipment costs, unlike the Somerset DC, which gets its Rx product from the Indianapolis DC.

Having this table of total and component logistic costs for all demand regions also made it very easy to add a function that would generate a report of logistic costs similar to the one for the individual demand region, but for the entire network. Once again, since logistic cost tables can be made for each product category, as well as for all products combined, it would be possible to generate a total logistic cost report for any of these tables. Figure 10 provides an example of the total Rx costs report for the entire CVS sample network.

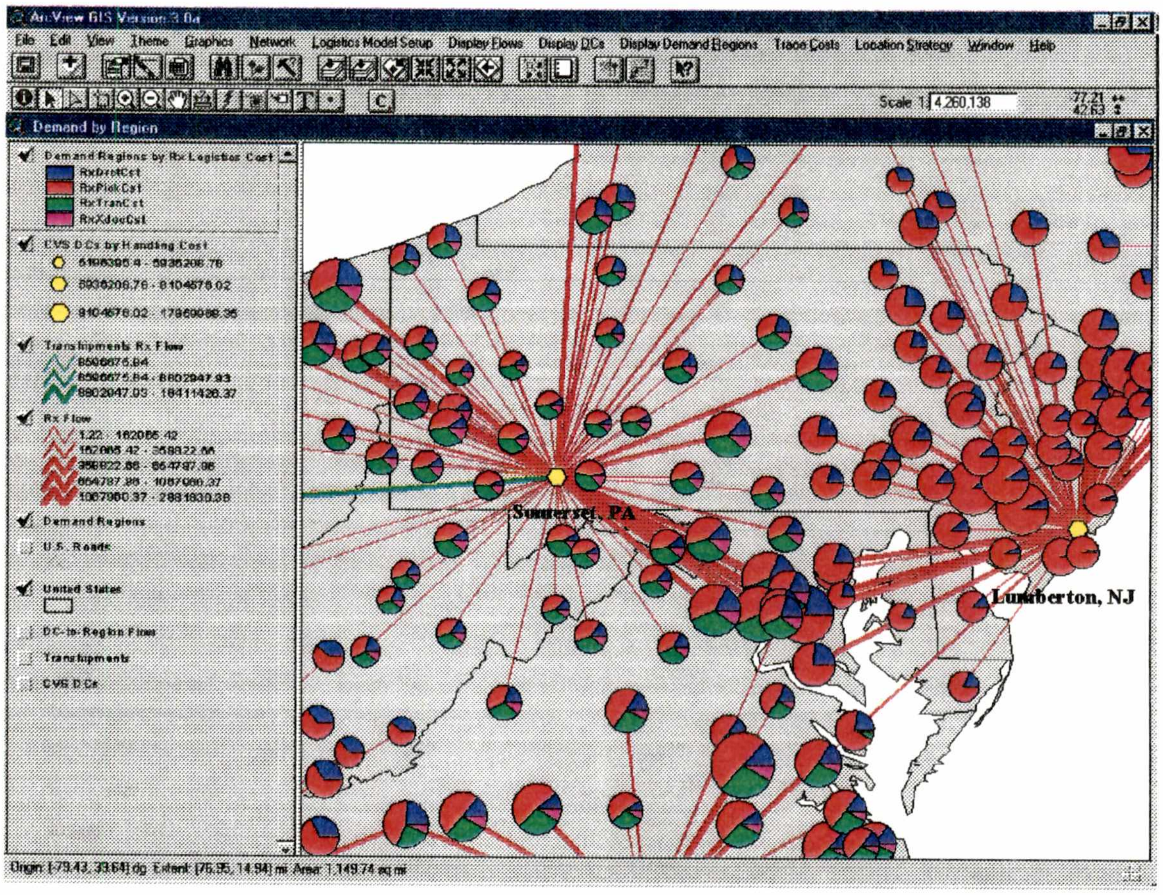
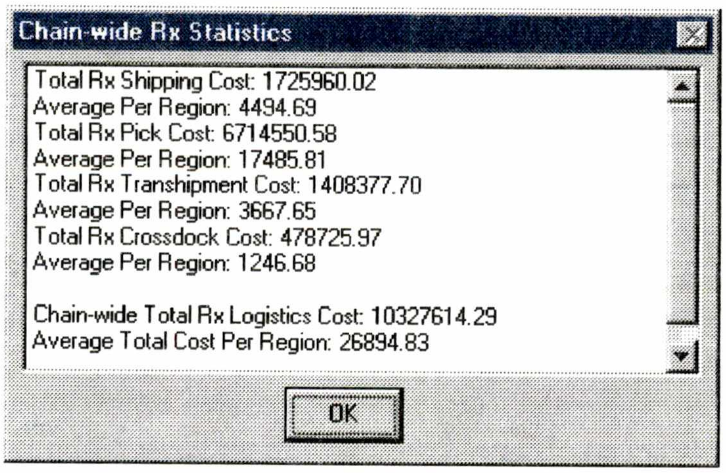


Figure 9. View of demand regions portrayed according to total Rx logistics costs. Pie slices represent shipping (blue), picking (red), transshipping (green), and crossdocking (magenta) costs for Rx products for the demand region, while the size of the pie represents the total Rx product logistics costs for the demand region.



```
Total Rx Shipping Cost: 1725960.02
Average Per Region: 4494.69
Total Rx Pick Cost: 6714550.58
Average Per Region: 17485.81
Total Rx Transhipment Cost: 1408377.70
Average Per Region: 3667.65
Total Rx Crossdock Cost: 478725.97
Average Per Region: 1246.68

Chain-wide Total Rx Logistics Cost: 10327614.29
Average Total Cost Per Region: 26894.83
```

Figure 10. Report of total and component Rx logistic costs for entire sample network.

Analyzing DC Location Strategies

Another functionality we decided to develop involved harnessing the ability of GIS to analyze the spatial relationship between geographically referenced features. The goal of CVS's network optimization was to find a configuration of DCs that would best serve the demand regions while lowering overall network costs. The value of a given optimized network scenario would be judged against a benchmark, such as the network costs being incurred prior to optimization. Another type of benchmark involves using the practices of a company's competitors within the industry as a standard for judging changes to the organization. It was felt, therefore, that some way of measuring each potential DC configuration against an industry benchmark would also be beneficial to a logistics manager.

To compare each optimized scenario against the network costs of competitor companies was not practical, since operational cost data are not publicly available from most companies. However, data on DC and store locations are publicly available, and with these data we could at least analyze the spatial patterns between store and DC locations that each company's network exhibited. Around each DC of a company's network, we would build a set of five 50-mile, nested distance ranges (based on road network of major highways), effectively covering a radius of 250 miles around each DC (Figure 11). Using a select-by-theme query process, we determined the distance of each store from its nearest DC. Using this information, we could then generate a histogram for each company detailing the percentage of stores located within 50, 100, 150, 200, and 250 miles of the nearest DC, and the percentage that falls outside of this service area. In Figure 12, the histogram of DC-store distances for the sample CVS network is shown

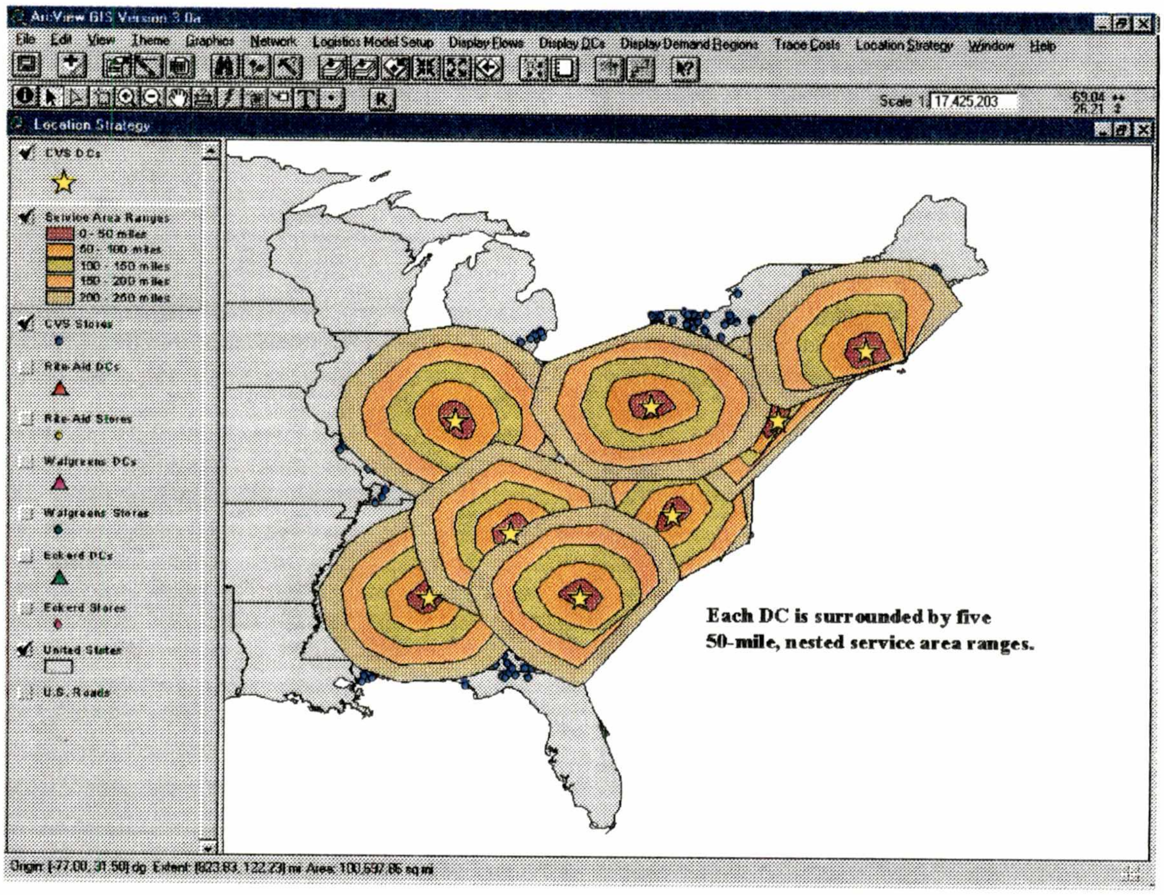

ArcView GIS Version 3.0a
File Edit View Theme Graphics Network Logistics Model Setup Display Flows Display DCs Display Demand Regions Trace Costs Location Strategy Window Help
Scale 1: 17,425,203
Location Strategy
CVS DCs
Service Area Ranges
0 - 50 miles
50 - 100 miles
100 - 150 miles
150 - 200 miles
200 - 250 miles
CVS Stores
Rite-Aid DCs
Rite-Aid Stores
Walgreens DCs
Walgreens Stores
Eckerd DCs
Eckerd Stores
United States
U.S. Roads
Each DC is surrounded by five 50-mile, nested service area ranges.
Origin: [-77.00, 31.50] dg Extent: [823.83, 122.23] mi Area: 100,597.86 sq mi

Figure 11. Five 50-mile, nested service ranges around each DC in the CVS sample network. Gold stars represent DC locations

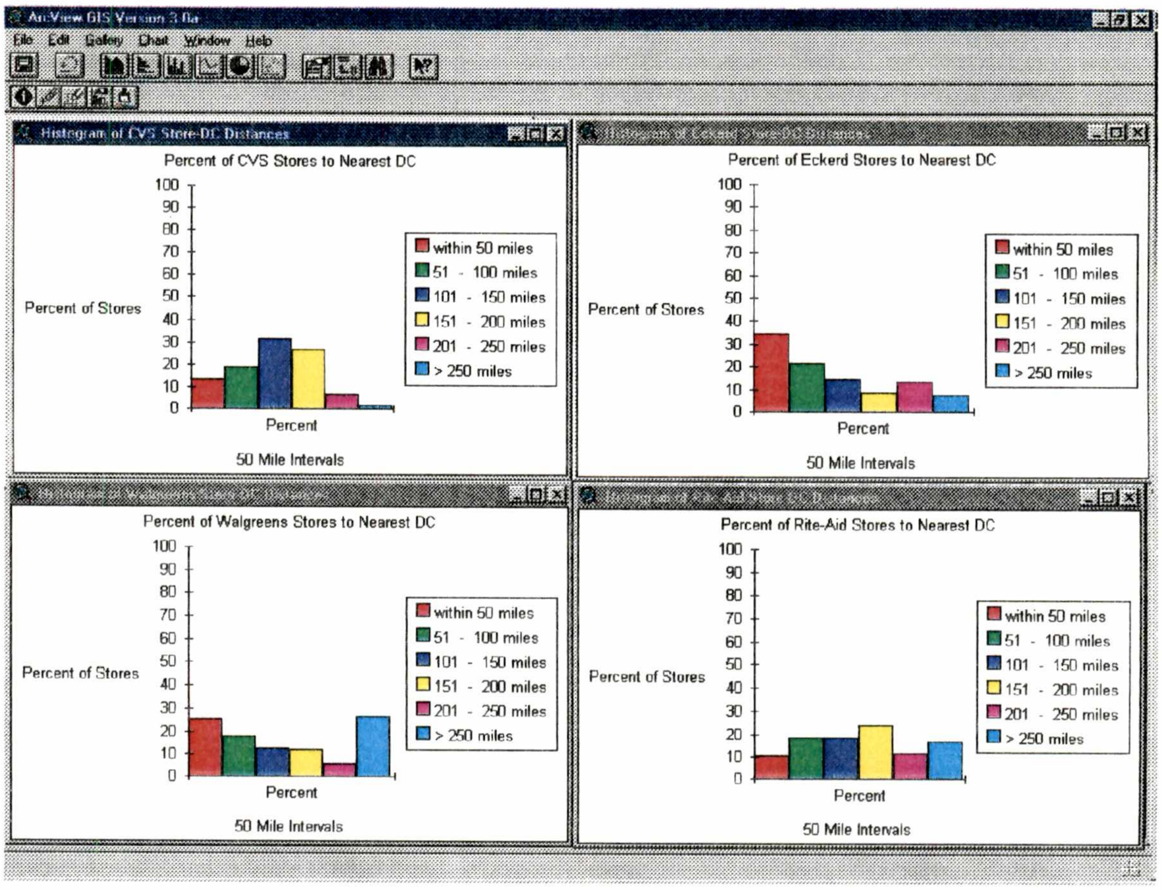

ArcView GIS Version 3.0a
File Edit Gallery Chart Window Help
Histogram of CVS Store-DC Distances
Percent of CVS Stores to Nearest DC
Percent of Stores
Percent
50 Mile Intervals
within 50 miles
51 - 100 miles
101 - 150 miles
151 - 200 miles
201 - 250 miles
> 250 miles
Percent of Eckerd Stores to Nearest DC
Percent of Walgreens Stores to Nearest DC
Percent of Rite-Aid Stores to Nearest DC

Figure 12. Histograms showing the percent of stores to nearest DC for CVS and its major competitors.

alongside the histograms for the networks of CVS's major competitors: Walgreens, Rite-Aid, and Eckerd.

This review of the custom GIS functions developed for the CVS network optimization project illustrates the visual display resulting from the application of these functions, and the value these displays bring to the network optimization process. These functions were automated using ArcView's scripting language, Avenue, so that each of these functions can be launched with the simple click of a menu item. The following three chapters of this thesis will address the details of the scripts controlling these functions.

Before turning to those scripts, however, it will be useful to describe the GUI developed for the CVS network optimization project. Table 2 lists the several custom menus that were added to ArcView's default View menubar, along with the menu items that appear in the drop-down boxes when a particular menu is clicked. The custom functions developed for the project are controlled using these menu items. As I address these functions in further detail in the chapters that follow, I will refer to these menu items by name.

Table 2. Custom menus added to the ArcView default View menubar.

	Custom Menus					
	Logistics Model Setup	**Display Flows**	**Display DCs**	**Display Demand Regions**	**Trace Costs**	**Location Strategy**
Menu Items	Get MS Access Tables	DC-to-Region by Rx Only	By Handling Cost	By Product Volume	Chain-wide Rx Only	DC-Store Range
	Build Transport Lines	DC-to-Region by CW Only	By Fixed Cost	By Total Demand	Chain-wide CW Only	Make Histogram
	Summarize DC Data	DC-to-Region by OTC Only	By Total Demand	By Rx Logistics Cost	Chain-wide OTC Only	
	Summarize Demand Region Data	DC-to-Region by Total Flow		By CW Logistics Cost	Chain-wide All Products	
		Transshipments by Rx Only		By OTC Logistics Cost	Chain-wide Rx Statistics	
		Transshipments by CW Only		By Total Logistics Cost	Chain-wide CW Statistics	
		Transshipments by Total Flow			Chain-wide OTC Statistics	
					Chain-wide All Products Statistics	
					Demand Region Rx Only	
					Demand Region CW Only	
					Demand Region OTC Only	
					Demand Region All Products	

Chapter 3

Importing and Visualizing the Optimized Logistics Network

Importing the Microsoft Access Tables

It has been noted earlier that Anderson Consulting collected the necessary data from CVS and Revco and conducted the post-merger network optimization. After the network optimization software had arrived at a solution, Anderson Consulting exported the solution data from the optimization software into several tables within a Microsoft Access database. For the purposes of constructing visual displays of the optimized network, it was necessary to import five of these tables into ArcView. Table 3 provides a listing of these tables and the fields pertinent to the display of the network solution.

The first step in using ArcView to analyze the logistics information is to input the Microsoft Access tables. This must be done before any other steps can be carried out. When the user starts the project in ArcView to begin the display of a new network solution, all the custom menu items (Table 2) are disabled ("grayed out"), except for Get MS Access Tables. Clicking this item launches a script, SQLTables.Get (Appendix), which establishes a structured query language (SQL) connection, called an SQLCon, with the Microsoft Access database application. An SQLCon enables the user to import an entire table or a subset of a table based upon the query string that is passed to the host application. The result of the query is automatically imported into the ArcView application as an SQL virtual table from which a table document can then be made.

Table 3. Microsoft Access tables and their pertinent fields.

Table Name:	DIRECT TO STORE		
	Field Name	*Field Type*	*Description*
	Facility	Text	Name of DC
	DemandRegion	Text	Number ID of Region
	Product	Text	Product Category Name
	OptimizedValue	Number	Product Flow Volume in Units
	ActualRate	Number	Shipping Rate per Unit

Table Name:	HANDLING		
	Field Name	*Field Type*	*Description*
	Facility	Text	Name of DC
	Product	Text	Product Category Name
	HandlingRate	Number	Handling Rate per Unit (applies to Picking and Crossdocking)

Table Name:	INPUT - FACILITIES		
	Field Name	*Field Type*	*Description*
	Facility	Text	Name of DC
	LatLon	Text	Geographic Coordinates of DC
	FixedCost	Number	Fixed Cost of DC
	OptimizedValue	Number	Product Flow Volume in Units

Table Name:	PICKING		
	Field Name	*Field Type*	*Description*
	Facility	Text	Name of DC
	Process	Text	Centralized Product Stocking Point Indicator

Table Name:	TRANSHIPMENTS		
	Field Name	*Field Type*	*Description*
	OriginFacility	Text	Name of Origin DC
	DestinationFacility	Text	Name of Destination DC
	Product	Text	Product Category Name
	OptimizedValue	Number	Product Flow Volume in Units
	ActualRate	Number	Transshipping Rate per Unit

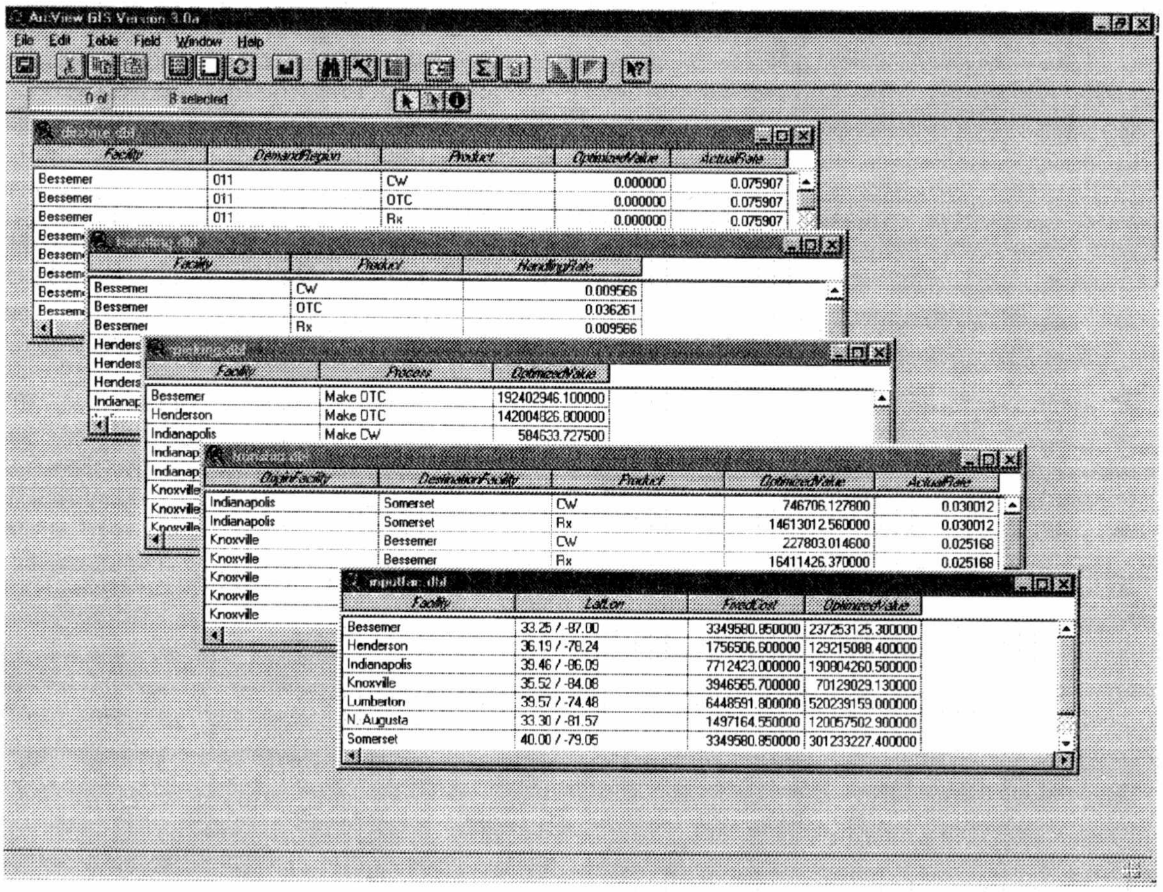
While this method of querying a database is quite easy, it has one drawback. The virtual table resulting from the SQLCon is only a visual representation of the data residing in the host database and is not tied to its own location in memory as a separate file. As such, the virtual table is a read-only document. To give the project read and write access to these tables, and also to dispense with the SQLCon, which is no longer needed, SQLTables.Get takes each virtual table and exports it as a dBase file, thus giving it its own pathname and location in memory. These files are then imported again back into the ArcView project (Figure 13) and given new names, as listed in Table 4.

At this point it may be helpful to define several objects in the ArcView's Avenue scripting language which will be referenced frequently in this thesis (Figure 14). The object hierarchy of Avenue contains five basic documents visible to the user while the application is running. These are Tables, Views, Charts, Layouts, and ScriptEditors. Of these, we are mainly concerned with Views and Tables.

Views are windows in which digital maps are rendered. Views are composed of Themes, which are often referred to as map layers. The most common type of Theme in ArcView is a feature theme, or FTheme. An FTheme is a set of similar geometric shapes – such as points, lines, and polygons – that represent geographically referenced objects, such as buildings, roads, or census tracts. These shapes are displayed on the screen with graphical symbols that may be sized and colored in many ways to reveal important information about the objects. A common way to alter a theme is to classify the data underlying the theme. Classifying data means grouping the data according to the values in one of the data fields. Classifications may be illustrated with graduated colors or

Table 4. Name changes of MS Access tables after export and re-import into ArcView.

MS Access Table Name	*ArcView Table Name*
DIRECT TO STORE	dirstore.dbf
HANDLING	handling.dbf
INPUT – FACILITIES	inputfac.dbf
PICKING	picking.dbf
TRANSHIPMENTS	tranship.dbf



Facility	DemandRegion	Product	OptimizedValue	ActualRate
Bessemer	011	CW	0.000000	0.075907
Bessemer	011	OTC	0.000000	0.075907
Bessemer	011	Rx	0.000000	0.075907

Facility	Product	HandlingRate
Bessemer	CW	0.009566
Bessemer	OTC	0.036261
Bessemer	Rx	0.009566

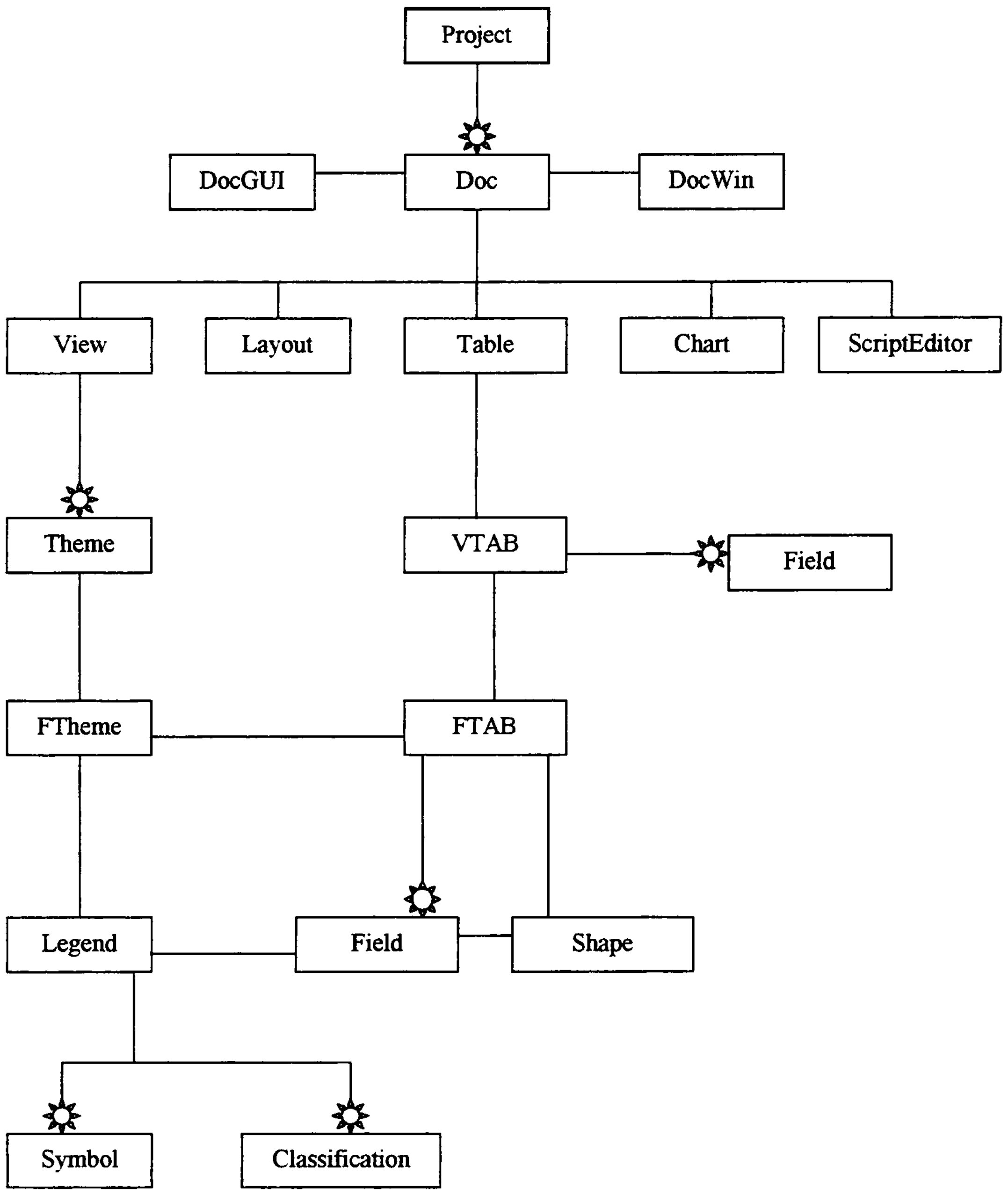
Facility	Process	OptimizedValue
Bessemer	Make OTC	192402946.100000
Henderson	Make OTC	142004826.800000
Indianapolis	Make CW	584633.727500

OriginFacility	DestinationFacility	Product	OptimizedValue	ActualRate
Indianapolis	Somerset	CW	746706.127800	0.030012
Indianapolis	Somerset	Rx	14613012.560000	0.030012
Knoxville	Bessemer	CW	227803.014600	0.025168
Knoxville	Bessemer	Rx	16411426.370000	0.025168

Facility	LatLon	FixedCost	OptimizedValue
Bessemer	33.25 / -87.00	3349580.850000	237253125.300000
Henderson	36.19 / -78.24	1756506.600000	129215088.400000
Indianapolis	39.46 / -86.09	7712423.000000	190804260.500000
Knoxville	35.52 / -84.08	3946565.700000	70129029.130000
Lumberton	39.57 / -74.48	6448591.800000	520239159.000000
N. Augusta	33.30 / -81.57	1497164.550000	120057502.900000
Somerset	40.00 / -79.05	3349580.850000	301233227.400000

Figure 13. ArcView tables created from Microsoft Access tables.

Avenue Object Model



☼ = Composed of zero or more (i.e., a project may be composed of zero, one or more Docs)

Figure 14. A partial model of Avenue objects relating to this thesis.

graduated symbol sizes. Control of a theme's symbols and classifications takes place in the theme's Legend.

Tables are documents built of records and fields that are common to most relational databases. A special type of Table called an attribute table holds the data upon which a Theme is built. The data for Tables and for Themes actually reside in memory as files. To access these files, Avenue provides an object called a virtual table (VTAB) that acts as an interface between the file and the programmer. A derivative of the VTAB is the feature table (FTAB), which provides the interface between the programmer and the file supporting the Theme object. Each FTAB automatically contains a Shape field to hold (and hide) the theme's geographic references. Except in the case of the SQL VTAB, creating a new VTAB or FTAB also creates a new file which is allocated space in memory. Further, creating a new FTAB creates a new shape, or map layer.

Lastly, Tables and Views, as well as the other types of Docs have a DocGUI, which is the collection of menus, buttons, and tools specific to that document type. They also have a DocWin, which is the visible window object we see on the computer screen in which Themes are rendered and the data from VTABs and FTABs are presented.

Building the Component Features of the Network

Once the tables listed in Table 4 have been created, the GUI is updated so that the Build Transport Lines menu item is enabled, which means the transport lines can now be created. Clicking on this menu item launches a script TransportationLines.Build (Appendix). This script is a master script that calls other scripts that build the DCs point theme, the DC-to-Region Flow line theme, and the Transshipments line theme. Figure 15 provides an outline of the flow of these scripts and what each one does.

Building the DCs Point Theme

The first script called by the TransportationLines.Build script is the FlowLine.Build script (Appendix) that creates a line theme connecting all the possible origin-destination pairs that exist between DCs and demand regions.
Before the script can do this, however, the DCs theme must be built (the Demand Regions theme is considered static for all network optimization scenarios, so it has already been constructed and appears in the project's view from the start). Since the inputfac.dbf table contains the geographic coordinates of each DC in the LatLon field, the FlowLine.Build script first acquires the VTAB for that table. Having done that, it calls the script SpliceLatLon (Appendix), and passes the inputfac.dbf VTAB as the argument. SpliceLatLon creates a new FTAB for a point theme, clones the fields in the inputfac.dbf field listed in Table 3 (except for the LatLon field), and adds them to the new FTAB. Then, for each record in the inputfac.dbf VTAB, it parses the LatLon field into its component latitude and longitude coordinates and adds them to the Shape field of the new record in the FTAB. At the same time, it copies the values from the other fields in the inputfac.dbf VTAB into the new FTAB. When the script is complete, a new point theme of the DCs has been created that contains data on each DC's name, fixed cost, and its optimized demand (i.e. supply) value. Just before the DCs FTAB is returned to FlowLine.Build as an argument, it is passed to the script AddXY, which creates two new

```
TransportationLines.Build script launched by click on Build Transport Lines menu item
1) Call FlowLine.Build script
   a) Call SpliceLatLon script with inputfac.dbf VTAB as argument
      i) Get latitude and longitude of each DC from VTAB
      ii) Build DCs point theme
      iii) Call AddXY script with DCs FTAB as argument
         (1) Add X and Y coordinate fields to DCs FTAB and populate them
      iv) Return with DCs FTAB as argument
   b) Build DC-to-Region Flow line theme using X and Y coordinate fields in
      both the DCs theme and the Demand Regions theme
   c) Call FlowValues.Calculate script with Flow FTAB as argument
      i) Create fields in Flow FTAB to hold volume data on Rx, CW, and OTC
         product flow, as well as total product flow
      ii) Select flows by commodity type from dirstore.dbf VTAB and transfer
          to new flow fields in Flow FTAB; sum them to populate total product flow field
      iii) Return with DCs FTAB as argument
2) Call TranshipLine.Build script with DCs FTAB as argument
   a) Build Transshipments line theme using the DCs as both the beginning points and the
      endpoints of the new lines
   b) Call TransFlowValues.Calculate script with Transshipments FTAB as argument
      i) Create fields in Transshipments FTAB to hold volume data on Rx,
         CW, and OTC product transshipments, as well as total product transshipments
      ii) Select flows by commodity type from tranship.dbf VTAB and transfer to new flow
          fields in Transshipments FTAB; sum them to populate total product transshipments
          field
   c) Return control to TransportationLines.Build script
3) TransportationLines.Build terminates
```

Figure 15. Outline of scripts launched by the Build Transport Lines menu item.

fields in the FTAB called X- coord and Y-coord and adds the longitude and latitude coordinates for each DC into those fields, respectively.

Building the DC-to-Region Flow Line Theme

FlowLine.Build then proceeds to build the DC-to-Region Flow FTAB (a map of flow lines). First it finds the Facility, X-coord, and Y-coord fields in the DCs FTAB, then it gets the Demand Regions FTAB, and finds the name field, as well as two fields holding each demand region's latitude and longitude. It then creates a new FTAB for a line theme, and adds fields to hold the DC and demand region names. The script then loops through each record of the DCs FTAB. For each DC, it loops through the Demand Regions FTAB and creates a line connecting that DC and the current record in the Demand Regions FTAB. The line is created by using the current DC's coordinates as the beginning point, and the current demand region's location as the end point, and adding them to the DC-to-Region Flow FTAB's shape field. The names for the DC and the demand region are also added to their respective fields. When this script is complete, a new line theme representing all possible flows between DCs and demand regions in the network has been created and added to the project. This new FTAB contains (n * (number of demand regions)) records, where n is the number of DCs in the optimized network.

Before returning to the TransportationLines.Build script, the FlowLine.Build script calls another script, FlowValues.Calculate (Appendix), and passes to it the newly created DC-to-Region Flow theme. This script adds new fields to the Flow FTAB to hold the total flow of each product type – Rx, CW, or OTC – as well as the total flow for all products combined. These flow volume data originally resided in the OptimizedValue field of the dirstore.dbf VTAB. However, the method for transferring these data to the FTAB is more complicated than establishing a one-to-one relationship between the Flow FTAB and the dirstore.dbf VTAB. This is because the VTAB contains three records for each demand region, one profiling the CW product flow, the second the OTC product flow, and the third the Rx product flow. Relating the tables based on origin and destination would yield only one-third of the data. To be sure each record of the Flow FTAB gets data on the flow volume of each product type, it is necessary to relate the tables three separate times, one for each type. This is accomplished by establishing a temporary field in both the Flow FTAB and the dirstore.dbf VTAB called ODP (that stands for Origin-Destination-Product), which holds a string resulting from the concatenation of the string values for the DC name, the demand region name, and the product type.

The FlowValues.Calculate script begins by getting the dirstore.dbf VTAB, creating the new ODP field, and calculating its value by concatenating values in the Facility, DemandRegion, and Product field. Next, it creates a similar ODP field in the Flow FTAB, as well as the fields to hold the individual product flow and total flow volumes. Then for each product type, the script calculates the value for the ODP field in the Flow FTAB by concatenating the DC field, the Store field, and a string naming the product type, such as "Rx." It relates the two tables by joining them based on the ODP fields (Figure 16). Thus, the Flow FTAB will be related one-to-one with the VTAB for the product type specified. The OptimizedValue field value is then populated in the product volume field of the Flow FTAB using the Calculate request, and the two virtual

DC-to-Region Flow Attribute Table

DC	*Store*	*CW Flow*	*Rx Flow*	*OTC Flow*	*Total Flow*	*ODP*
Knoxville	247		225846.23			Knoxville247Rx

Temporary Key Fields

Dirstore.dbf Table

Facility	*Demand Region*	*Product*	*Optimized Value*	*ActualRate*	*ODP*
Knoxville	247	Rx	225846.23	0.036351	Knoxville247Rx

Figure 16. Joining tables to capture flow line attributes. The dirstore.dbf table is joined to the DC-to-Region attribute table by a temporary origin-destination-product key field and the optimized flow value is copied over to the respective flow field in the attribute table.

tables are unjoined. When the flow volume values for each product type have been copied, the temporary ODP field in each virtual table is removed. Finally, for each record in the Flow FTAB, the flow volumes for each product type are summed and added to the Total Flow field in the Flow FTAB using the Calculate request on the Total Flow field. When the FlowValues.Calculate script is finished, control passes back to the FlowLine.Build script, which terminates by returning the DCs FTAB back to the TransportationLines.Build script.

Building the Transshipments Line Theme

The second script called by the TransportationLines.Build script is TranshipLine.Build (Appendix), which receives the DCs FTAB as an argument. This script is very similar to the FlowLine.Build script, in that it creates a new FTAB for the Transshipments line theme by looping through the DCs FTAB and getting the X-coord and Y-coord field values and using them as the beginning point for each line. The difference is that for each DC in the FTAB, the script then loops back through the DCs FTAB and uses each DC as an end point and adds a new record to the Transshipments FTAB, except in each case where the beginning point DC is the same as the end point DC. Thus, for an optimized network with n DCs, the new Transshipments FTAB will contain (n * (n –1)) records.

After the new Transshipments line theme is created and added to the project, TranshipLine.Build calls the script TransFlowValues.Calculate (Appendix), and passes the new Transshipments line theme as an argument. This script is identical to the FlowValues.Calculate script described above, except the tranship.dbf file is used instead of the dirstore.dbf file. The tranship.dbf file holds data about transhipment flows of Rx and CW product, but not OTC product. This is because the distribution of Rx and CW products is more cost effective when they are centrally warehoused, whereas the distribution of OTC products is cheaper when stocking points are decentralized and dispersed. As with the dirstore.dbf file, each origin and destination pair listed in the tranship.dbf file contains one record profiling the CW flow, and one profiling the Rx flow, and so the Transshipments FTAB and the tranship.dbf VTAB have to be related once for each product type using ODP fields in each table. When TransFlowValues.Calculate finishes, both that script and the TranshipLine.Build script return control to the TransportationLines.Build script, which then terminates.

Upon termination of this master script, the project now contains the base themes of the DCs, the DC-to-Region Flow, and the Transshipments. Several of the custom functions are also made available to the user. The DC-to-Region options under the DCs may be displayed by fixed cost or total demand using those respective items under Display Flows menu allow the user to display flows by product type or by total flow using line symbols whose thickness is graduated according to volume (Figures 17 and 5). Likewise, the Transhipment options under the Display Flows menu are enabled. The DCs may be displayed by fixed cost or total demand using those respective items under the Display DCs menu. However, the item that displays the DCs by handling cost is still disabled, as are all of the items under the Display Demand Regions menu. To enable these, the data on network flows need to be summarized for the DCs and for the demand regions.

Summarizing the DC Data

To get a better understanding of what the Summarize DC Data and Summarize Demand Region Data items do, it will be helpful to list the fields in the attribute tables of each of the base themes (Table 5).

Clicking on the Summarize DC Data item under the Logistics Model Setup launches a script CalcDCs (Appendix), which in turn calls several other scripts (Figure 18). CalcDCs first gets the FTAB of the DCs theme and passes it as an argument in a call to the script HasCWRx (Appendix). This script determines whether or not a DC is a centralized warehouse for Rx or CW products. After receiving the DCs FTAB, HasCWRx gives it two new fields, HasRx and HasCW, then it gets the VTAB of the picking.dbf table. It then loops through the DCs FTAB, and for each DC, it loops through the picking.dbf VTAB. If the name of the DC in the DCs FTAB matches the

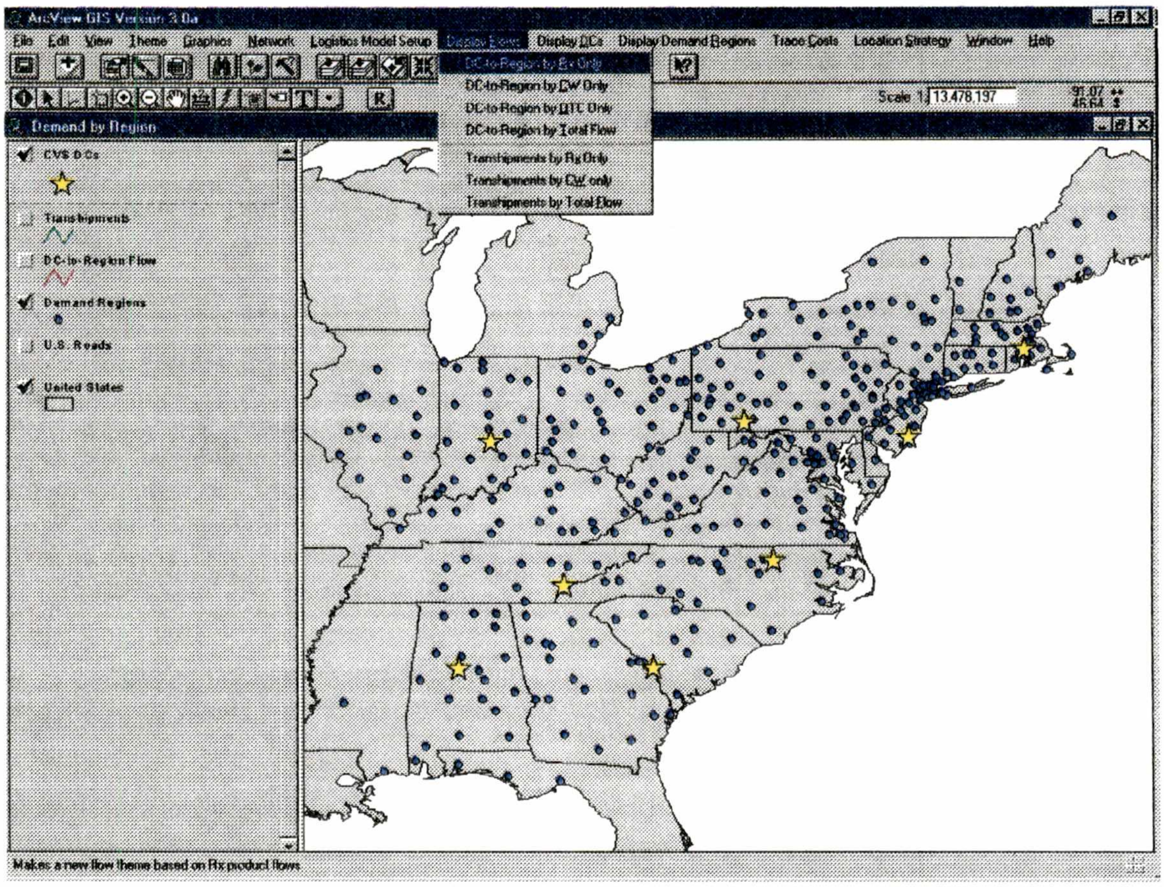


Figure 17. Display Flows drop-down menu after building the project's base themes.

Table 5. Fields in the attribute tables of the project's base themes.

	Themes			
	DCs	**DC-to-Region Flow**	**Transshipments**	**Demand Regions**
Fields	Facility FixedCost OptimizedValue	DC Store CW Flow Rx Flow OTC Flow Total Flow	Origin Destination CW Flow Rx Flow Total Flow	Demand Region

CalcDCs script launched by clicking on the Summarize DC Data menu item

1) Call **HasCWRx** script with DCs FTAB as an argument
   a) Add HasRx and HasCW field to DCs FTAB
   b) Get picking.dbf VTAB
   c) Check each DC in the DCs FTAB against the picking.dbf VTAB to determine if the DC stocks Rx or CW product, and record result in HasRx and HasCW fields
2) Call **SummTS** script
   a) Select records of Rx transshipments in tranship.dbf VTAB and sum the OptimizedValue field for each unique origin DC – creates table summarizing Rx picked for transshipment
   b) Select records of CW transshipments in tranship.dbf VTAB and sum the OptimizedValue field for each unique origin DC – creates table summarizing CW picked for transshipment
   c) Select records of Rx transshipments in tranship.dbf VTAB and sum the OptimizedValue field for each unique destination DC – creates table summarizing Rx crossdocked
   d) Select records of CW transshipments in tranship.dbf VTAB and sum the OptimizedValue field for each unique destination DC – creates table summarizing CW crossdocked
3) Call **SummD2S** script
   a) Get dirstore.dbf VTAB
   b) Select records of Rx shipments in dirstore.dbf VTAB and sum the OptimizedValue field for each unique DC – creates table summarizing Rx picked at servicing DC
   c) Select records of CW shipments in dirstore.dbf VTAB and sum the OptimizedValue field for each unique DC – creates table summarizing CW picked at servicing DC
   d) Select records of OTC shipments in dirstore.dbf VTAB and sum the OptimizedValue field for each unique DC – creates table summarizing OTC picked at servicing DC
4) Call **JoinSumms** script
   a) Joins summary tables to the DCs FTAB
   b) Sum joined fields to create Rx Picked, CW Picked, and OTC Picked fields
   c) Get the handling rates for each product type and each DC from the handling.dbf VTAB and copy to DCs FTAB
   d) Calculate cost for Rx, CW, and OTC picked, as well as for Rx and CW crossdocked and copy into new cost fields
   e) Sum cost fields to get Total Handling and store value in new field

Figure 18. Outline of scripts launched by Summarize DC Data menu item.

name of the DC in the picking.dbf VTAB, it then checks the value of the Process field in the latter. Initially, both the HasRx and the HasCW field in the DCs FTAB are set to zero. However, if the Process field in the picking.dbf VTAB contains the string "MakeRx", then the HasRx field in the DCs FTAB is set to one. Likewise, if the Process field contains "MakeCW", the HasCW field is set to one. Once finished, control is passed back to the CalcDCs script.

CalcDCs next calls the script SummTS (Appendix). The purpose of this script is to determine the volume of Rx and CW products that are picked at each DC stocking these products, and to determine the volume of Rx and CW products that are being crossdocked at those DCs that don't stock Rx and CW products, and therefore must have them transshipped from other DCs. This information is important to know because transshipped products incur extra shipping and handling costs.

SummTS begins by selecting all records in the tranship.dbf VTAB whose Product field contains the "Rx" string. Once a subset of records has been selected, the script sums the OptimizedValue field for each unique DC in the OriginFacility field by making the Summarize request on the tranship.dbf VTAB. This request creates a new VTAB whose file is RxPicked.dbf. The same summary is done for records whose product is "CW", which creates the file CWPicked.dbf. The summed OptimizedValue fields in these new VTABs are changed to "RxPicked for TS" and "CWPicked for TS" respectively. This process is then repeated for each product type, but this time over each unique DC in the DestinationFacility. This produces two new files called Rx_X_Doc.dbf and CW_X_Doc.dbf, which hold the data about which DCs are crossdocking transshipments and what are the transshipment volumes. The summed OptimizedValue fields in these

VTABs are changed to Rx_X_Doc and CW_X_Doc respectively. Control is then passed back to CalcDCs.

CalcDCs then calls the script SummD2S (Appendix). This script summarizes the flows of product from the DCs to the demand regions for each unique DC, thus providing information about the volume of product flow that originates at the servicing DC (i.e., flow that does not involve a transshipment). The data for these flows are held in the dirstore.dbf table. The structure and logic of this script is similar to the SummTS script. Product-specific records in the dirstore.dbf VTAB are selected by issuing a query to the VTAB of records according to the Product field value, and these selected records are summed over the Facility field holding the DC name. After performing this procedure once for each product type, the script returns control to CalcDCs having created three new files, RxDirect.dbf, CWDirect.dbf, and OTCDirect.dbf. The OptimizedValue fields in these summary VTABs are changed to Rx D2S, CW D2S, and OTC D2S.

The last script called by CalcDCs is JoinSumms (Appendix). This script joins the RxPicked.dbf, CWPicked.dbf, Rx_X_Doc.dbf, CW_X_Doc.dbf, RxDirect.dbf, CWDirect.dbf, and OTCDirect.dbf VTABs to the DCs FTAB based on the Facility fields in the joined VTABs and in the DCs FTAB. Essentially, what these joined tables give to the DCs VTAB are seven new fields: RxPicked for TS, CWPicked for TS, Rx_X_Doc, CW_X_Doc, Rx D2S, CW D2S, and OTC D2S. Before terminating, the script sums the volume picked for each product type and places those summed values in three new fields, Rx Picked, CW Picked, and OTC Picked. The script then calculates the cost associated with the volumes provided in these fields, as well as the Rx_X_Doc and the CW_X_Doc fields, and adds these values to respective cost fields that are also created by the script.

To calculate these costs, the script gets the handling.dbf VTAB and transfers the handling rate for each product over to new Rx, CW, and OTC rate fields in the DCs FTAB. As with the dirstore.dbf and tranship.dbf tables, this table has three records for each DC, one holding the CW handling rate, the second holding the OTC handling rate, and the third the Rx handling rate. To transfer these rates to a single record in the DCs FTAB, it is necessary to loop through that FTAB, and for each DC, to loop through the handling.dbf VTAB and copy the handling rates for each of the three records whose Facility field name matches with the DC name in the FTAB. Once this is accomplished, the script creates six new cost fields in the FTAB: Rx Pick Cost, Rx X Doc Cost, CW Pick Cost, CW X Doc Cost, OTC Pick Cost, and, finally, Total Handling. The Rx Pick Cost and Rx X Doc cost are calculated by multiplying the values in their respective volume fields by the value in the Rx Rate field. The CW Pick Cost and CW X Doc Cost fields are likewise calculated using the CW Rate field value. Calculating the OTC Pick Cost is similar to the previous calculations, except that since it is not necessary to transship OTC products, there is no OTC crossdock cost to calculate. The Total Handling field is calculated last by summing all of the cost fields described above. At this point the summary of the DC data is complete, and the By Total Handling item under the Display DCs menu is enabled.

Summarizing the Demand Regions Data

The last item under the Logistics Model Setup view menu is the Summarize Demand Region Data item. Clicking on this item launches the script SummDems (Appendix). This script is somewhat similar to the CalcDCs script, except that it summarized the network flow data over each unique demand region, rather than each unique DC.

SummDems first gets the VTAB of the dirstore.dbf table. Then for each product type, it queries the VTAB using the product type as the selection criteria. For example, it first selects all records in the dirstore.dbf VTAB whose Product field contains "Rx". The script then summarizes this selected set over each unique demand region, yielding a new VTAB holding the total Rx volume going to each demand region. The file for this VTAB is called Rx2Store.dbf. In the same fashion, the CW2Store.dbf and the OTC2Store.dbf files are created. These VTABs are then joined to the Demand Regions FTAB using the demand region name as the common key. After the join, the Demand Regions FTAB has three new joined fields titled after the summary tables that were joined. To prevent null values for some of the demand regions' product flows from disrupting the calculations, it is necessary to create three new fields in the FTAB to which is transferred the data in the three joined fields. These three fields are called Rx_Vol, CW_Vol, and OTC_Vol. First the values for these fields are set to zero, then all non-null values in the Rx2Store, CW2Store, and OTC2Store fields are copied over. After copying the data, the joined fields are no longer necessary so they are unjoined from the Demand Regions VTAB. Before terminating, the script creates one more field called Total Demand, which holds the sum for the values in the three volume fields.

Displaying the Network Features

Once the summary of the DC and demand regions data is complete, the user will notice that the attribute tables for the DCs and Demand Regions themes have several new fields (Table 6). At this point, several more menu items under the Display Demand Regions are also enabled (Figure 19), in addition to the By Handling Cost item under the Display DCs menu. The only display items that remain disabled are the items to display logistics costs for each demand region by product type and by all products. These items require the application of the chain-wide-by-product, and chain-wide-by-all-products items under the Trace Costs menu. These items will be discussed in the next chapter.

Like the fixed cost and total demand display options under the Display DC menu, the handling cost option clones the DCs theme and changes the new theme's legend so that the DC symbols are graduated in size based on a natural break classification of the Total Handling field in the DCs FTAB. The scripts run by the By Fixed Cost and By Total Demand items operate in a similar fashion, sizing the DC symbols of the cloned DCs by the FixedCost field and the OptimizedValue field respectively.

Likewise, the Demand Regions theme may be altered according to the Total Demand field in the Demand Regions FTAB. The By Product Volume option under the Display Demand Regions menu, on the other hand, creates a pie chart symbol for each demand region (Figure 20). With these pies, the individual product volumes are represented as pie slices, while the total demand volume is represented by the size of the pie.

The scripts launched by the display options for the DC-to-Region and Transshipments flows utilizes an interesting graphical manipulation to draw the spider diagrams which result from a network optimization. As previously noted, the DC-to-Region Flow and Transshipments themes are the base themes for the project, and they

Table 6. Fields in the base theme attribute tables after summarizing DC and Demand Regions data.

	Themes			
	DCs	**DC-to-Region Flow**	**Transshipments**	**Demand Regions**
Fields	Facility	DC	Origin	Demand Region
	FixedCost	Store	Destination	OTC_Vol
	OptimizedValue	CW Flow	CW Flow	Rx_Vol
	Rx Picked for TS	Rx Flow	Rx Flow	CW_Vol
	Rx_X_Doc	OTC Flow	Total Flow	Total Demand
	Rx D2S	Total Flow		
	CW Picked for TS			
	CW_X_Doc			
	CW D2S			
	OTC Picked			
	Rx Picked			
	CW Picked			
	Rx Rate			
	CW Rate			
	OTC Rate			
	Rx X Doc Cost			
	Rx Pick Cost			
	CW X Doc Cost			
	CW Pick Cost			
	OTC Pick Cost			
	Total Handling			

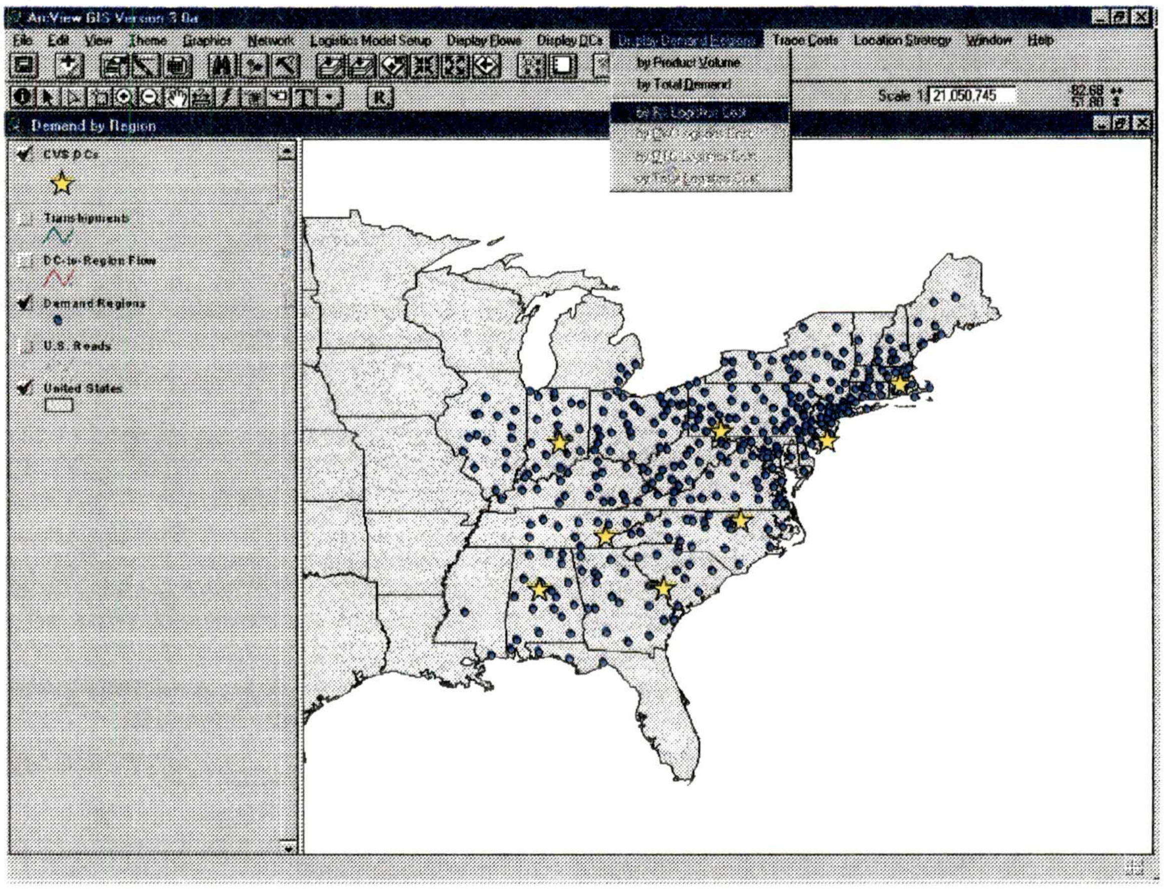


Figure 19. Display Demand Regions drop-down menu after summarizing demand region data.

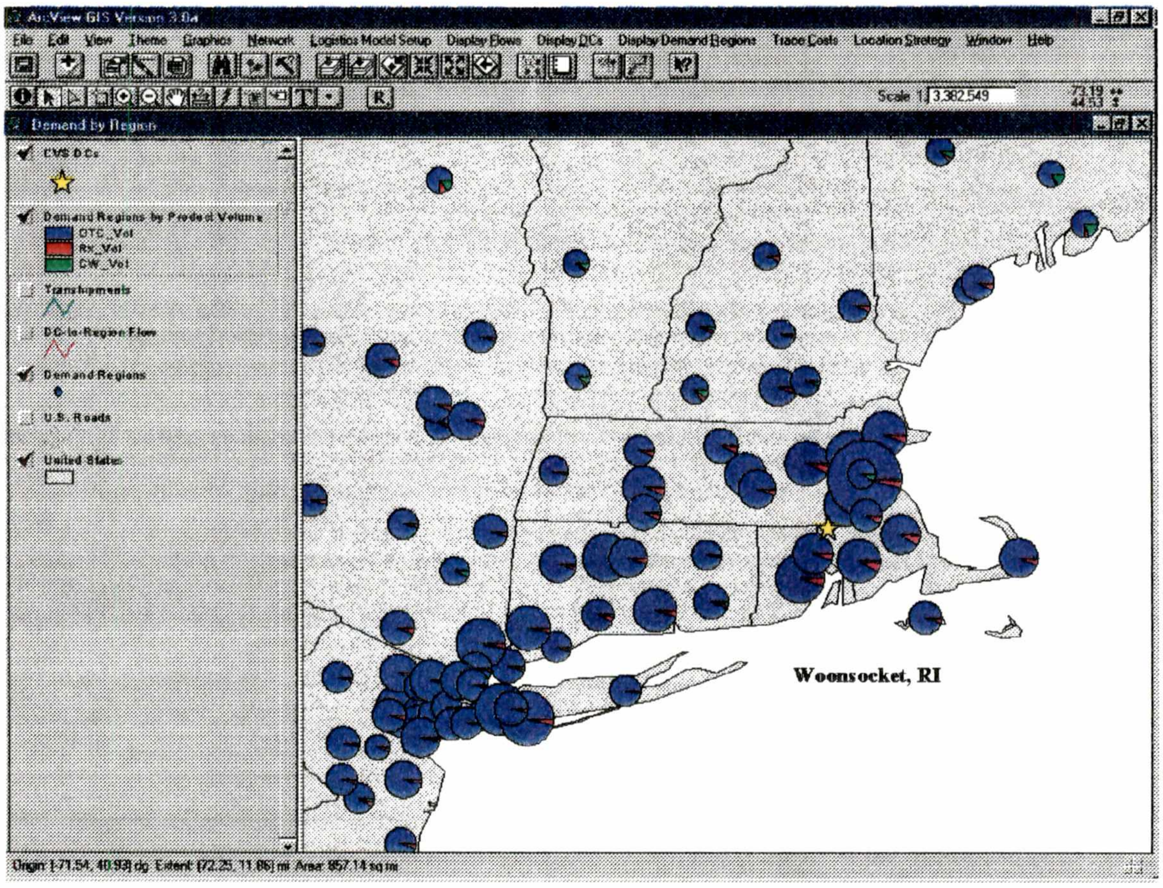

ArcView GIS Version 3.0a
File Edit View Theme Graphics Network Logistics Model Setup Display Flows Display DCs Display Demand Regions Trace Costs Location Strategy Window Help
Scale 1: 3,382,549
Demand by Region
CVS DCs
Demand Regions by Product Volume
OTC_Vol
Rx_Vol
CW_Vol
Transhipments
DC-to-Region Flow
Demand Regions
U.S. Roads
United States
Woonsocket, RI
Origin: [-71.54, 40.93] dg Extent: [72.25, 11.86] mi Area: 857.14 sq mi

Figure 20. Pie chart symbols representing demand regions. Pie slices represent product volume, while the size of the pie represents total volume.

contain all of the possible origin-destination pairs in the optimized network. Once the optimized flow values are added to the FTAB of these themes, it would be possible to generate a new theme by selecting out the positive flows and building a new FTAB with these records. However, this method has the disadvantage of creating a new file for each theme, which takes up space in memory.

When classifying a theme according to a field in the FTAB, ArcView provides the capability of specifying a null value in that field, and displaying records having that value with a special null symbol. For example, if the field contains records whose values are empty, or are flagged with a common null value, such as –9999, these records may be displayed with a certain symbol or with a certain color. This functionality in ArcView comes in very handy when displaying the network flows. Because we are only interested in seeing the flows with positive values, we simply make all the zero value flows invisible by setting the null value to zero, and setting the null symbol to a transparent color (Figure 21). Thus, it is possible to display the various DC-to-Region and Transhipment flows listed under the Display Flows menu by cloning the base theme and simply altering the legend, and at the same time avoiding the creation of a new file in memory. See the TotalFlowTheme.Make and the TransTotalFlowTheme.Make scripts in Appendix as examples of how this is accomplished for each flow display option.

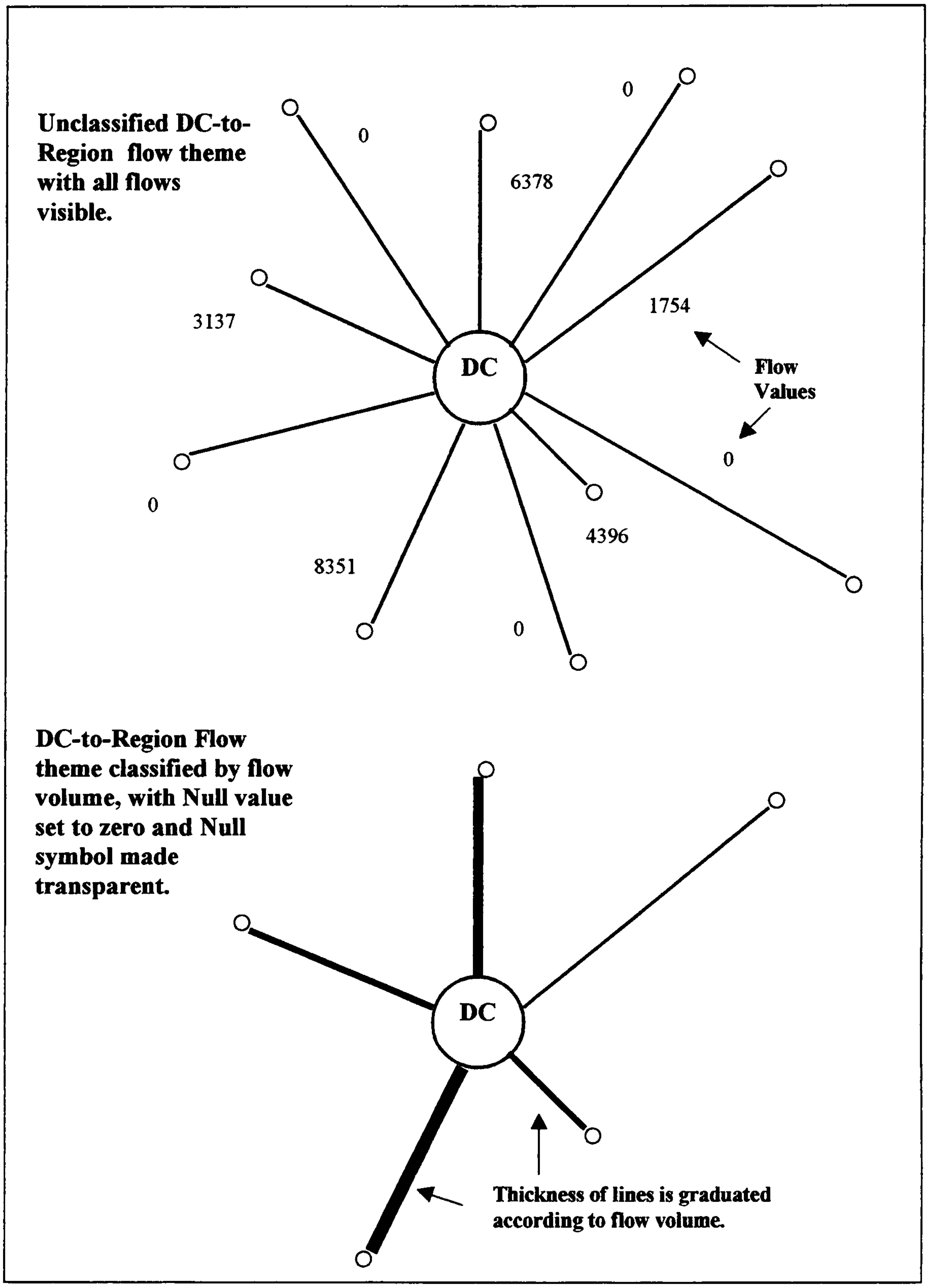

Unclassified DC-to-Region flow theme with all flows visible.
0
6378
0
1754
Flow Values
3137
DC
0
0
4396
8351
0
DC-to-Region Flow theme classified by flow volume, with Null value set to zero and Null symbol made transparent.
DC
Thickness of lines is graduated according to flow volume.

Figure 21. Making zero value flows to demand regions invisible using null symbol.

Chapter 4

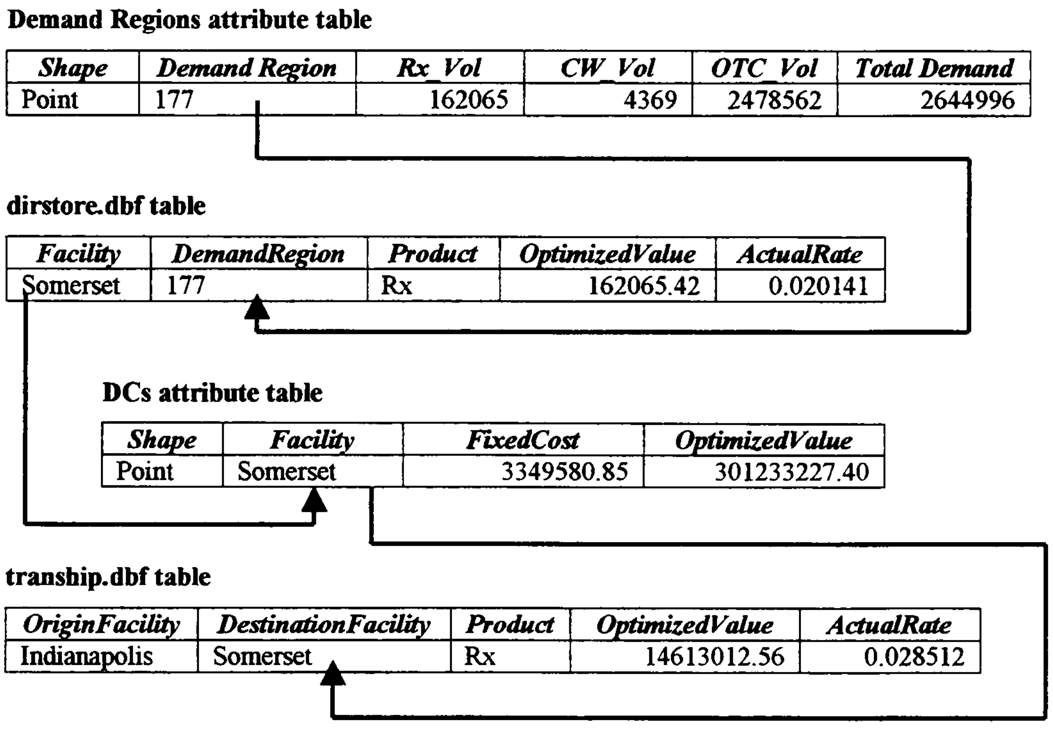
Tracing Demand Region Logistics Costs

Tracing Logistics Costs of Individual Demand Regions

As was discussed in Chapter 2, the individual components of a logistics network, such as the lines of product flow, or the origin and destination facilities of those flows, are well suited for graphical manipulation and display. Tracing logistics costs from the ultimate destination back up through the network to the ultimate origin, on the other hand, involves summing the costs of different components of the network and, consequently, associating features of different themes. Such trans-thematic events cannot be rendered easily with graphical displays, yet ArcView does offer a way to relate the data from each theme's attribute table so that logistics costs can be analyzed.

ArcView uses relational databases. This means that two tables having fields with common data types can be related to each other. This common field is often called the key field. In the last chapter, one type of relation called a join was reviewed. Recall that a selected set of the dirstore.dbf table was taken and joined to the DC-to-Region Flow table based on the ODP key field. Such tabular joins merge the two tables in the project interface while keeping separate their files in memory. Another type of relation ArcView provides is called a link. Unlike a join, two linked tables do not actually merge. Rather, the link establishes a relationship between the two tables in which the selected records of the *linking* table will automatically select one or more of those records in the *linked* table having an identical value in the key field. Moreover, unlike with the join, ArcView allows tables to be linked in a chain-like fashion, so that a table can be both the *linked* and the *linking* table.

For the network optimization project, we make use of this linking capability to trace the logistics costs from the demand chain up to the origin facilities. The basic logic for all of the trace cost scripts is to link the demand regions to the shipment table using the demand region name as the common data element, then to link the shipment table to the DCs table, and the DCs table to the transshipment table, using the DC name as common data element (Figure 22). The most complex of these traces occurs with Demand Region Rx Only and Demand Region CW Only items, since these products often involve extra crossdocking and transshipment costs, so we will use the script for Rx cost tracing, RxTrace (Appendix) as an example here (see Figure 23 for an outline).



Demand Regions attribute table

Shape	*Demand Region*	*Rx_Vol*	*CW_Vol*	*OTC_Vol*	*Total Demand*
Point	177	162065	4369	2478562	2644996

dirstore.dbf table

Facility	*DemandRegion*	*Product*	*OptimizedValue*	*ActualRate*
Somerset	177	Rx	162065.42	0.020141

DCs attribute table

Shape	*Facility*	*FixedCost*	*OptimizedValue*
Point	Somerset	3349580.85	301233227.40

tranship.dbf table

OriginFacility	*DestinationFacility*	*Product*	*OptimizedValue*	*ActualRate*
Indianapolis	Somerset	Rx	14613012.56	0.028512

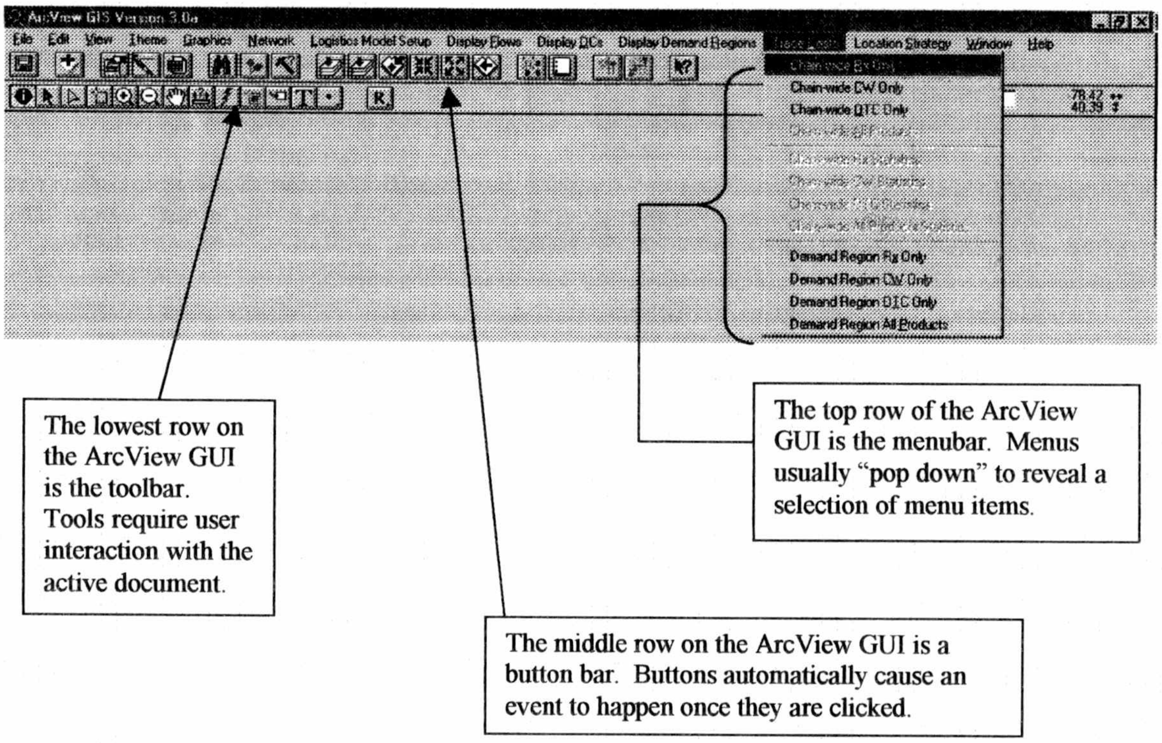
Figure 22. Linking tables to trace logistics costs.

```
RxTrace launched by clicking on Demand Regions Rx Only menu item or "R" tool
1) Call Tables.Link script
    a) Link the Demand Regions attribute table to the dirstore.dbf table by the demand region
       name and link the dirstore.dbf table to the DCs attribute table, and the DCs attribute table
       to the tranship.dbf table, by the DC name
2) Get the VTABs for the above tables
3) Call the system script View.SelectPoint
    a) If the mouse is clicked on one or more features in the Demand Regions theme, select the
       records for those demand regions in the Demand Regions attribute table
    b) If the mouse is clicked anything that is not a demand region, then exit
4) If demand regions were selected, make a list of Boolean values indicating whether or not the
   demand region was selected
5) For each true value in the list
    a) Select the demand region in the Demand Regions FTAB
    b) Reselect the selection in the dirstore.dbf VTAB that results from the link with the
       Demand Regions FTAB, so that Rx product flow from servicing DC is selected
    c) Calculate the shipping cost by multiplying the Rx flow volume by the shipping rate
    d) Get the selected DC in the DCs FTAB and check to see if it stocks Rx product
        i) If it does, multiply the Rx handling rate for that DC by the flow volume in the
           dirstore.dbf VTAB to get the picking cost for that demand region
        ii) If it does not, do the same multiplication, but this is crossdocking cost
            (1) Reselect from the selected set in the tranship.dbf VTAB that results from the
                link with the DCs FTAB so that Rx product flow to servicing DC is selected
                (a) If there is only one transshipment record selected, multiply the flow volume
                    in the dirstore.dbf VTAB by the transship rate in the tranship.dbf VTAB to
                    get the transshipment cost
                (b) Else there is more than one origin DC transshipping to servicing DC, so
                    multiply flow volume in the dirstore.dbf VTAB by the weighted average
                    transship rate of the multiple origin DCs to get the transshipment cost
    e) Sum the shipping, crossdocking, transhipment, and picking costs to get the total logistics
       cost for Rx products for the demand region
    f) Issue the report message box to the screen with itemized and total logistics costs listed
6) Call Tables.Unlink to unlink the tables.
```

Figure 23. Outline of scripts launched by Demand Regions Rx Only menu item or "R" tool.

RxTrace is the script for a tool. Unlike a GUI button or menu item, which immediately causes a script to run, a tool waits to receive input from the user that will be needed to run the script (Figure 24). Before the RxTrace script receives the user input, it first calls a script Tables.Link (Appendix), which links the Demand Regions attribute table to the dirstore.dbf table, the dirstore.dbf table to the DCs attribute table, and the DCs attribute table to the tranship.dbf table. Control then returns to RxTrace, which gets the VTABs of the dirstore.dbf and tranship.dbf tables, and the FTABs for the Demand Regions and DCs themes. Once RxTrace has retrieved these linked virtual tables, the script calls the ArcView system script View.SelectPoint. This script makes the mouse cursor a selection cursor. If, when the mouse is clicked, the cursor is positioned over a feature from the active theme, then that feature's record in the FTAB is selected. Since RxTrace makes the Demand Regions theme active, the script will proceed if the feature clicked is a demand region point in the view. If the mouse is clicked on any other part of the screen other than a Demand Regions point feature, the script terminates. If the mouse click event takes place over more than one point in the Demand Regions theme, then the logistics costs for each selected demand region are calculated. The script keeps track of the number of selected points by converting the entire set of records in the Demand Regions FTAB into a list of Boolean values where 1 means the record was selected, and 0 means the record was not selected.

For each list element equaling one, the RxTrace script selects that record in the Demand Regions FTAB. Doing this automatically selects the records in the dirstore.dbf VTAB whose demand region name matches the demand region name of the selected record. This in turn selects matching records in the other linked tables (Figure 25).

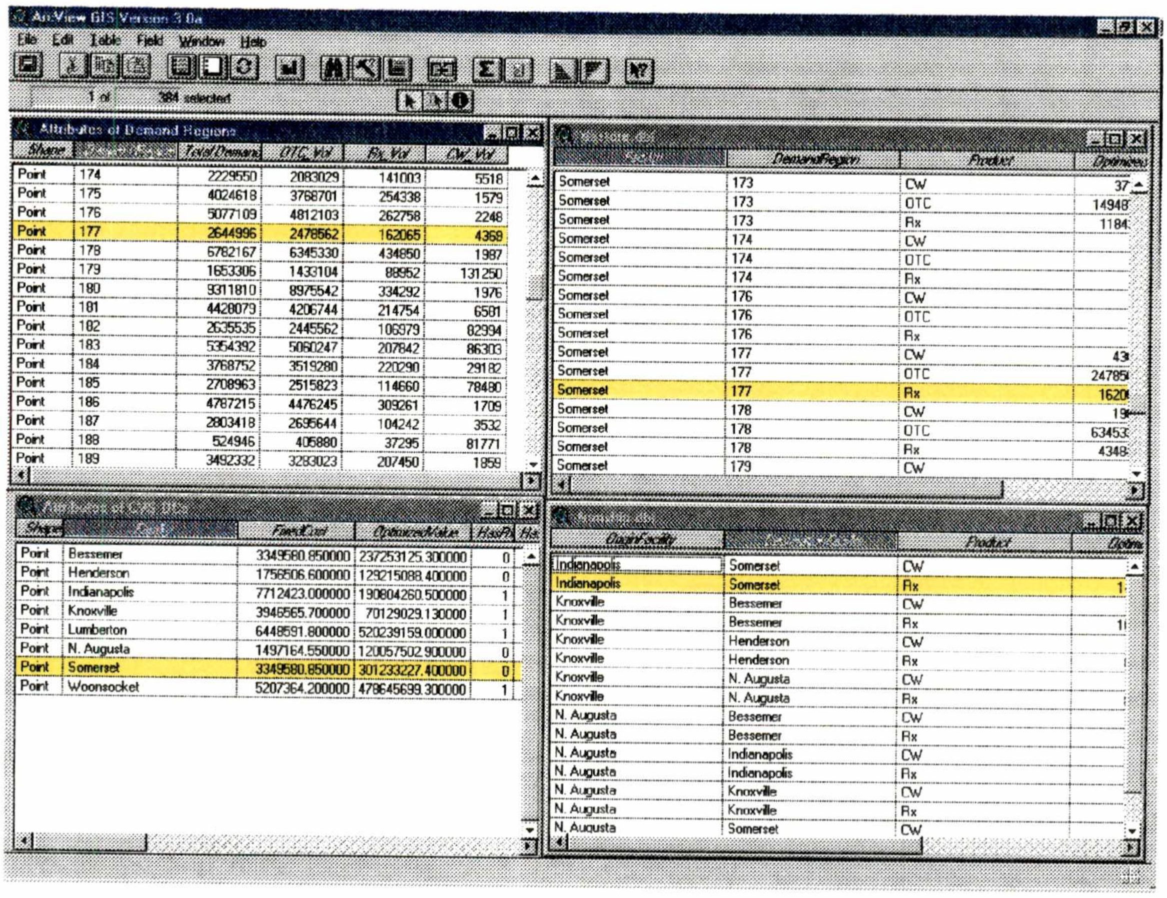

ArcView GIS Version 3.0a
File Edit View Theme Graphics Network Logistics Model Setup Display Flows Display DCs Display Demand Regions Location Strategy Window Help
Chain-wide CW Only
Chain-wide DTC Only
Demand Region Rx Only
Demand Region CW Only
Demand Region DTC Only
Demand Region All Products
78.42
40.39

The lowest row on the ArcView GUI is the toolbar. Tools require user interaction with the active document.

The top row of the ArcView GUI is the menubar. Menus usually "pop down" to reveal a selection of menu items.

The middle row on the ArcView GUI is a button bar. Buttons automatically cause an event to happen once they are clicked.

Figure 24. ArcView GUI for a View document.


ArcView GIS Version 3.0a
File Edit Table Field Window Help
1 of 384 selected
Attributes of Demand Regions

Shape	[illegible]	TotalDemand	OTC_Vol	Rx_Vol	CW_Vol
Point	174	2229550	2083029	141003	5518
Point	175	4024618	3768701	254338	1579
Point	176	5077109	4812103	262758	2248
Point	177	2644996	2478562	162065	4369
Point	178	6782167	6345330	434850	1987
Point	179	1653306	1433104	88952	131250
Point	180	9311810	8975542	334292	1976
Point	181	4428079	4206744	214754	6581
Point	182	2635535	2445562	106979	82994
Point	183	5354392	5060247	207842	86303
Point	184	3768752	3519280	220290	29182
Point	185	2708963	2515823	114660	78480
Point	186	4787215	4476245	309261	1709
Point	187	2803418	2695644	104242	3532
Point	188	524946	405880	37295	81771
Point	189	3492332	3283023	207450	1859

[illegible]	DemandRegion	Product	Optimized
Somerset	173	CW	37
Somerset	173	OTC	14948
Somerset	173	Rx	1184
Somerset	174	CW	
Somerset	174	OTC	
Somerset	174	Rx	
Somerset	176	CW	
Somerset	176	OTC	
Somerset	176	Rx	
Somerset	177	CW	43
Somerset	177	OTC	24785
Somerset	177	Rx	1620
Somerset	178	CW	19
Somerset	178	OTC	63453
Somerset	178	Rx	4348
Somerset	179	CW	

Shape	[illegible]	FixedCost	OptimizedValue	HasRx	Ha
Point	Bessemer	3349580.850000	237253125.300000	0	
Point	Henderson	1756506.600000	129215088.400000	0	
Point	Indianapolis	7712423.000000	190804260.500000	1	
Point	Knoxville	3946565.700000	70129029.130000	1	
Point	Lumberton	6448591.800000	520239159.000000	1	
Point	N. Augusta	1497164.550000	120057502.900000	0	
Point	Somerset	3349580.850000	301233227.400000	0	
Point	Woonsocket	5207364.200000	478645699.300000	1	

OriginFacility	[illegible]	Product	Optim
Indianapolis	Somerset	CW	
Indianapolis	Somerset	Rx	1
Knoxville	Bessemer	CW	
Knoxville	Bessemer	Rx	1
Knoxville	Henderson	CW	
Knoxville	Henderson	Rx	
Knoxville	N. Augusta	CW	
Knoxville	N. Augusta	Rx	
N. Augusta	Bessemer	CW	
N. Augusta	Bessemer	Rx	
N. Augusta	Indianapolis	CW	
N. Augusta	Indianapolis	Rx	
N. Augusta	Knoxville	CW	
N. Augusta	Knoxville	Rx	
N. Augusta	Somerset	CW	

Figure 25. Example of linked tables after selecting a demand region to trace logistics costs. The Demand Regions attribute table (top left) is linked to the dirstore.dbf table (top right) by the demand region name field. The dirstore.dbf table is linked to DCs attribute table (bottom left) by the DC name in their Facility fields. The DCs attribute table is linked to the tranship.dbf table by the DC name in the tranship.dbf table's DestinationFacility field.

Since each DC in the dirstore.dbf VTAB contains three records for each demand region representing the flow of the three different product types, multiple records are selected in the dirstore.dbf. To distinguish the servicing DC from the others, and to distinguish the Rx product flows from the CW and OTC product flows, the selected set in the dirstore.dbf VTAB is reselected for all records whose Product field value is "Rx", and whose OptimizedValue field value is greater than zero. This reselection will isolate the Rx flow coming from the DC assigned to that demand region, which means there will be only one selected record (the script assumes that each demand region will be assigned only one DC). It then gets the shipping rate from the ActualRate field, the flow volume from the OptimizedValue field, and the DC name from the Facility field, for that record. Finally, it calculates the shipping cost by multiplying the flow volume by the shipping rate.

To illustrate how this process of tracing logistics costs works, let us look at demand region 229 near Charlottesville, Virginia as an example (Figure 26). Demand Region 229 is allocated to the Henderson, North Carolina DC, as evident from the Rx Flow lines in the figure. It will also be noted from the figure that the Henderson DC does not stock Rx products and so must be transshipped from the Knoxville, Tennessee, DC. When this demand region is clicked with the mouse, its record in the Demand Regions table is selected. Because the Demand Regions VTAB is linked to the dirstore.dbf VTAB, all of the records in that table having "229" in the DemandRegion field are selected. Keep in mind that the dirstore.dbf table contains three records for all possible shipments between all DCs and all demand regions, one record for each product type. Since there are eight DCs in this sample optimized network, selecting demand region

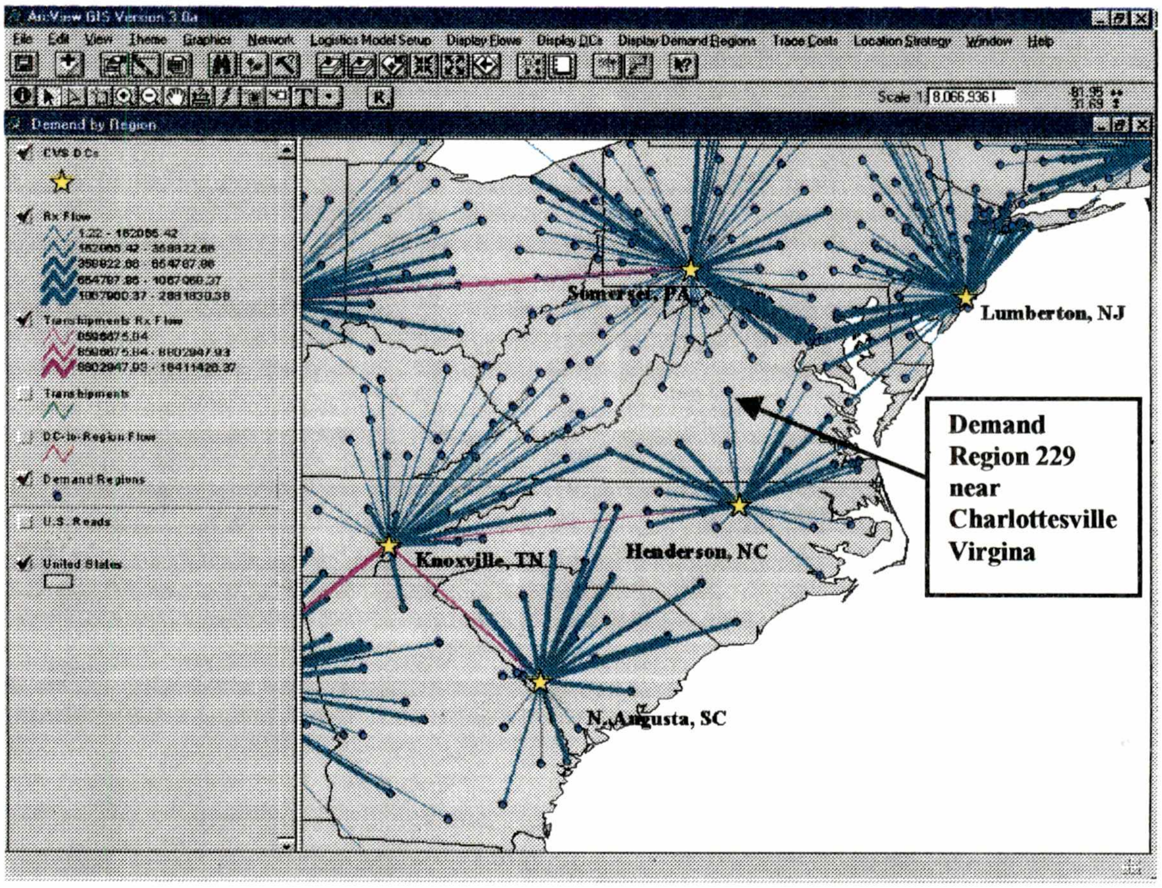


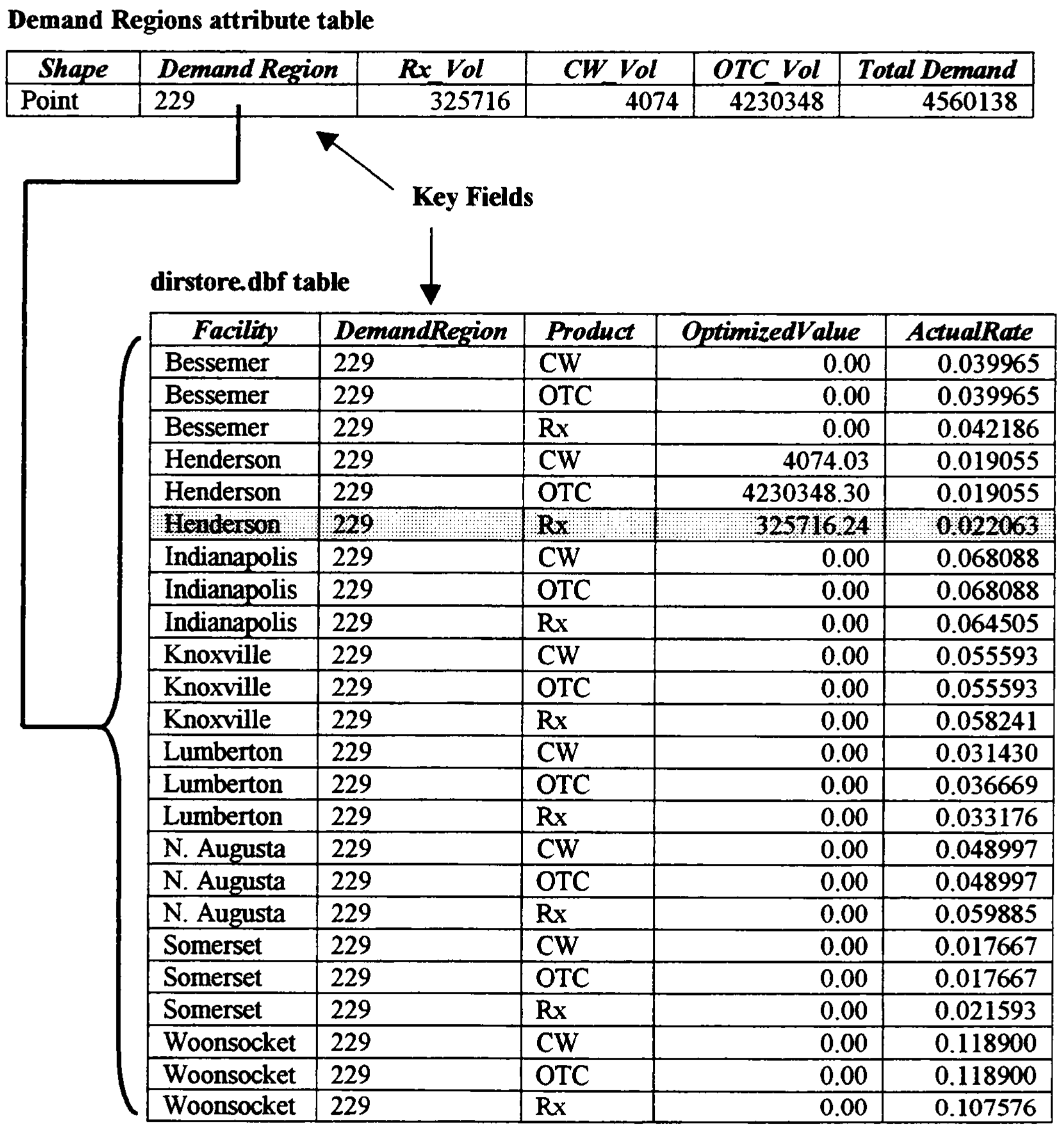
Figure 26. Demand region 229 near Charlottesville, Virginia in the Rx product network. Demand Region 229 is allocated to the Henderson, North Carolina (Rx flow in green). The Henderson DC does not stock Rx products, and receives Rx transshipments from the Knoxville, Tennessee, DC (Rx transshipments in purple).

229 will cause 24 records in the dirstore.dbf table to be selected (Figure 27). Notice that most of the records in Figure 27 have an OptimizedValue field value (i.e., a flow value) of zero. In fact, only the Henderson facility has positive flow values. To ensure that only the record for the Henderson DC serving Rx product to demand region 229 is selected, TraceRx then reselects all records from the selected set that have an OptimizedValue field value greater than zero *and* that have a Product field value of "Rx". Doing this operation selects the targeted record as indicated by the shaded record in the figure.

Once the proper record has been isolated, TraceRx gets the value in OptimizedValue field and multiplies it by the value in the ActualRate field to get the shipping cost. Thus, the shipping cost for demand region 229 is

$$\mathbf{325{,}716.24\ units \times \$0.022063\ per\ unit = \$7{,}186.28.}$$

At this point, the script needs to determine if the Rx flow coming from this DC is stocked by this DC or is transshipped to this DC from another DC. It does this by looking to the selected DC in the DCs FTAB, which was linked to the dirstore.dbf VTAB by the Facility fields in both of those tables. The script then gets the selected DC's values for the Rx Rate and the HasRx fields. If the HasRx value is 1, this DC is an Rx product stocking point, and the script need only calculate the picking cost at this DC, which is done by multiplying the flow volume by the Rx Rate field value. If it is zero, the DC is not an Rx product stocking point, so the script must determine the crossdocking cost at this DC, the transshipment cost, and the picking cost at the origin DC.



Demand Regions attribute table

Shape	*Demand Region*	*Rx_Vol*	*CW_Vol*	*OTC_Vol*	*Total Demand*
Point	229	325716	4074	4230348	4560138

Key Fields

dirstore.dbf table

Facility	*DemandRegion*	*Product*	*OptimizedValue*	*ActualRate*
Bessemer	229	CW	0.00	0.039965
Bessemer	229	OTC	0.00	0.039965
Bessemer	229	Rx	0.00	0.042186
Henderson	229	CW	4074.03	0.019055
Henderson	229	OTC	4230348.30	0.019055
Henderson	229	Rx	325716.24	0.022063
Indianapolis	229	CW	0.00	0.068088
Indianapolis	229	OTC	0.00	0.068088
Indianapolis	229	Rx	0.00	0.064505
Knoxville	229	CW	0.00	0.055593
Knoxville	229	OTC	0.00	0.055593
Knoxville	229	Rx	0.00	0.058241
Lumberton	229	CW	0.00	0.031430
Lumberton	229	OTC	0.00	0.036669
Lumberton	229	Rx	0.00	0.033176
N. Augusta	229	CW	0.00	0.048997
N. Augusta	229	OTC	0.00	0.048997
N. Augusta	229	Rx	0.00	0.059885
Somerset	229	CW	0.00	0.017667
Somerset	229	OTC	0.00	0.017667
Somerset	229	Rx	0.00	0.021593
Woonsocket	229	CW	0.00	0.118900
Woonsocket	229	OTC	0.00	0.118900
Woonsocket	229	Rx	0.00	0.107576

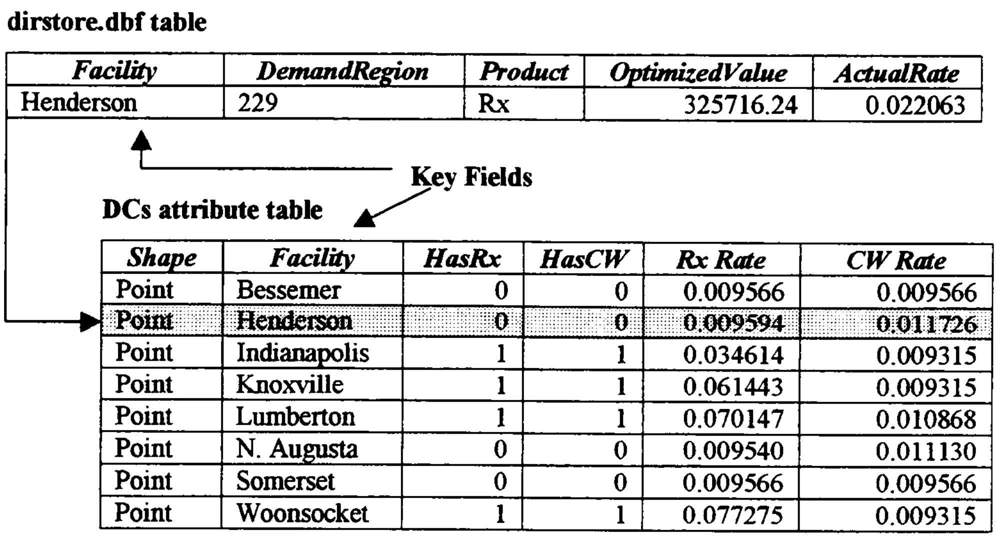
Figure 27. Selected records in dirstore.dbf table after clicking on demand region 229. The desired record (shaded) must be reselected from this set by querying for OptimizedValue field values greater than zero *and* Product field values equal to "Rx".

The crossdocking cost is easily calculated, since it is simply the flow volume in the dirstore.dbf VTAB multiplied by the Rx rate in the DCs FTAB. When the same Rx handling rate applies to both picking and crossdocking, calculating the crossdocking cost and the picking cost are identical.

Applying this to our example of demand region 229, note that the selected record in the dirstore.dbf table has caused the record for the Henderson DC in the DCs attribute table to be selected (Figure 28). Had the value in the HasRx field been one, then the script would have calculated the picking cost using the value in the Rx Rate field for that record, so that

$$\mathbf{325{,}716.24\ units \times \$0.009594\ per\ unit = \$3{,}124.92}$$

would have been the picking cost for the demand region. However, the value in the



dirstore.dbf table

Facility	*DemandRegion*	*Product*	*OptimizedValue*	*ActualRate*
Henderson	229	Rx	325716.24	0.022063

Key Fields

DCs attribute table

Shape	*Facility*	*HasRx*	*HasCW*	*Rx Rate*	*CW Rate*
Point	Bessemer	0	0	0.009566	0.009566
Point	Henderson	0	0	0.009594	0.011726
Point	Indianapolis	1	1	0.034614	0.009315
Point	Knoxville	1	1	0.061443	0.009315
Point	Lumberton	1	1	0.070147	0.010868
Point	N. Augusta	0	0	0.009540	0.011130
Point	Somerset	0	0	0.009566	0.009566
Point	Woonsocket	1	1	0.077275	0.009315

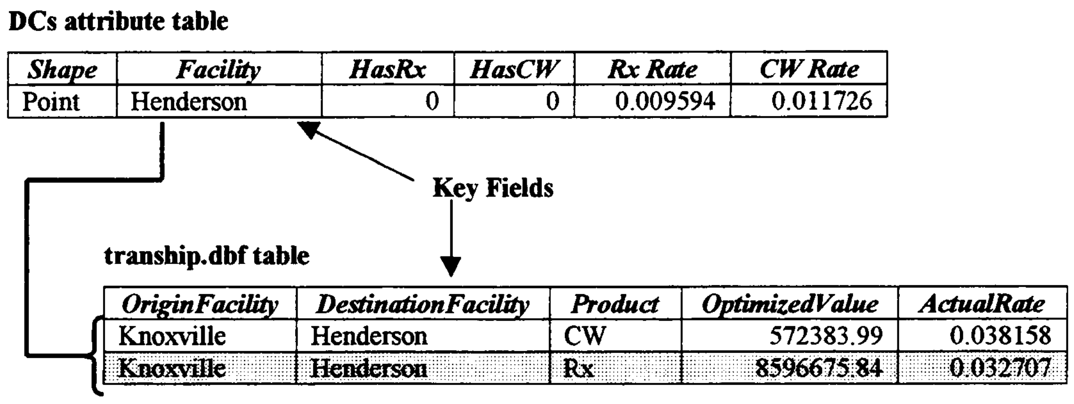
Figure 28. The dirstore.dbf table linked to the DCs attribute table by the Facility fields. After reselecting the Rx product flow from Henderson to demand region 229, the record for the Henderson DC in the DCs attribute table is automatically selected (shaded).

HasRx field is zero, which means the Henderson DC does not stock Rx products, and therefore the above dollar figure becomes the crossdocking cost for the demand region.

The script then looks to the selected set in the tranship.dbf VTAB, which was linked to the DCs by its DestinationFacility field, in order to determine from which DC the transshipments are coming. The tranship.dbf table is similar to the dirstore.dbf table, in that each DC-to-DC transshipment pair listed in the table contains more than one record, one for Rx transshipments, and the other for CW transshipments. However, unlike the dirstore.dbf table, the transship.dbf table does not necessarily list all possible transshipment pairs. Nevertheless, to be sure the proper record is obtained, it is necessary to reselect from this selected set only those transshipments whose OptimizedValue field value is greater than zero, and whose Product field value is "Rx". Once the desired transshipment record is isolated, the transshipment cost is calculated by multiplying the flow volume previously obtained from the dirstore.dbf VTAB by the transshipment rate, which is held in the ActualRate field of the transhipment.dbf VTAB.

Getting back to our example, the selected record for the Henderson DC in the DCs attribute table automatically selects the records from the tranship.dbf table whose DestinationFacility field values match with Henderson (Figure 29). Reselecting for Product field values equal to "Rx" and OptimizedValue field values greater than zero, the desired field (shaded) becomes selected. Using the ActualRate field value, the transshipment cost associate with Rx products going from the Henderson DC to demand region 229 becomes

$$\textbf{325,716.24 units} \times \textbf{\$0.032707 per unit} = \textbf{\$10,653.20.}$$



DCs attribute table

Shape	*Facility*	*HasRx*	*HasCW*	*Rx Rate*	*CW Rate*
Point	Henderson	0	0	0.009594	0.011726

Key Fields

tranship.dbf table

OriginFacility	*DestinationFacility*	*Product*	*OptimizedValue*	*ActualRate*
Knoxville	Henderson	CW	572383.99	0.038158
Knoxville	Henderson	Rx	8596675.84	0.032707

Figure 29. Records in the tranship.dbf table that are selected based on its link with the DCs attribute table. Reselecting for OptimizedValue field values greater than zero and Product field values equal to "Rx" yields the desired record (shaded).

Keep in mind that the flow volume used for this equation is the same value used in the other equations, and is not to be confused with the OptimizedValue field of the tranship.dbf VTAB.

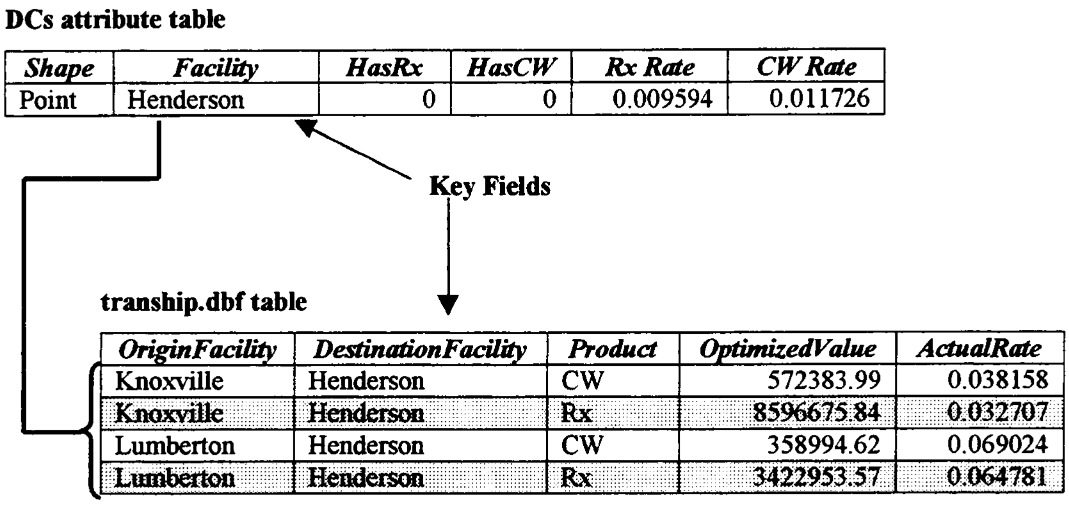
The final cost to calculate is the picking cost at the DC where the transhipment originates. To do this, the script gets the name of the DC from the OriginFacility field in the tranship.dbf VTAB, then it loops through the DCs FTAB until it finds the record for the DC with a matching name. It then obtains the value in the Rx Rate field for this record and multiplies it by the flow volume used throughout the other equations.

In our example, Knoxville is the origin DC for Henderson's transshipments. In the record for Knoxville, the value stored in the Rx Rate field is \$0.061443. Using this Rx handling rate for Knoxville, the picking cost for demand region 229 becomes

325,716.24 units × \$0.061443 per unit = \$20,012.98.

With transshipments, however, it can happen that a DC not stocking a certain product will receive transshipment from more than one other DC, so the script must take this situation into account. It does this by first summing the flow volume in the OptimizedValue field for all reselected transshipments going to the servicing DC. The script then loops back through the selected set of transshipments and determines the ratio of the transshipment flow volume to the total flow volume for each transshipment. It then multiplies the ActualRate field value for that transshipment by the ratio. After the rate for each selected transshipment has been factored by its proportion to the total flow volume, the factored rates are then summed to produce the final rate. In essence, the final rate is an average rate for the total transshipment flow volume that is weighted according to the volumes of each component flow. This weighted average rate is then multiplied by the total transshipment flow to get the overall transshipment cost. The same procedure is used to find the weighted average picking rate and the overall picking cost for the transshipments at the origin DCs. Average rates are used for both the transshipment and picking costs because it is ultimately not possible to determine the exact origin of a demand region's product volume, if that product type is supplied to the demand region's assigned DC by more than one origin DC. Of course, these averages are weighted because doing so yields a more accurate cost rate than just taking the simple average of the rates.

To illustrate these calculations, let us suppose that the Henderson DC receives Rx product transshipments not only from the Knoxville DC, but also from the Lumberton DC. When the record for the Henderson DC is selected in the DCs attribute table, the records for the transshipments from Knoxville and Lumberton are selected (Figure 30).



DCs attribute table

Shape	*Facility*	*HasRx*	*HasCW*	*Rx Rate*	*CW Rate*
Point	Henderson	0	0	0.009594	0.011726

tranship.dbf table

OriginFacility	*DestinationFacility*	*Product*	*OptimizedValue*	*ActualRate*
Knoxville	Henderson	CW	572383.99	0.038158
Knoxville	Henderson	Rx	8596675.84	0.032707
Lumberton	Henderson	CW	358994.62	0.069024
Lumberton	Henderson	Rx	3422953.57	0.064781

Figure 30. Records in the tranship.dbf table that are selected when the servicing DC receives transshipments from two origin DCs. Reselecting for OptimizedValue field values greater than zero and Product field values equal to "Rx" yields the records for both Knoxville's and Lumberton's Rx transshipment to the Henderson DC.

These records are then reselected for Rx product flows greater than zero, which yields the two records shaded in the figure. The Rx product flows into Henderson from these DCs are stored in the OptimizedValue field. These flows are summed to produce the total inflow of Rx product as follows,

$$\mathbf{8{,}596{,}675.84 + 3{,}422{,}953.57 = 12{,}019{,}629.41 \text{ total units.}}$$

Next, the script finds the proportion of each transshipment to the total flow. The Knoxville DC's contribution to the total flow is

$$\mathbf{8{,}596{,}675.84 \div 12{,}019{,}629.41 = 0.72}$$

or 72%, while the Lumberton DC's contribution is

$$\mathbf{3,422,953.57 \div 12,019,629.41 = 0.28}$$

or 28%. Next, the script multiplies the ratio for each transshipment by the transshipment rate stored in the ActualRate field of the tranship.dbf table, and computes their sum. Thus, the weighted average rate is

$$\mathbf{(.72 \times \$0.032707) + (.28 \times \$0.064781) = \$0.023549 + \$0.018139 = \$0.041688.}$$

This weighted average rate is then multiplied by the flow volume, as follows

$$\mathbf{325,716.24 \text{ units} \times \$0.041688 \text{ per unit} = \$13,578.47}$$

to produce the transshipment cost. These same ratios apply to the Rx handling rates for the origin DCs. Multiplying the weighted average Rx handling rate by the flow volume produces a picking cost for demand region 229 of **\$20, 806.75** (see Figure 31 for details).

Knoxville Rx handling rate =	\$0.061443
Lumberton Rx handling rate =	\$0.070147

Weighted average Rx handling rate (wahr) is

$$\text{wahr} = (.72 \times \$0.061443) + (.28 \times \$0.070147) = \$0.044239 + \$0.019641 = \$0.063880$$

$$\text{Picking Cost for demand region 229} = 325,716.24 \text{ units} \times \$0.063880 \text{ per unit} = \$20,806.75$$

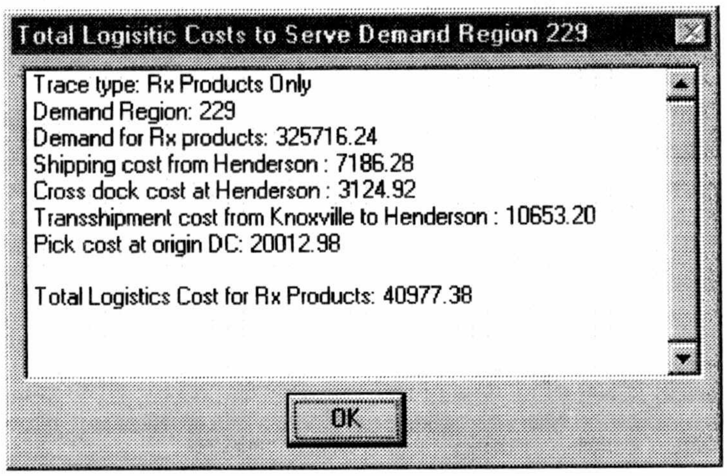
Figure 31. Computing picking cost at two transshipment origin DCs for demand region 229.

Once these transshipment and picking costs have been calculated, RxTrace then sums the shipping, crossdocking, transshipment, and picking costs to determine the total logistics cost for that product type. These cost variables are initialized to zero at the beginning of the script, so that the same summing calculation can be used for demand regions that do not incur a crossdocking or transshipment cost for that product. Finally, RxTrace calls a Tables.Unlink (Appendix) script that unlinks all of the tables.

In the example of demand region 229, where only the Knoxville DC supplies Rx transshipments, the total logistics costs associated with Rx products would be

Shipping cost:	**$7,186.28**
+ Crossdocking cost:	**$3,124.92**
+ Transshipment cost:	**$10,653.20**
+ Picking cost:	**$20,012.98**
Total Rx Logistics Cost:	**$40,977.38**

The logic of the CWTrace (Appendix) script is virtually identical to the RxTrace script. The OTCTrace (Appendix) script, however, does not attempt to calculate the crossdocking and transshipment costs, because it is known in advance that each DC acts as a stocking point for OTC products. The TraceAll (Appendix) script launched by the Demand Regions All Products item under the Trace Costs menu does nothing more than run a combined version of the RxTrace, CWTrace, and OTCTrace scripts, which calculates the grand total of all the logistics costs by all the products combined. Each of these scripts issues a message box to the screen (Figure 32) which gives an itemized report of the logistics costs for the region, including all the component costs, as well as the total cost.



```
Trace type: Rx Products Only
Demand Region: 229
Demand for Rx products: 325716.24
Shipping cost from Henderson : 7186.28
Cross dock cost at Henderson : 3124.92
Transshipment cost from Knoxville to Henderson : 10653.20
Pick cost at origin DC: 20012.98

Total Logistics Cost for Rx Products: 40977.38
```

Figure 32. Report of total Rx logistics cost for demand region 229. This window pops up after clicking on demand region 229 with the Rx cost trace tool.

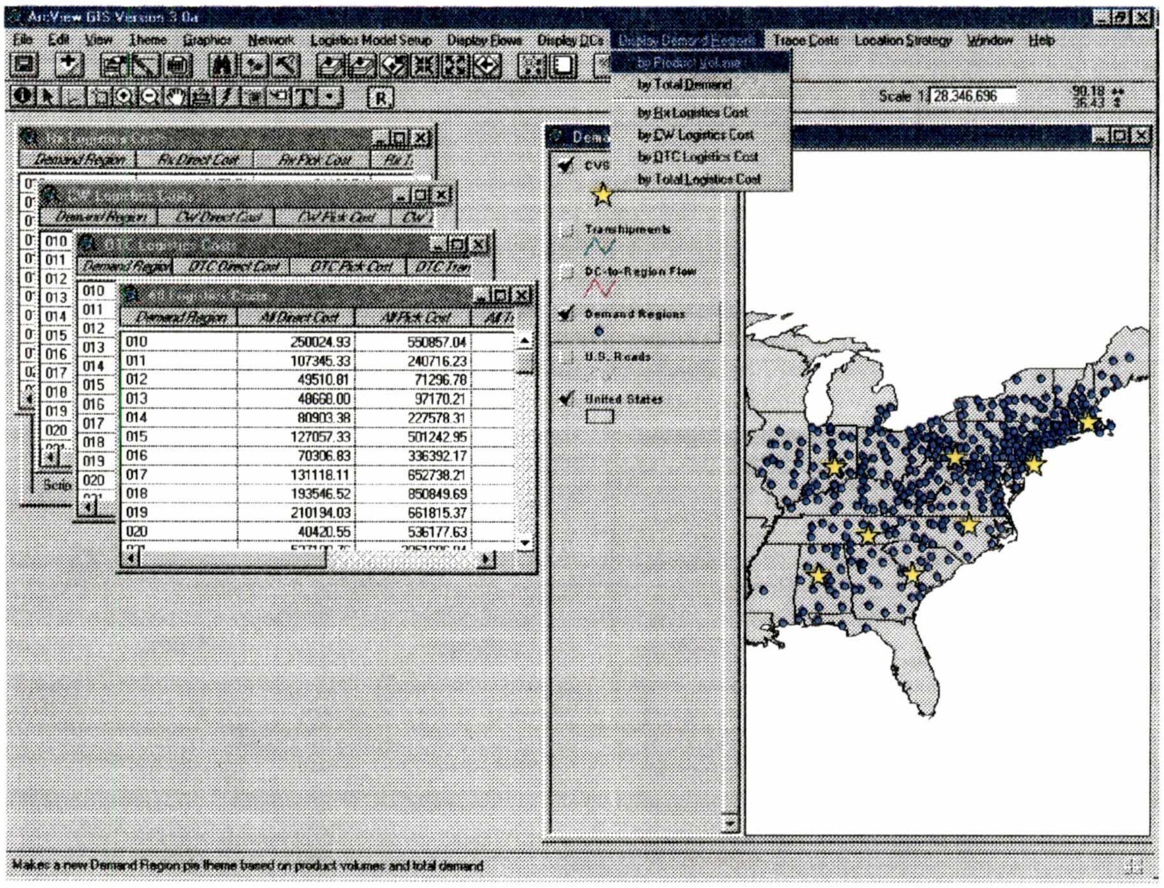
The trace cost tools for individual demand regions are accessible to the user through the Demand Region items under the Trace Cost menu, or from the drop-down tool menu provided at the far right end of the View document's toolbar. These tools are labeled "R", "C", and "O" for the Rx, CW, and OTC product traces, and "A" for the all products cost trace.

Tracing Logistics Costs for the Entire Network of Demand Regions

The scripts (TraceRxAll, TraceCWAll, TraceOTCAll – see Appendix) launched by the Chain-wide Rx Only and other product specific items falling in the same section under the Trace Costs menu vary only slightly from the TraceRx, TraceCW, and TraceOTC scripts described above. These scripts trace the logistics cost for each demand region in the Demand Regions theme and add the result as a new record to a new VTAB created to hold these values. When the costs for all the demand regions have been calculated, a new table made from the VTAB is created and joined to the Demand Regions attribute table. Because there are 384 demand regions to be calculated, these scripts take several minutes to complete. To create a similar VTAB holding each demand region's logistics costs for all products, the TraceAllAll (Appendix) does not recompute the logistics costs for each product type. Rather, it sums each demand region's cost values stored in the three product-specific logistics cost tables previously created. Utilizing these tables, calculating the total logistics costs for all products at each demand region takes only a few moments. Of course the Chain-wide All Products item remains disabled until the product specific cost calculations have been completed.

Once the total logistics cost tables for each product type, and for all products, has been added to the project and joined to the Demand Regions attribute table, the display-by-logistics-cost options listed under the Display Demand Regions menu are enabled (Figure 33). Clicking these items launches a script that clones the Demand Regions theme, and creates a pie chart legend for each demand region (Figure 34). The logic of these scripts (see the DRRxLogTheme.Make and similar scripts in Appendix) is similar to the ProdDemRegTheme.Make (Appendix) launched by the By Product Volume item that makes a pie chart legend for the demand regions. However, instead of representing volume by product type, the pie slices represent the component logistics costs, such as the shipping and picking costs, while the size of the pie represents the total logistics cost.

With the total logistics cost for each product grouping already made, a report of the total logistics costs for the entire network of demand regions can be generated. The



Demand Region	All Direct Cost	All Pick Cost
010	250024.93	550857.04
011	107345.33	240716.23
012	49510.81	71296.78
013	48668.00	97170.21
014	80903.38	227578.31
015	127057.33	501242.95
016	70306.83	336392.17
017	131118.11	652738.21
018	193546.52	850849.69
019	210194.03	661815.37
020	40420.55	536177.63

Figure 33. Logistics costs tables support the display of the Demand Regions theme by logistics cost. Note that the By Logistics Cost menu items are now enabled.

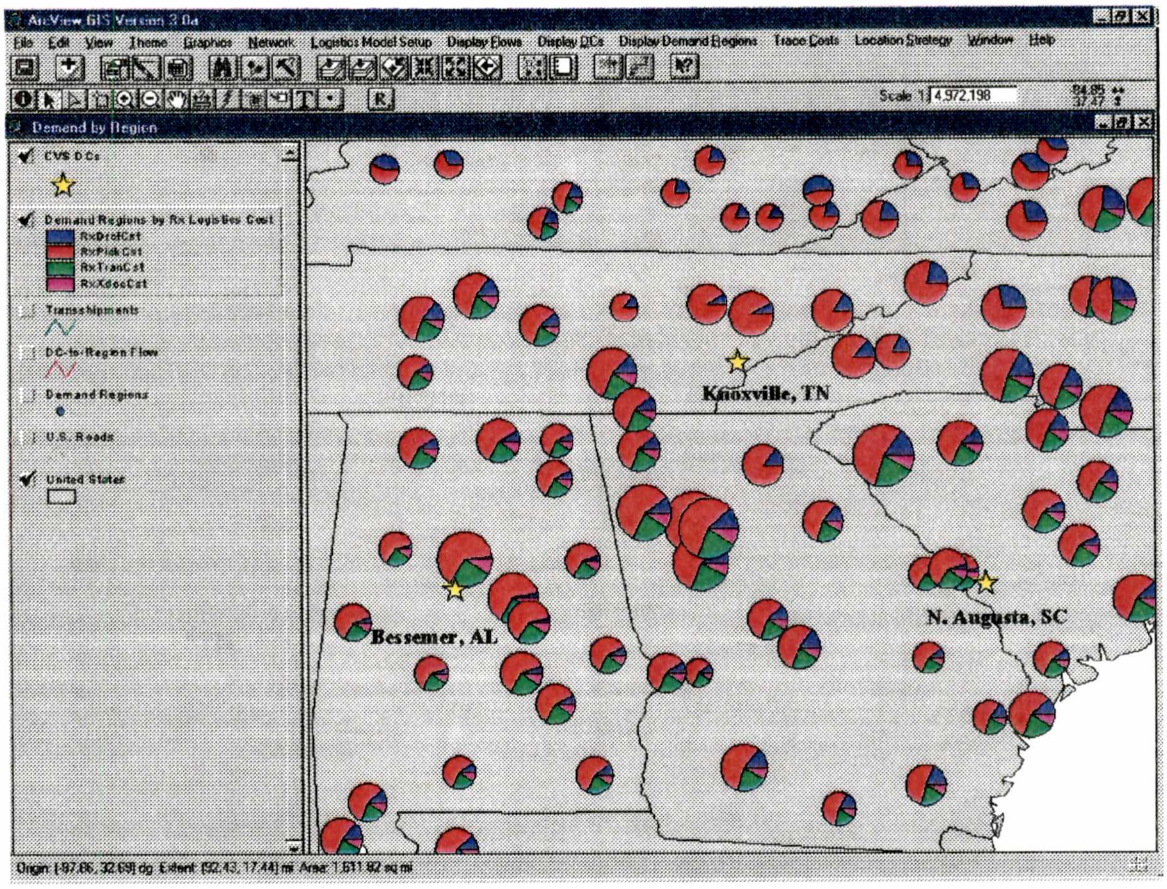


Figure 34. Demand Regions theme displayed by Rx logistics cost. The slices represent the shipping cost (blue), the picking cost (red), the transshipment cost (green), and the crossdocking cost (magenta), while the size of the pie represents the total Rx logistics cost.

Chain-wide Rx Statistics item, for example, launches the script RxStatistics.Generate (Appendix). This script takes each field in the Rx Logistics Costs table representing the component costs and the total costs, and sums them over all the records in the table. These totals are then reported to the user in a pop-up message box. The CWStatistics.Generate, OTCStatistics.Generate, and the AllStatistics.Generate (see Appendix) are identical in their operation to the RxStatistics.Generate script.

Chapter 5

Comparing DC Location Strategies

Generating the Service Areas

All of the custom functions reviewed so far demonstrate several of the powerful capabilities GIS technology incorporates, such as generating geometric shapes which represent real world events and their geographic location, embedding data about those events into the graphical features, changing the graphical display of thematic features to illustrate their attributes, and relating tables to enhance thematic data or to analyze trans-thematic events. While this list of capabilities is impressive in itself, none of the previous functions demonstrates what is perhaps the most powerful and most distinguishing capability of GIS technology – namely the analysis of spatial relationships between thematic features.

In the review of the CVS project's custom GIS functions, the importance of establishing a benchmark by which to judge organizational change and performance was addressed. In the case of a logistics network optimization that seeks to minimize costs, it would make sense to compare the total costs of the optimized network to the costs incurred by the network before optimization. Such a comparison, however, offers only a measure of the internal performance improvement of the company. It is common practice for many companies to undertake benchmark comparisons between itself and other companies within the industry. Applying this idea to CVS's network optimization project, it would be desirable, for example, to compare the optimized and pre-optimized network logistics costs of CVS against the network costs incurred by CVS's major competitors. Unfortunately, such operational data are jealously guarded by nearly all companies and are not made available to the public. That is why the cost and flow data in this thesis have been altered.

Data about the location of competitor DCs and stores, on the other hand, does reside in the public domain. By gathering the geographical reference data on the competing companies, it is possible to analyze the placement of each company's DCs among its network of stores by determining which of the DCs is closest for each store, and what that distance is. The store records can then be grouped by the range of distance within which the closest DC lies. For the purposes of this project, we chose five 50-mile ranges nested within each other to form a 250-mile service area around each DC (Figure 35). This grouping schema enables us to report the percentage of stores falling within 50 miles of the nearest DC, the percentage falling within 100 miles, and so on.

To set up the analysis, it is first necessary to create the themes for the DCs and stores of CVS's competitors, and to add them to a new view called Location Strategy. This is done manually by matching the zip code of each store to a theme of U.S. 5-digit zip code boundaries that is geocoded for address-matching. Similarly, the competitors' DCs are located by matching the city name of each DC to a theme of U.S. cities. The CVS stores will already have been address-matched and added to the view. The CVS DCs theme changes with each new optimization scenario, and is added automatically to the Location Strategy view when the new DCs theme is first added to the project by the SpliceLatLon script. The naming convention for these themes requires that each store theme be named after the company, followed by the identifying word "Stores", and each DC theme be name after the company, followed by the identifying word "DCs".

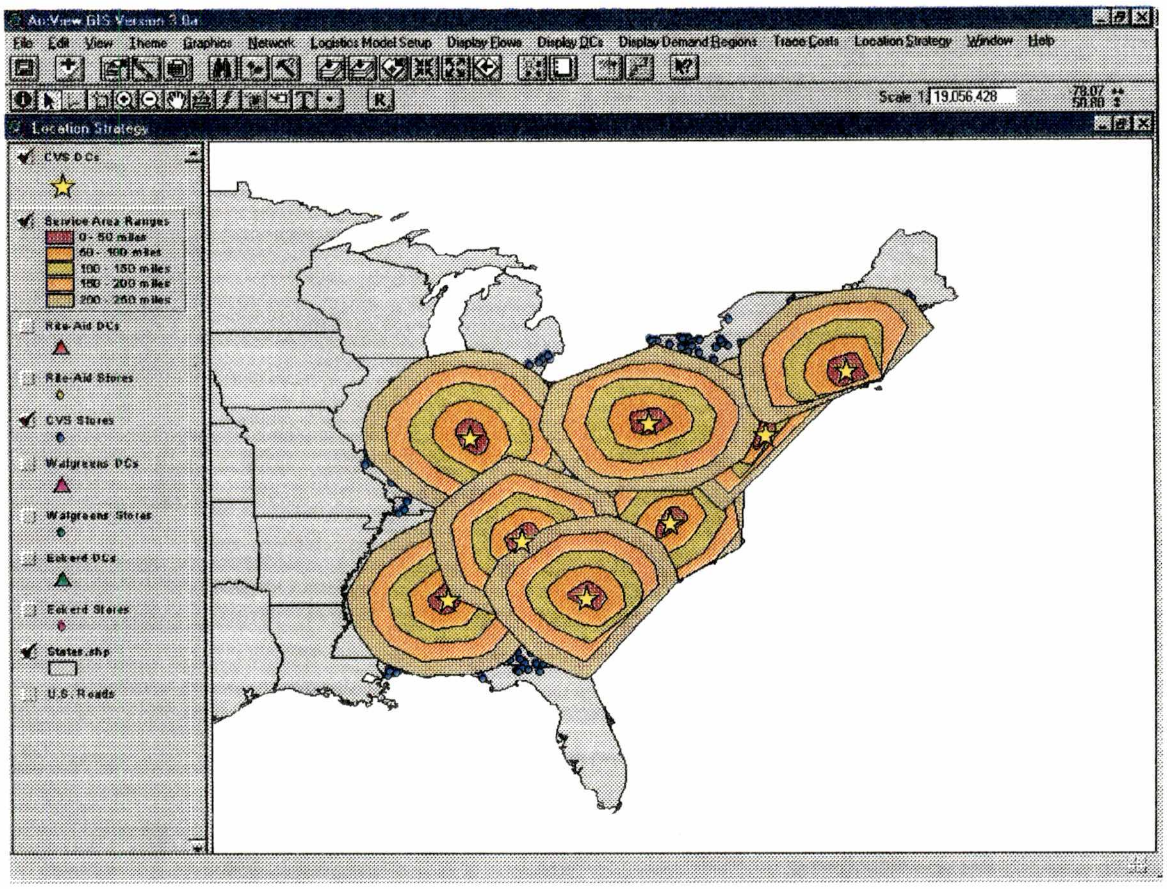


Figure 35. Service areas around DCs. Each service area has a 250-mile radius and is composed of five, 50-mile ranges nested within each other. Each of these ranges is a separate polygon having its own record in the Service Area FTAB.

This analysis also requires a line theme representing the road network over which a particular area will be serviced. This is because the algorithm ESRI's Network Analyst extension employs to find service areas on a network uses the costs associated with traversing network links to determine the geographic extent of the service area. These costs may be in distance units or time units. For the sake of processing speed and memory utilization, we selected a 1:2,000,000-scale road theme produced by ESRI that includes only interstate and state highways and major through-roads. In the road network we selected, the distance for each link was calculated in miles and located in a field called Miles. Once all the DC and store themes are in the Location Strategy view, and that view has been made the active window, then the items under the Location Strategy menu are enabled (Figure 36).

Clicking on the DC Range menu item launches a script Stores.SelectByRange (Appendix). This script prompts the user to select the line theme that will be used for generating the service areas. The user is then asked to select the DCs theme around which the service areas will be generated. The script uses the line theme to define and build a network file that the FindServiceArea algorithm will use. Although the 250-mile extent of the five, 50-mile nested service areas is an arbitrary set of distance costs, it was hard-coded into the script in order to ensure the comparability of different drugstore chains and their DC location strategies. Finally, passing the DCs FTAB and the set of distance costs as arguments, the script calls the FindServiceArea function which operates on the network.

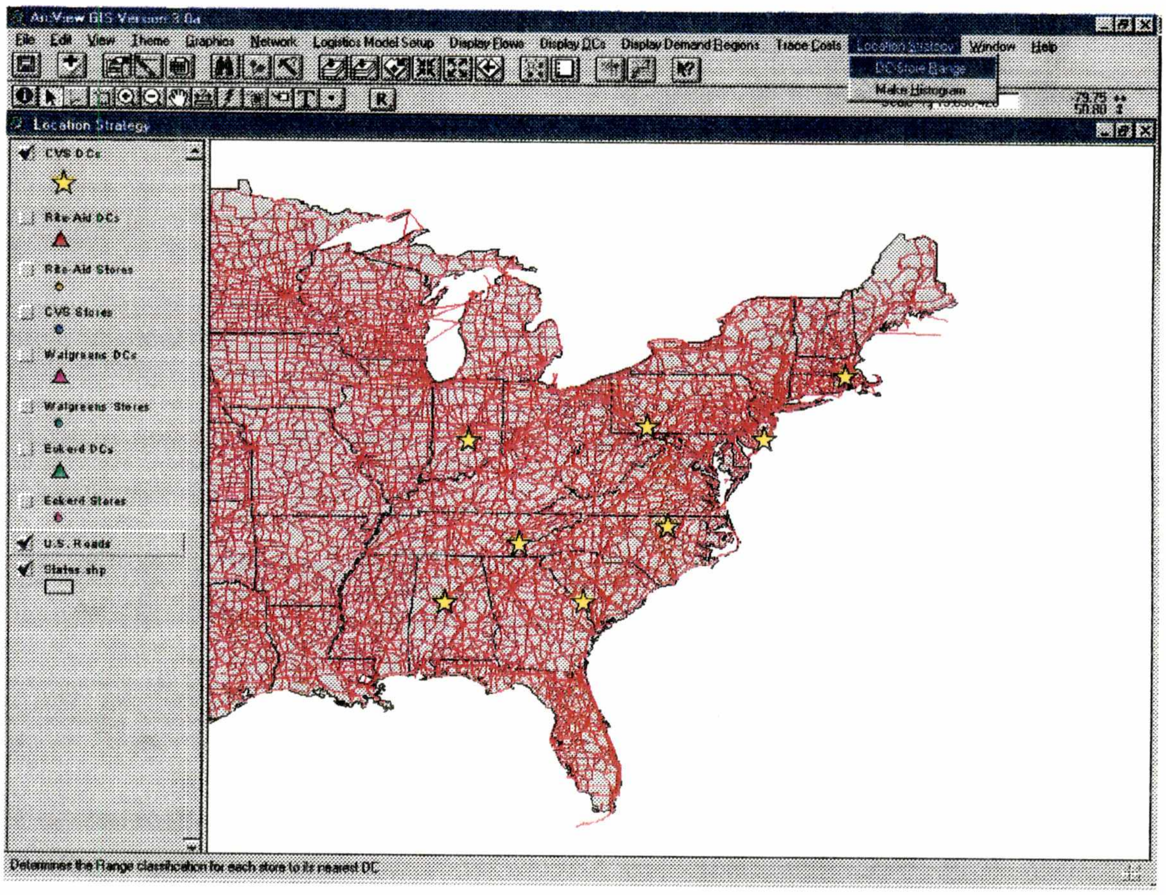

ArcView GIS Version 3.0a
File Edit View Theme Graphics Network Logistics Model Setup Display Flows Display DCs Display Demand Regions Trace Costs Location Strategy Window Help
DC Store Range
Make Histogram
Location Strategy
CVS DCs
Rite Aid DCs
Rite Aid Stores
CVS Stores
Walgreens DCs
Walgreens Stores
Eckerd DCs
Eckerd Stores
U.S. Roads
States.shp
Determines the Range classification for each store to its nearest DC

Figure 36. Location strategy menu and U.S. road network theme (red lines) displayed. The location strategy menu items are disabled until the Location Strategy view is made active. The U.S. Roads attribute table contains a Miles field that holds the distance cost values used by the Network Analyst algorithm to calculate the service areas.

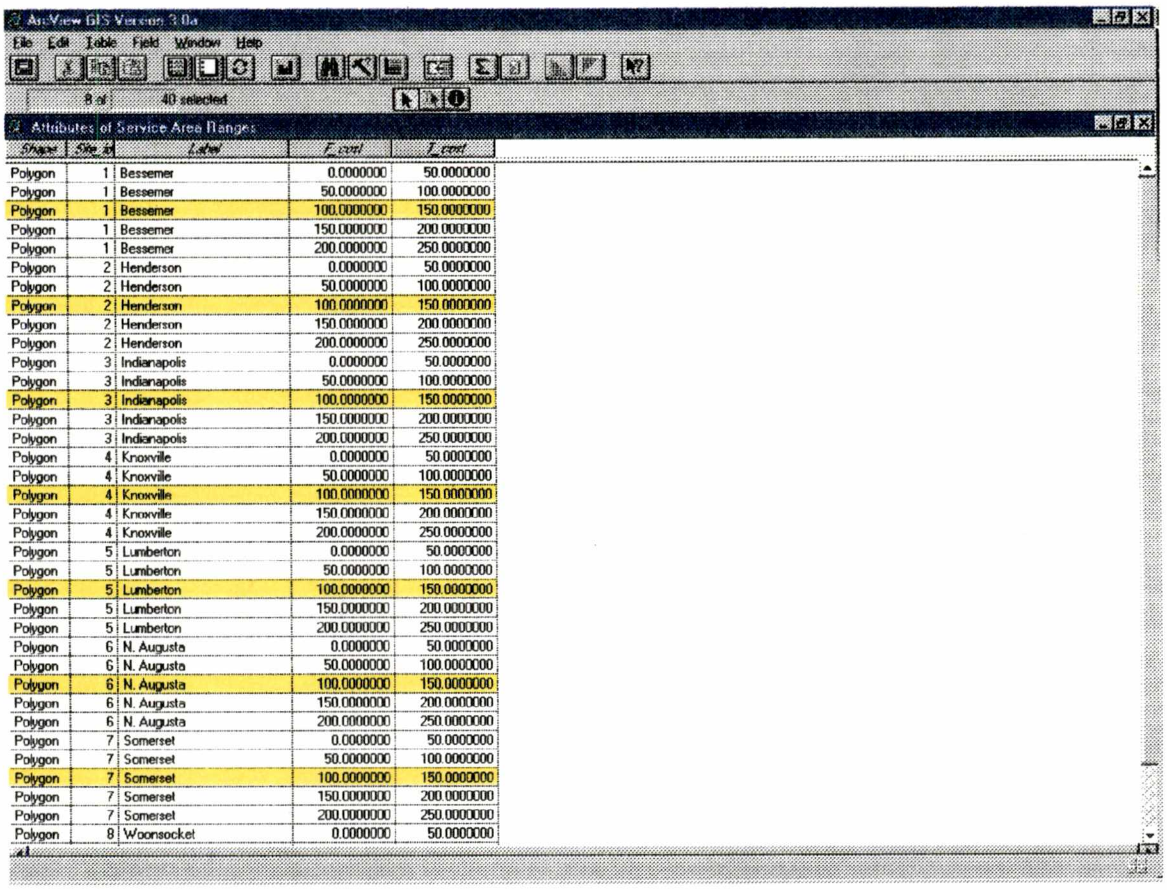
The result of the FindServiceArea function is two themes. One is a line theme representing the portion of the road network falling within the service area extent. The other is a polygon theme which defines the two-dimensional geographic extent covered by the service area. The latter, called the Service Area theme, is used to perform a SelectByFTab operation on the stores FTAB from the respective company. Before doing this, the script adds to the stores FTAB a new field called DC Range whose value indicates the range in which each store's closest DC lies. This field's value for each record is then initialized to zero.

The SelectByFTab request is made on the FTAB of one theme, and selects all those features in that theme that have a defined spatial relationship to another theme. In this project, the SelectByFTab request is made on the stores FTAB, and selects all those point features in the stores theme that intersect the features of the service area theme. However, we want to be careful to distinguish between those stores falling within the first 50-mile range of the nearest DC from those falling within the second, and the third, etc. To accomplish this, the script must first select the records in the service area FTAB representing the 0 – 50 mile range polygons for each DC.

The Service Area FTAB contains 5 polygon records (representing the 0-50 mile, 50-100 mile, 100-150 mile, 150-200 mile, and 200-250 miles service ranges respectively) for each DC. The number of records in the Service Area FTAB, therefore, will always be (5 $*$ n), where n is the number of DCs in the company's logistics network. Since the records in the Service Area FTAB are sorted by DC and not by range value, selecting out each unique distance range, such as the 50-mile range, for each DC requires selecting every fifth record in the Service Area FTAB by looping through it. The record number from which the loop is started determines which unique distance range is being selected. Starting on record zero in the Service Area FTAB (which is the 50-mile range of the first DC) will select all the 50-mile range records for all the DCs. Starting on record one (the 100-mile range of the first DC) will likewise select all the 100-mile range records for all the DCs, and so on (Figure 37).

Each time the SelectByFTab request is made on the stores FTAB using the currently selected range value, the resulting selection of stores is passed as an argument in a call to a script DC.Range (Appendix), as is the current range value. The DC.Range script receives the stores FTAB and the range value and then populates the DC Range field for the selected set of stores with the current range value. For example, after making the RequestByFTab request using the selected set of 50-mile ranges in the Service Area FTAB, the resulting selection of stores is passed to the DC.Range script and each selected record is given a value of 50 for the DC Range fields, meaning that the closest DC for these stores is within 50 miles. DC.Range then returns control to the Stores.SelectByRange script. This same procedure is repeated for the 100-mile, 150-mile, 200-mile, and 250 mile ranges (Figure 38). After the last SelectByFTab request is made, those store records that still have a DC Range value of zero are, by default, more than 250 miles away from the nearest DC, and so the value of their DC Range field is calculated to 251. At this point, the Stores.SelectByRange script terminates.

Since many of the service areas overlap, the situation can arise where a store will be within one range of one DC, and within another range of another DC, and therefore will be selected more than once. To ensure that the DC Range field holds the distance value to the closest DC, the DC Range field value for a store is calculated only if its value



ArcView GIS Version 3.0a

8 of 40 selected

Attributes of Service Area Ranges

Shape	Site_id	Label	F_cost	T_cost
Polygon	1	Bessemer	0.0000000	50.0000000
Polygon	1	Bessemer	50.0000000	100.0000000
Polygon	1	Bessemer	100.0000000	150.0000000
Polygon	1	Bessemer	150.0000000	200.0000000
Polygon	1	Bessemer	200.0000000	250.0000000
Polygon	2	Henderson	0.0000000	50.0000000
Polygon	2	Henderson	50.0000000	100.0000000
Polygon	2	Henderson	100.0000000	150.0000000
Polygon	2	Henderson	150.0000000	200.0000000
Polygon	2	Henderson	200.0000000	250.0000000
Polygon	3	Indianapolis	0.0000000	50.0000000
Polygon	3	Indianapolis	50.0000000	100.0000000
Polygon	3	Indianapolis	100.0000000	150.0000000
Polygon	3	Indianapolis	150.0000000	200.0000000
Polygon	3	Indianapolis	200.0000000	250.0000000
Polygon	4	Knoxville	0.0000000	50.0000000
Polygon	4	Knoxville	50.0000000	100.0000000
Polygon	4	Knoxville	100.0000000	150.0000000
Polygon	4	Knoxville	150.0000000	200.0000000
Polygon	4	Knoxville	200.0000000	250.0000000
Polygon	5	Lumberton	0.0000000	50.0000000
Polygon	5	Lumberton	50.0000000	100.0000000
Polygon	5	Lumberton	100.0000000	150.0000000
Polygon	5	Lumberton	150.0000000	200.0000000
Polygon	5	Lumberton	200.0000000	250.0000000
Polygon	6	N. Augusta	0.0000000	50.0000000
Polygon	6	N. Augusta	50.0000000	100.0000000
Polygon	6	N. Augusta	100.0000000	150.0000000
Polygon	6	N. Augusta	150.0000000	200.0000000
Polygon	6	N. Augusta	200.0000000	250.0000000
Polygon	7	Somerset	0.0000000	50.0000000
Polygon	7	Somerset	50.0000000	100.0000000
Polygon	7	Somerset	100.0000000	150.0000000
Polygon	7	Somerset	150.0000000	200.0000000
Polygon	7	Somerset	200.0000000	250.0000000
Polygon	8	Woonsocket	0.0000000	50.0000000

Figure 37. Service Area attributes table with every fifth record selected. This example shows the 150-mile service area range, which extends from 100 to 150 miles away from each DC. Note that the records are sorted by DC and not by range value.

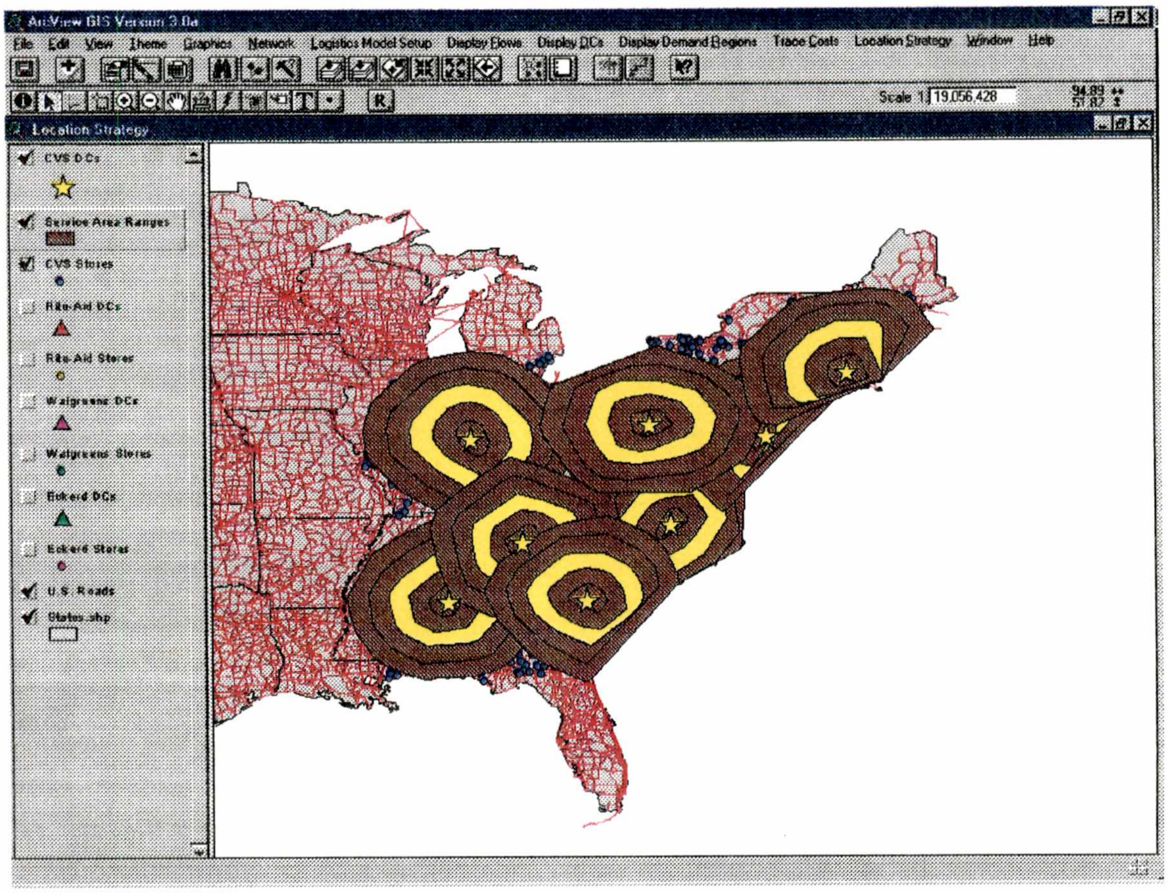


Figure 38. The 150-mile service area range selected for each DC. The selected ring polygons are shown in yellow. The gold stars are the DCs around which the service area ranges are centered.

is zero, which means the store has not yet been selected. If it already has a value greater than zero, it means the distance range to the nearest DC has already been determined.

Making the Histogram

Once the DC Range field has been populated, a histogram of the percent of stores falling within each distinct range from the nearest DC may be created. The making of the histogram is initiated by clicking on the Make Histogram item under the Location Strategy menu. The stores theme that is currently active is the theme for which a histogram is created. The Make Histogram item launches a script DC.HistPct (Appendix) which gets the FTAB of the active stores theme. It then creates a file in memory to hold the histogram data and generates a new VTAB from that file. The VTAB is given three new fields, a Range field to hold the range label, a Count field to hold the count of stores falling within that range, and a Percent field to hold the percentage that count is compared to the total store count. Next, a record for each unique range value, including the range value of 251, is added to the VTAB and its label is set in the Range field.

The script then loops through the stores FTAB, getting the value of the DC Range field for each record, and incrementing the Count field of the corresponding record in the histogram VTAB. Once the count of stores falling within each range is complete, the script calculates the value of the Percent field for each range by taking the value in the Count field, dividing it by the total number of stores in the stores FTAB, and multiplying by 100. With the histogram VTAB complete, the script makes a histogram chart from it. The chart is titled and then added to the project (Figure 39). Once charts for all the companies have been created, the changes in CVS's DC configuration with each new

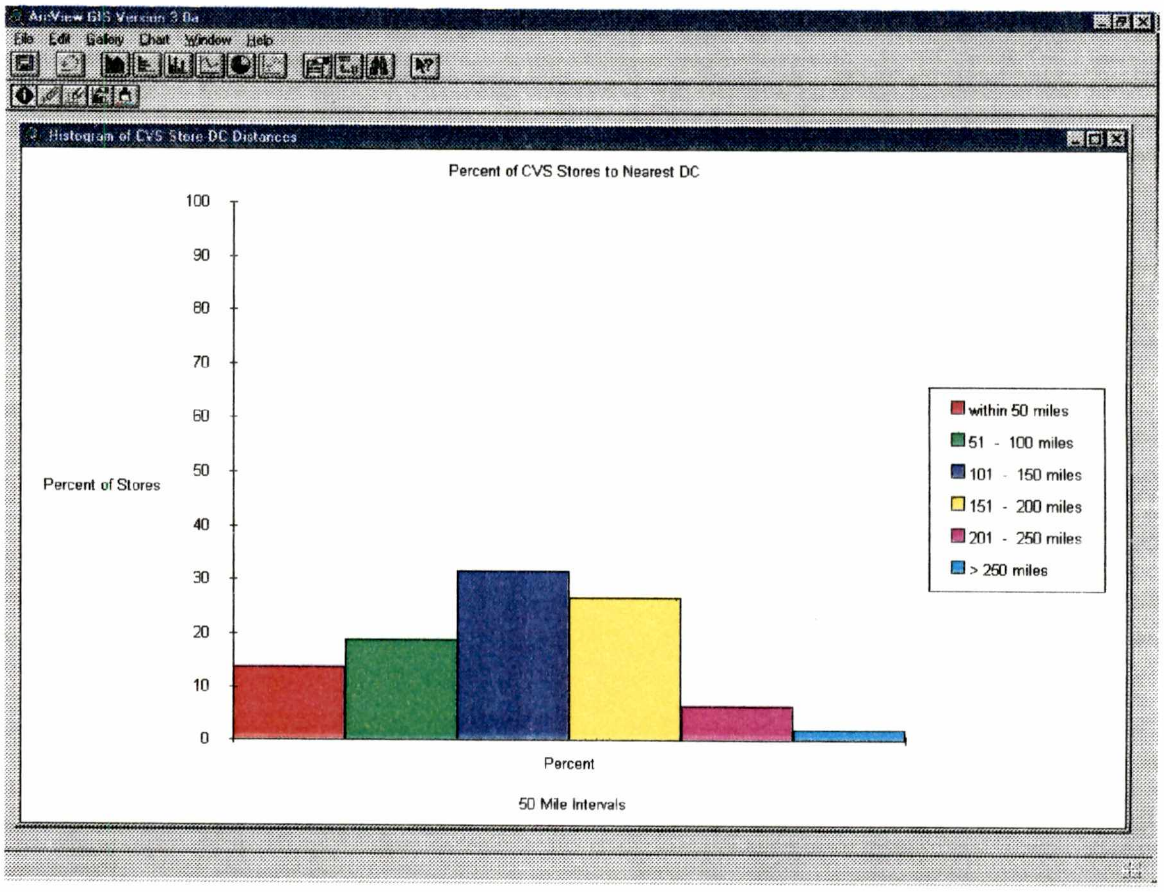


Figure 39. Histogram showing the percent of stores falling within each range of the nearest DC for the CVS sample network.

optimization scenario may be judged against the DC network configurations of one or more industry competitors (Figure 40).

By analyzing these charts, and the maps from which they were generated, a logistics manager may be able to arrive at a better understanding of how the location and number of DCs affect the kind of distances which must be traversed to supply the network of stores, and therefore affect the shipping costs for drugstore products. Combining this information with reports about the industry's logistics trends and publicly available statistics on competitor companies, such as asset utilization, operating costs, or inventory carrying costs, will also give a logistics manager an idea of where a proposed network stands in relation to the industry in general.

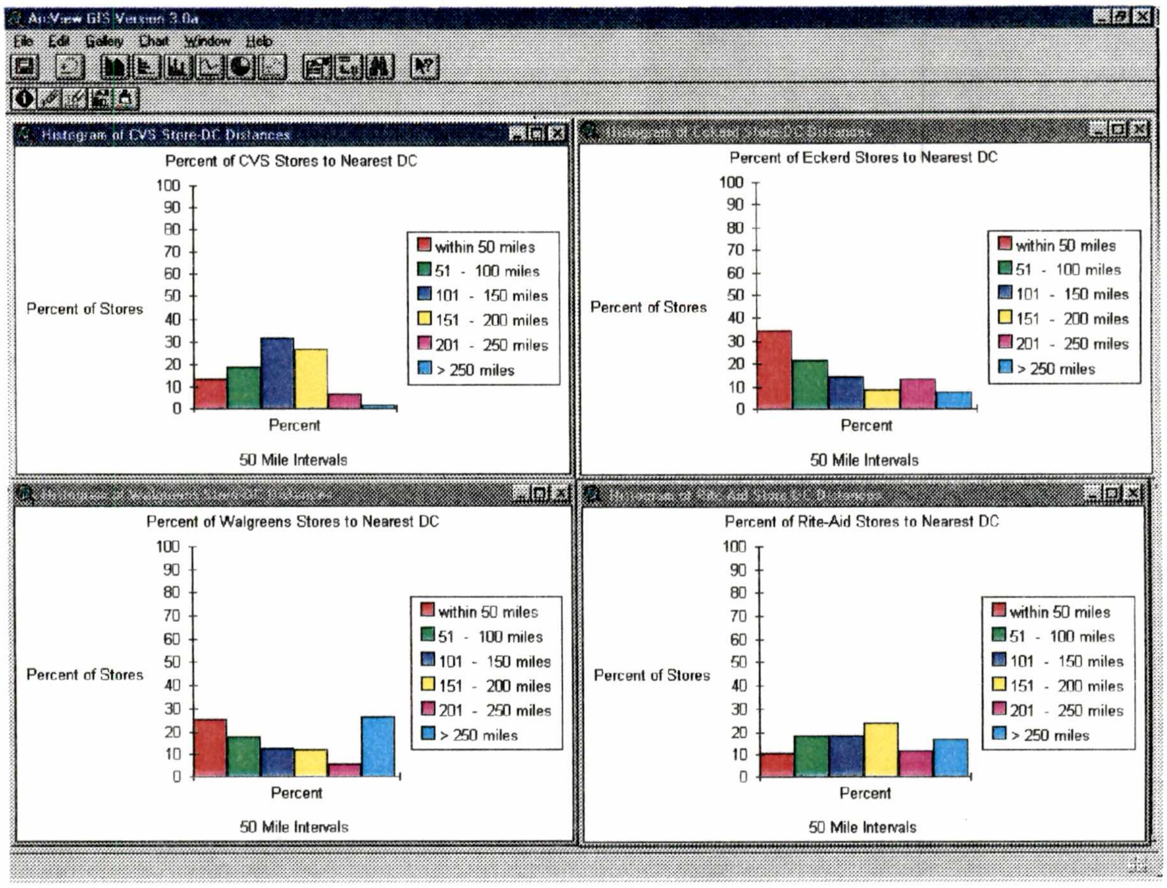


Figure 40. Using histograms to compare the DC configuration of a CVS sample optimized network to the DC configurations of competitors' networks.

Chapter 6

Conclusion

Advantages of GIS for Logistics

The purpose of this thesis has been to highlight the unique advantages GIS technology offers the business logistics industry for analyzing complex logistics systems, or "supply chains." One of these advantages is the ability to visualize data. Visualization has become an important area in the fields of information and computer science over the past several decades, and its rise has been fueled largely by the previous success of those same fields. As information systems have become more powerful and sophisticated since their arrival to mainstream business in the 1960s, the volume of data companies have collected about their operations has become staggering. Consequently, corporate information strategies are moving their emphasis away from data collection toward data exploration, in an effort to leverage corporate information resources to remain competitive in the accelerating, global economy of the 1990s and beyond. The ability of GIS technology to not only graphically display geographically-referenced data, but also to allow more intuitive interaction with the data, gives GIS users the ability to filter pertinent information from vast databases, to effectively communicate complex data sets, to optimize the processing of data, and to guide the analytical methods of data modeling and interpretation.

The other advantage of GIS technology is its ability to process spatial data. It is increasingly recognized that GIS is not just useful for automated mapping, and thus limited to "map intensive" industries. A large portion of corporate data resources contain references to geographic location, and therefore can take advantage of GIS technology and visualization. So, more and more companies are beginning to explore ways to "mine" these data out of their databases. This trend has been encouraged by the growth, and consequent decline in cost, of spatial data resources offered by both commercial data companies and by government agencies, making it much easier for companies to acquire the base maps necessary for referencing their internal data. While the early thrust of business GIS applications has been in the areas of sales territory management, niche marketing, and retail location analysis, other corporate functional areas are beginning to see the advantage of GIS technology. My argument has been that business logistics, especially because it is an inherently spatial discipline, represents an ideal application for GIS technology, and in Chapter 1 I cited several instances in which GIS technology has been successfully applied to this area.

As a practical example of a GIS application to logistics, I have reviewed the CVS distribution network optimization project for which I worked, and the custom GIS application developed for the project using ESRI's ArcView GIS software. This review began with a discussion of the graphical displays the application can produce in order to demonstrate the power GIS technology possesses for communicating information about large, complex sets of data, such as occurs with the logistics networks of major corporations. I have also explained in detail the Avenue scripts controlling the project's custom functions. The customization process fell naturally into three sections, and so my review of the scripts was likewise broken into three chapters – Chapters 3, 4, and 5.

In Chapter 3, I reviewed the ArcView Avenue scripts responsible for importing and visualizing the results of an exogenous logistics optimization application. This section of the thesis emphasized the ease with which logistics network data, if organized in a predefined and consistent format, can be brought into the ArcView application. It also highlighted the options for displaying information about the network features – such as DCs, demand regions, and the product flows that occur between them – that resulted from summarizing tabular data for the network DCs and demand regions.

In Chapter 4, I reviewed the scripts that give the custom GIS application a simple and intuitive interface between the user and the underlying network data. I demonstrated how, with the click of the mouse, the user could access complete logistics cost data, both for an individual demand region and for the network as a whole. It provided a good example of how information may be embedded in objects, thus allowing the user to focus on the geographical distribution of the information, rather that on the processing of the data. This chapter also emphasized the ability of GIS to manipulate data about trans-thematic events using the table linking capabilities of a relational database.

Finally, in Chapter 5, I discussed the scripts employing theme-on-theme spatial analysis that enabled the application to create histograms that could be used for comparing the DC configuration of an optimized logistics network with those of industry competitors. These histograms, combined with the thematic mapping of the network based on the logistics cost tables produced by the network-wide trace cost functions, make it possible for a logistics manager to understand quickly how different network optimization scenarios will affect the total logistics costs of the retail facilities throughout the chain, and how a solution compares to industry benchmarks.

Suggestions for Improving the CVS Project's GIS Customization

Although there was a considerable amount of thought and programming that went into the development of CVS's ArcView customization, it could have better demonstrated the potential of GIS for business logistics if the GIS portion of the project had been granted more scope. The use of GIS for CVS's network optimization project was limited to the display and analysis of the optimization results. Given the opportunity, ESRI's ArcView software, or some other GIS software, might have been integrated with a linear program solver to create a seamless application in which the optimization, display, and analysis of solutions could be performed in a single working environment, thus avoiding the confusion and time delays associated with having separate and geographically dispersed groups working on different aspects of the project. A single working environment would also avoid the necessity of going through a third application, such as Microsoft Access, to transfer the data between applications, and may have resulted in a more efficiently organized database needing less data querying and summarization to extract useful feature attribute information. The application of GIS to CVS's logistics operations might also have been broadened to include other aspects of logistics analysis, such as inventory monitoring, retail store location, and demand forecasting using geo-demographic data.

The Future of Using GIS for Logistics Analysis

It is possible to speculate about the reasons why GIS, in the CVS customization and in general, does not get applied more comprehensively to business information processes. The notions that GIS is a tool of use only to geographers, that current GIS applications are too complex and generic for traditional business information processes, and that widespread ignorance of spatial analysis retards the growth of GIS technology, were already touched upon in the first chapter. What seems to be true is that, increasingly, computer technology is user-driven, rather than application-driven, and therefore tends to focus more on minimizing implementation time and maximizing ease of use, rather than on application robustness and flexibility. As a result of this "plug and play" mentality, the last decade has witnessed a boom in user-friendly, desktop applications targeted to the information needs of specific niche industries, including business logistics. Today, applications such as InterTrans Logistics Solutions' *Supply Chain Strategist* and Caps Logistics' *ToolKit* are giving the power of the operations research consultant to a desktop computer sitting on the logistics manager's desk.

Because of the increasing supply of specialized, user-friendly, logistics software, it may be difficult for GIS technology to penetrate the field of business logistics if it remains a complex, generic application. To rectify this situation, it has been suggested that the GIS industry needs to develop applications that are tailored to specific business industries. Zwart has analyzed GIS industry trends and argues that GIS technology is also becoming user-oriented, and that many applications – in particular desktop mapping systems that have limited, more intuitive functionality – have been directed to particular market/user segments and have contributed to the growth of GIS outside of the traditional, land-based industries (Zwart, 1993). It appears, however, that business logistics has not been targeted. Ralston has corroborated this assertion by pointing out the lack of comprehensive logistics algorithms in contemporary GIS applications, and he argues that until they are included, the logistics industry will find GIS of limited use (cited in Black, 1997).

Emerging Trends in GIS Technology

There are several trends, however, suggesting that GIS technology is converging with mainstream information systems, and that using GIS technology to analyze and visualize business activities, and thereby to leverage corporate data, will become commonplace in the future. As the desktop computer running a version of Microsoft Corp.'s Windows operating system becomes the de facto standard for business computing environments, most GIS technology developers are moving to make their applications compliant with Microsoft's object-linking-and-embedding (OLE) automation model (Francica, 1998). OLE compatibility will result in easier and quicker integration of GIS objects, such as thematic maps, into other business applications, such as spreadsheets or word processors, and vice versa.

Progress has also been made in building spatial data processing capabilities directly into database management systems, as evidenced by the introduction of ESRI's spatial data engine (SDE), Oracle's spatial data option (SDO) and Spatial Cartridge server extension, and Informix's Spatial Datablade module (Szajgin, 1997; Francica, 1998). As spatial data objects and processing capabilities become more widely understood and incorporated into mainstream information technologies, GIS applications as comprehensive, stand-alone tools are likely to devolve into GIS function and object libraries, such as ESRI's MapObjects, that will enable IS managers and application developers to selectively embed GIS functionality into their systems (Cooke and Montgomery, cited in Hughes, 1997).

Another trend is occurring on the academic side of GIS. As GIS technology becomes more advanced, and as it moves into the mainstream of information processing, academic geographers have begun a sophisticated dialog on the uses of GIS, both inside and outside the academic setting, and on the role academic geographers have to play in the diffusion of GIS technology in society. Johnson has discussed at length the increased interest of the business world in GIS technology, and mentions several GIS courses with a business orientation. He argues that academic geographers should assume a leading role in the teaching of GIS for business, and suggests it is time to consider developing a national curriculum on GIS in business (Johnson, 1996). When GIS and spatial analysis are formally taught as business tools in academic institutions, misunderstandings about GIS should give way to a rapid expansion of GIS technology into business information systems, including those of business logistics.

REFERENCES

References

Ballou, R. H. 1992. *Business Logistics Management, Third Edition* (Englewood Cliffs, New Jersey: Prentice Hall).

Barone, A. 1997. "Basically Visual," *WWS/World Wide Shipping*, Vol. 60, No. 5, pp. 20-21.

Black, W. R. 1997. "Conference Report: Transport Geography Sessions at the Association of American Geographers Annual Meeting, 1 April 1997, Fort Worth, Texas," *Journal of Transport Geography*, Vol. 5, No. 3, pp. 221-223.

Buttenfield, B. P. and Mackaness, W. A. 1991. "Visualizaton," in D. Maguire, M. Goodchild and D. Rhind, eds., *Geographical Information Systems: Principles and Applications* (Essex, England: Longman Scientific & Technical).

Camm, J. D., Chorman, T. E., Franz, A. D., Sweeney, D. J., and Wegryn, G. W. 1997. "Blending OR/MS, Judgement, and GIS: Restructuring P&G's Supply Chain," *Interfaces*, Vol. 27, No. 1, pp. 128-142.

Coppock, J. T. and Rhind, D. W. 1991. "The History of GIS," in D. Maguire, M. Goodchild and D. Rhind, eds., *Geographical Information Systems: Principles and Applications* (Essex, England: Longman Scientific & Technical).

Dewitt, W. J. and Ralston, B. A. 1996. "GIS: Existing and Potential Applications for Logistics and Transportation," in *Business Geographics for Educators and Researchers, 30 May 1996 Proceedings* (Washington D.C.: Association of American Geographers and GIS World, Inc.)

Dewitt, W. J., Ralston, B. A., and Langley, J.C. 1997. "The Impact of GIS on Supply Chains," in Brace, G., ed., *Logistics Technology International 1997* (London: Sterling Publishing Group).

Francica, J. R. 1998. "Corporate GIS: Spatial Information Technology is Now in the Mainstream," *Business Geographics*, Vol. 6, No. 3, pp. 34-35.

Fung, D. S. and Remsen, A. P. 1997. "Geographic Information Systems Technology for Business Applications," *Journal of Applied Business Research*, Vol. 13, No. 3, pp. 17-24.

Gates, L. 1997. "GIS Puts Supply Chains on the Map," *Software Magazine*, Vol. 17, No. 10, pp. 69-73.

Goodchild, M. F. 1991. "The Technological Setting of GIS," in D. Maguire, M. Goodchild and D. Rhind, eds., *Geographical Information Systems: Principles and Applications* (Essex, England: Longman Scientific & Technical).

Grimshaw, D. J. 1994. *Bringing Geographical Information Systems Into Business* (Essex, England: Longman Scientific & Technical).

Hamilton, D. 1996. "A Mappable Feast," *CIO*, Vol. 9, No. 11, pp. 64-66.

Harder, C. 1997. *ArcView GIS Means Business* (Redlands, California: Environment Systems Research Institute, Inc.).

Hughes, J. R. 1997. "GIS Industry Outlook: The Dawn of a New Decade," *GIS World*, Vol. 10, No. 12, pp. 40-47.

Huxhold, W. E. and Levinsohn, A. G. 1995. *Managing Geographic Information System Projects* (New York: Oxford University Press).

Jacobs, A. 1996. "GIS Technology Makes Inroads," *Computerworld*, Vol. 30, No. 24, p. 71.

Johnson, M. 1993. "GIS Popularity Growing," *Computerworld*, Vol. 27, No. 12, pp. 41-42.

Johnson, M. L. 1996. "GIS in Business: Issues to Consider in Curriculum Decision-Making," *Journal of Geography*, Vol. 95, No. 3, pp. 98-105.

Korte, G. 1997. *The GIS Book* (Santa Fe, New Mexico: OnWord Press).

Rao, S.S. 1995. "Corporate Treasure Maps," *CIO*, Vol. 8, No. 18, pp. S1-S6.

Rogers, E. M. 1993. "The Diffusion of Innovations Model," in I. Masser and H. Onsrud, eds., *Diffusion and Use of Geographic Information Technologies* (Dordrecht, The Netherlands: Kluwer Academic Publishers).

Sherwood, N. 1995. "'Business Geographics' - A U.S. Perspective," in P. Longley and G. Clarke, eds., *GIS for Business and Service Planning* (New York: John Wiley & Sons, Inc.).

Szajgin, J. 1997. "Blending GIS and IT," *Business Geographics*, Vol. 5, No. 1, pp. 36-39.

Swenson, J. 1996. "GIS Software Goes Corporate," *InformationWeek*, No. 582, p. 103.

Tetzeli, R. 1993. "Mapping for Dollars," *Fortune*, Vol. 128, No. 9, pp. 90-96.

Tufte, E. R. 1997. *Visual Explanations: Images and Quantities, Evidence and Narrative* (Cheshire, Connecticut: Graphics Press).

Zwart, P. R. 1993. "Embodied GIS - A Concept for GIS Diffusion," in I. Masser and H. Onsrud, eds., *Diffusion and Use of Geographic Information Technologies* (Dordrecht, The Netherlands: Kluwer Academic Publishers).

Appendix

```
'*****************************************************************
' Scriptname:   AddXY
'
' Filename:     addxy.ave
'
' Description:  Adds two new fields, named X-coord and Y-coord,
'               to the table of the first active theme in the
'               TOC and fills the respective fields with the X,Y
'               coordinates of the selected points (or all points
'               if no selection is defined) in a point theme.  If
'               instead the active theme is a polygon theme, then
'               the X,Y coordinates of the polygon centroid are
'               calculated. If the theme is projected, the output
'               coordinates will also be projected.
'
' Requires:     An active point or polygon theme. This script
'               does minimal error checking and assumes that
'               there is an active theme.

' Called by:    SpliceLatLon

' Calls:        Nil

' SELF:         Nil

' Returns:      an FTab
'****************************************************************
theView = av.GetActiveDoc
'must be global to work in Calc exp below
_theProjection = theView.GetProjection
project_flag = _theProjection.IsNull.Not  'true if projected
theTheme = theView.GetActiveThemes.Get(0)

'Check if point or polygon theme
if (((theTheme.GetSrcName.GetSubName = "point") or
    (theTheme.GetSrcName.GetSubName = "polygon")).Not) then
  MsgBox.Info("Active theme must be polygon or point theme","")
  exit
end

'get the theme table and current edit state
theFTab = theTheme.GetFTab
theFields = theFTab.GetFields
edit_state = theFTab.IsEditable

'make sure table is editable and that fields can be added
if (theFtab.CanEdit) then
  theFTab.SetEditable(true)
  if ((theFTab.CanAddFields).Not) then
    MsgBox.Info("Can't add fields to the table."+NL+
                "Check write permission.",
                "Can't add X,Y coordinates")
    exit
  end
else
  MsgBox.Info("Can't modify the feature table."+NL+
  "Check write permission.","Can't add X,Y coordinates")
  exit
end

'Check if fields named "X-coord" and Y-coord" exist
x_exists = (theFTab.FindField("X-coord") = NIL).Not
y_exists = (theFtab.FindField("Y-coord") = NIL).Not
if (x_exists or y_exists) then
  if (MsgBox.YesNo("Overwrite existing fields?",
  "X-coord, Y-coord fields already exist", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (x_exists) then
      theFTab.RemoveFields({theFTab.FindField("X-coord")})
    end
    if (y_exists) then
      theFTab.RemoveFields({theFTab.FindField("Y-coord")})
    end
  else
    exit
  end  'if (MsgBox...)
end  'if
x = Field.Make ("X-coord",#FIELD_DECIMAL,18,5)
y = Field.Make ("Y-coord",#FIELD_DECIMAL,18,5)
theFTab.AddFields({x,y})

'Get point coordinates or polygon centroid coordinates
if (theTheme.GetSrcName.GetSubName = "point") then
  if (project_flag) then
    'Projection defined
    theFTab.Calculate("[Shape].ReturnProjected
                      (_theProjection).GetX", x)
    theFTab.Calculate("[Shape].ReturnProjected
                      (_theProjection).GetY", y)
  else
    'No projection defined
    theFTab.Calculate("[Shape].GetX", x)
    theFTab.Calculate("[Shape].GetY", y)
  end  'if
else  'polygon case
  if (project_flag) then
    theFTab.Calculate("[Shape].ReturnCenter.ReturnProjected
                       (_theProjection).GetX", x)
    theFTab.Calculate("[Shape].ReturnCenter.ReturnProjected
                       (_theProjection).GetY", y)
  else
    theFTab.Calculate("[Shape].ReturnCenter.GetX", x)
    theFTab.Calculate("[Shape].ReturnCenter.GetY", y)
  end  ' if
end
'Return editing state to pre-script running state
theFTab.SetEditable(edit_state)

return theFTab
```

```
'*********************************************************

' Scriptname:     AllStatistics.Generate

' Filename:       Allstati.ave

' Author:         Kenneth Bennett

' Date:           May 6, 1998

' Description:    Script sums each of the fields in the
'                 All Logistics Costs table and reports
'                 it to the user.

' Requires:       All Logistics Costs table exists

' Called by:      View menu item click
'                 ("Trace Costs: Chain-wide All-Products
'                  Statistics")

' Calls:          Nil

' SELF:           Nil

' Returns:        Nil

'*********************************************************
Scriptname = "AllStatistics.Generate"

' Ensure two decimal places in each number

Script.The.SetNumberFormat("d.dd")

' Get the table

theTable = av.GetProject.FindDoc("All Logistics Costs")
if (theTable = Nil) then
  MsgBox.Error("All Logistics Costs table not found.",
               Scriptname)
  exit
end

' Get the VTab for the table

theVTab = theTable.GetVTab

' Get the number of records

num = theVTab.GetNumRecords

' Get the list of fields in the VTab

theFieldList = theVTab.GetFields

' Set the field variables
```

```
shipFld = theFieldList.Get(1)
pickFld = theFieldList.Get(2)
tranFld = theFieldList.Get(3)
xdocFld = theFieldList.Get(4)
totlFld = theFieldList.Get(5)

' Initialize summing variables

shipSum = 0
pickSum = 0
tranSum = 0
xdocSum = 0
totlSum = 0

' Loop through the VTab and sum each fld

for each rec in theVTab
  shipSum = shipSum + theVTab.ReturnValue(shipFld, rec)
  pickSum = pickSum + theVtab.ReturnValue(pickFld, rec)
  tranSum = tranSum + theVTab.ReturnValue(tranFld, rec)
  xdocSum = xdocSum + theVTab.ReturnValue(xdocFld, rec)
  totlSum = totlSum + theVTab.ReturnValue(totlFld, rec)
end

' Calculate the per store average for each field

shipAvg = shipSum / num
pickAvg = pickSum / num
tranAvg = tranSum / num
xdocAvg = xdocSum / num
totlAvg = totlSum / num

' Issue the report

reportString = "Total All-Products Shipping Cost:"++
                  shipSum.AsString+nl+
               "Average Per Region:"++shipAvg.AsString+nl+
               "Total All-Products Pick Cost:"++
                  pickSum.AsString+nl+
               "Average Per Region:"++pickAvg.AsString+nl+
               "Total All-Products Transhipment Cost:"++
                  tranSum.AsString+nl+
               "Average Per Region:"++tranAvg.AsString+nl+
               "Total All-Products Crossdock Cost:"++
                  xdocSum.AsString+nl+
               "Average Per Region:"++xdocAvg.AsString+nl+nl+
               "Chain-wide Grand Total Logistics Cost:"++
                  totlSum.AsString+nl+
               "Average Grand Total Cost Per Region:"++
                  totlAvg.AsString+nl

MsgBox.Report(reportString,
               "Chain-wide All-Products Statistics")

return Nil
```

```
'*****************************************************************

' Scriptname:    AllTraceAll

' Filename:      alltrace.ave

' Description:   Script sums the data stored in the Rx, CW, and
'                OTC Logistics costs tables to create a new
'                table of total logistics costs

' Requires:      Demand Regions theme, and the Rx, CW, and OTC
'                Logistics Costs tables must exist

' Called by:     View menu click event
                 ("Trace Costs: Chain-wide All Products")

' Calls:          Tables.Link, Tables.Unlink

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "AllTraceAll"

' Set the number format for the script

Script.The.SetNumberFormat("d.dd")

theDirectory = av.GetProject.GetWorkDir.AsString

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theStoreVTab = theTheme.GetFTab

' Get the store field from the Demand Regions FTab

thestorefld = theStoreVTab.FindField("Demand Region")
if (thestorefld = Nil) then
  MsgBox.Error("ERROR: Demand Region name field not found.",
               Scriptname)
  exit
end

'  Get the Rx, CW, and OTC cost tables

rxTable = av.GetProject.FindDoc("Rx Logistics Costs")
cwTable = av.GetProject.FindDoc("CW Logistics Costs")
otcTable = av.GetProject.FindDoc("OTC Logistics Costs")
if((rxTable = Nil) OR (cwTable = Nil) OR (otcTable = Nil)) then
  MsgBox.Error("One or more logistics cost tables are missing."
               +nl+"Bailing out of program...", "ERROR")
  exit
end

'check if table exists
sumallcst_exists = (av.GetProject.FindDoc
                     ("AllLgCst.dbf") = NIL).Not
skip = 0
if (sumallcst_exists) then
  thedoc = av.GetProject.FindDoc("AllLgCst.dbf")
  if (MsgBox.YesNo("Overwrite existing logistics cost table?",
    "The Table AllLgCst already exists",false)) then
    if (sumallcst_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    skip = 1
  end
end
'create a newtable
if (skip = 0) then
  flnm = theDirectory + "/AllLgCst.dbf"
  newVTab   = VTab.MakeNew( flnm.AsFileName, dBase )
  storefld  = Field.Make ("DemRegion",#FIELD_CHAR,16, 0)
  directfld = Field.Make ("AlDrctCst",#FIELD_DECIMAL,16,2)
  pickfld   = Field.Make ("AlPickCst",#FIELD_DECIMAL,16,2)
  transfld  = Field.Make ("AlTranCst",#FIELD_DECIMAL,16,2)
  xdocfld   = Field.Make ("AlXdocCst",#FIELD_DECIMAL,16,2)
  totfld    = Field.Make ("AlTotlCst",#FIELD_DECIMAL,16,2)
  newFldList = {storefld, directfld,pickfld,
                transfld,xdocfld,totfld}
  newVTab.AddFields(newFldList)
  storefld.SetAlias("Demand Region")
  directfld.SetAlias("All Direct Cost")
  pickfld.SetAlias("All Pick Cost")
  transfld.SetAlias("All Tranship Cost")
  xdocfld.SetAlias("All Crossdock Cost")
  totfld.SetAlias("All Total Cost")

  ' Get the VTabs of the tables to be summed

  rxVTab = rxTable.GetVTab
  cwVTab = cwTable.GetVTab
  otcVTab = otcTable.GetVTab

  ' Get the count of one of them, since they should
' be same size and set for zero-based indexing

count = rxVTab.GetNumRecords
count = (count - 1).SetFormat("d")


' Get their bitmaps

rxBitMap = rxVTab.GetSelection
cwBitMap = cwVTab.GetSelection
otcBitMap = otcVTab.GetSelection


' Clear each bitmap

rxBitMap.ClearAll
cwBitMap.ClearAll
otcBitMap.ClearAll

'Create field lists for each production category

rxList = {"Rx Direct Cost", "Rx Pick Cost",
          "Rx Tranship Cost",
          "Rx Crossdock Cost", "Rx Total Cost"}

cwList = {"CW Direct Cost", "CW Pick Cost",
          "CW Tranship Cost",
          "CW Crossdock Cost", "CW Total Cost"}

otcList = {"OTC Direct Cost", "OTC Pick Cost",
           "OTC Tranship Cost",
           "OTC Crossdock Cost", "OTC Total Cost"}

' Remove the Demand Region name fld from the
' newVTab field list

newFldList.Remove(0)

listCount = rxList.Count
listCount = (listCount - 1).SetFormat("d")

' Get the name field to be added to each new
' record of the newVTab

nameFld = rxVTab.FindField("Demand Region")
if (nameFld = nil) then
  MsgBox.Error("Could not find name field from tables."+nl+
               "Quitting program...", "ERROR")
  exit
end

' Loop through each record of each bitmap and for each
' field, grab the value from each table, sum them, and
' add that new value to the All Logistics Costs table.

av.ShowStopButton
```

```
    for each rec in 0..count
      'Make sure each bitmap is clear
      rxBitMap.ClearAll
      cwBitMap.ClearAll
      otcBitMap.ClearAll
      'Set the same record for each bitmap
      rxBitMap.Set(rec)
      cwBitMap.Set(rec)
      otcBitMap.Set(rec)

      ' Add a new record to the All Logistics Cost Table
      ' and add the name to the name field

      newRec = newVTab.AddRecord
      regionName = rxVTab.ReturnValue(nameFld, rec)
      newVTab.SetValue(storefld, newRec, regionName)
      ' Set control for double loop in case fields not found

      stopLoop = FALSE

      'Loop through the field lists and grab the values
      'from each table, sum them, and copy the new value
      'to the All Logistics Cost table
      for each fld in 0..listCount
        rxFld = rxVTab.FindField(rxList.Get(fld))
        cwFld = cwVTab.FindField(cwList.Get(fld))
        otcFld = otcVTab.FindField(otcList.Get(fld))
        if ((rxFld = Nil) OR (cwFld = Nil) OR (otcFld = Nil)) then
          stopLoop = TRUE
          break
        end
        ' Get the field values from tables
        rxVal = rxVTab.ReturnValue(rxFld, rec)
        cwVal = cwVTab.ReturnValue(cwFld, rec)
        otcVal = otcVTab.ReturnValue(otcFld, rec)
        ' Sum them
        totVal = rxVal + cwVal + otcVal
        ' Add the new value to the new table
        newVTab.SetValue(newFldList.Get(fld), newRec, totVal)
      end ' inner for loop
      if (stopLoop) then
        MsgBox.Error("A field in Rx, CW, or OTC table not found."
                     +nl+"Quitting the building of All Logistics
                     Costs table.","ERROR")
        exit
      end
      rxBitMap.Clear(rec)
      cwBitMap.Clear(rec)
      otcBitMap.Clear(rec)
      rxVTab.UpdateSelection
      cwVTab.UpdateSelection
      otcVTab.UpdateSelection
    end 'outer for loop
  end 'if skip = 0

  ' Set the new table to uneditable and write to file
```

```
newVTab.SetEditable(FALSE)
newVTab.Flush

checkTable = av.GetProject.FindDoc("All Logistics Costs")
if (checkTable <> Nil) then
  av.GetProject.RemoveDoc(checkTable)
end

' Bring the new table into the project

newTable = Table.Make(newVTab)
newTable.SetName("All Logistics Costs")
av.GetProject.AddDoc(newTable)

' Join the newTable to the Demand Regions table

theStoreVTab.Join(thestorefld, newVTab, storefld)
MsgBox.Info("Tracing of all-products logistics cost for"+nl+
            "each demand region is complete.",
            "Trace Costs: Chain-wide All-Products")

return Nil
```

```
'*****************************************************************
' Scriptname:    CalcDCs

' Filename:      calcdcs.ave

' Description:   Script to generate the cross doc and pick info

' Requires:      Nil

' Called By:     Menu item click event ("Sum DCs")

' Calls:         HasCWRx, SummTS, SummD2S, JoinSumms

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "CalsDCs"

'Get the Demand by Region view and the CVS DCs FTab

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("CVS DCs")
theFTab = theTheme.GetFTab
editstate = theFTab.IsEditable
if (editstate.Not) then
  theFTab.SetEditable(true)
end


' Create fields Has Rx and Has CW, from Pick table

av.Run("HasCWRx", {theView, theFTab})

' Summarize table transship.dbf for CW and Rx for origin and dest

av.Run("SummTS", Nil)

' Summarize table dirstore.dbf for all three categories
' (Rx, OTC, CW)

av.Run("SummD2S", Nil)

' Join all of the summaries to the DCs theme table
```

```
av.Run("JoinSumms", {theView, theFTab})

'Need to create total picked for OTC = summarized value
'Need to create total picked for CW = sum on transship for
'origin + Has CW * sum on direct to store
'Need to create total picked for Rx = sum on transship for
'origin + Has Rx * sum on direct to store
'Need to create total cross docked for CW = sum transship
'for dest
'Need to create total cross docked for Rx = sum transship
'for dest
'Need to create handling costs at warehouse = handling
'rate * total picked for CW
'Need to create cross dock costs at warehouse = handling
'rate * totol cross dock.


' First tally the total CW and RX picked

rx_exists = (theFTab.FindField("Rx Picked") = NIL).Not
cw_exists = (theFTab.FindField("CW Picked") = NIL).Not
recalc = 1
if (rx_exists or cw_exists) then
  if (MsgBox.YesNo("Overwrite existing fields?",
  "The Rx and CW Picked fields already exist", false)) then
    'if ok to overwrite, delete the fields as they may not
    'be defined as required by this script (eg., created from
    'another script).
    if (rx_exists) then
      theFTab.RemoveFields({theFTab.FindField("Rx Picked")})
    end
    if (cw_exists) then
      theFTab.RemoveFields({theFTab.FindField("CW Picked")})
    end
  else
    recalc = 0
  end  'if (MsgBox...)
end  'if
if (recalc = 1) then
  rx = Field.Make ("Rx Picked",#FIELD_DECIMAL,16, 2)
  cw = Field.Make ("CW Picked",#FIELD_DECIMAL,16, 2)
  theFTab.AddFields({rx,cw})
  hasrx = theFTab.FindField("HasRx")
  hascw = theFTab.FindField("HasCW")
  rxts  = theFTab.FindField("RxPicked for TS")
  rxd2s = theFTab.FindField("Rx D2S")
  cwts  = theFTab.FindField("CWPicked for TS")
  cwd2s = theFTab.FindField("CW D2S")
  for each i in theFTab
    hasrxval = theFTab.ReturnValue(hasrx, i)
    hascwval = theFTab.ReturnValue(hascw, i)
    if (hasrxval = 0) then
      theval = 0
    else
      theval = theFTab.ReturnValue(rxts, i) +
               theFTab.ReturnValue(rxd2s, i)
    end
    theFTab.SetValue(rx, i, theval)
    if (hascwval = 0) then
      theval = 0
    else
      theval = theFtab.ReturnValue(cwts, i) +
               theFTab.ReturnValue(cwd2s, i)
    end
    theFTab.SetValue(cw, i, theval)
  end
end

rx_exists = (theFTab.FindField("Rx Rate") = NIL).Not
cw_exists = (theFTab.FindField("CW Rate") = NIL).Not
otc_exists = (theFtab.FindField("OTC Rate") = NIL).Not
recalc = 1
if (rx_exists or cw_exists or otc_exists) then
  if (MsgBox.YesNo("Overwrite existing fields?",
  "The Handling Rate fields already exist", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (rx_exists) then
      theFTab.RemoveFields({theFTab.FindField("Rx Rate")})
    end
    if (cw_exists) then
      theFTab.RemoveFields({theFTab.FindField("CW Rate")})
    end
    if (otc_exists) then
      theFTab.RemoveFields({theFTab.FindField("OTC Rate")})
    end
  else
    recalc = 0
    'exit
  end  'if (MsgBox...)
end  'if
if (recalc = 1) then
  rx = Field.Make ("Rx Rate",#FIELD_DECIMAL,9, 6)
  cw = Field.Make ("CW Rate",#FIELD_DECIMAL,9, 6)
  otc = Field.Make("OTC Rate", #FIELD_DECIMAL, 9, 6)
  theFTab.AddFields({rx,cw, otc})

  ' Get the handling table

  theDCFld = theFTab.FindField("Facility")
  theRateTab = av.FindDoc("handling.dbf").GetVTab
  theFacFld = theRateTab.FindField("Facility")
  theProdFld = theRateTab.FindField("Product")
  theRateFld = theRateTab.FindField("HandlingRate")
  for each i in theFTab
    found = 0
    theDC = theFTab.ReturnValue(theDCFld, i)
    for each j in theRateTab
      if (found = 3) then
        break
      end
      theFacility = theRateTab.ReturnValue(theFacFld, j)
      if (theDC <> theFacility) then
        continue
      end
      theProduct = theRateTab.ReturnValue(theProdFld, j)
      if (theProduct = "CW") then
        theRate = theRateTab.ReturnValue(theRateFld, j)
        theFTab.SetValue(cw, i, theRate)
        found  = found+1
        continue
      end
      if (theProduct = "Rx") then
        theRate = theRateTab.ReturnValue(theRateFld, j)
        theFTab.SetValue(rx, i, theRate)
        found = found+1
        continue
      end
      if (theProduct = "OTC") then
        theRate = theRateTab.ReturnValue(theRateFld, j)
        theFTab.SetValue(otc, i, theRate)
        found = found + 1
        continue
      end 'end if
    end 'end for on j
  end 'end for on i
end

' Now add the cost fields

rxpcost_exists = (theFTab.FindField("Rx Pick Cost")= NIL).Not
rxxdoccost_exists = (theFTab.FindField("Rx X Doc Cost") = NIL).Not
cwpcost_exists = (theFTab.FindField("CW Pick Cost") = NIL).Not
cwxdoccost_exists =(theFTab.FindField("CW X Doc Cost") = NIL).Not
otcpcost_exists = (theFTab.FindField("OTC Pick Cost") = NIL).Not
totcost_exists = (theFTab.FindField("Total Handling") = NIL).Not
recalc = 1
if (rxpcost_exists or rxxdoccost_exists or cwpcost_exists or
  cwxdoccost_exists or otcpcost_exists or totcost_exists) then
  if (MsgBox.YesNo("Overwrite existing fields?",
  "The Handling Cost fields already exist", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (rxpcost_exists) then
      theFTab.RemoveFields({theFTab.FindField("Rx Pick Cost")})
    end
    if (rxxdoccost_exists) then
      theFTab.RemoveFields({theFTab.FindField("Rx X Doc Cost")})
    end
    if (cwpcost_exists) then
      theFTab.RemoveFields({theFTab.FindField("CW Pick Cost")})
    end
    if (cwxdoccost_exists) then
      theFTab.RemoveFields({theFTab.FindField("CW X Doc Cost")})
      end
      if (otcpcost_exists) then
        theFTab.RemoveFields({theFTab.FindField("OTC Pick Cost")})
      end
      if (totcost_exists) then
        theFTab.RemoveFields({theFTab.FindField("Total Handling")})
      end
    else
      recalc = 0
    end  'if (MsgBox...)
  end  'if
  if (recalc = 1) then
    rxx = Field.Make ("Rx X Doc Cost",#FIELD_DECIMAL,16, 2)
    rxd = Field.Make ("Rx Pick Cost",#FIELD_DECIMAL,16, 2)
    cwx = Field.Make ("CW X Doc Cost",#FIELD_DECIMAL,16, 2)
    cwd = Field.Make ("CW Pick Cost",#FIELD_DECIMAL,16, 2)
    otc = Field.Make ("OTC Pick Cost", #FIELD_DECIMAL, 16, 2)
    tot = Field.Make ("Total Handling",#FIELD_DECIMAL,16, 2)
    theFTab.AddFields({rxx,rxd,cwx,cwd,otc,tot})


  ' There may be blank entries because of no matches on a join.
  ' We cannot change the values of those entries because they belong
  ' to another table.

    theBitMap = theFTab.GetSelection

    expr = "(([RX_X_Doc].IsNull.Not) and ([Rx Rate].IsNull.Not))"
    theFTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    theval = "[RX_X_Doc] * [RX Rate]"
    theFTab.Calculate(theval,rxx)

    expr = "(([RX Picked].IsNull.Not) and ([Rx Rate].IsNull.Not))"
    theFTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    theval = "[RX Picked] * [RX Rate]"
    theFTab.Calculate(theval,rxd)

    expr = "(([CW_X_Doc].IsNull.Not) and ([CW Rate].IsNull.Not))"
    theFTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    theval = "[CW_X_Doc] * [CW Rate]"
    theFTab.Calculate(theval,cwx)

    expr = "(([CW Picked].IsNull.Not) and ([OTC Rate].IsNull.Not))"
    theFTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    theval = "[CW Picked] * [OTC Rate]"
    theFTab.Calculate(theval,cwd)

    expr = "(([OTC Picked].IsNull.Not) and ([OTC Rate].IsNull.Not))"
    theFTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    theval = "[OTC Picked] * [OTC Rate]"
    theFTab.Calculate(theval,otc)
    theFTab.GetSelection.ClearAll

    theval = "[Rx X Doc Cost] + [Rx Pick Cost] + [CW X Doc Cost]
             + [CW Pick Cost] + [OTC Pick Cost]"
    theFTab.Calculate(theval,tot)
```

```
end

' Reset edit state to false

theFTab.SetEditable(false)

MsgBox.Info("Summary of DC Data Complete", "NOTICE")

return Nil
```

```
'****************************************************************
'  Scriptname:  ColorPalette.SelectColor

'  File Name:   colorpal.ave

'  Author:      Kenneth Bennett

'  Date:        February 10, 1998

'  Description: Makes a palette from ArcView's default
'               color palette and then asks a user to select
'               a color.  AV's default.avp file must be in the
'               working directory for this script to work.

'  Requires:    Nil

'  Called by:   FlowLines.Build

'  Calls:       Nil

'  SELF:        Nil

'  Return:      a Color object

'****************************************************************
Scriptname = "ColorPalette.SelectColor"

' Check to see if the default AVP file is
' in the project working directory

theWorkDirString = av.GetProject.GetWorkDir.AsString
theDefPalFile = (theWorkDirString + "\Default.avp").AsFileName
exists = File.Exists(theDefPalFile)
if (exists.not) then
  MsgBox.Error("ArcView's default palette is not in"+NL+
               "the project's working directory."+NL+
               "Please load file Default.avp into working"+NL+
               "directory and rerun script.", Scriptname)
  exit
end

' Create a palette using ArcView's default palette file

thePalette = Palette.MakeFromFile(theDefPalFile)

' Create a color list that corresponds index-wise to the symbollist
' of the Color Palette

theColorList = {"Transparent", "White", "Light Grey", "Medium
               Grey", "Dark Grey", "Black", "Light Pink", "Dark
               Pink", "Candy Red", "Red", "Lt Red-Brown", "Dark
               Red-Brown", "Lt Pastel Green", "Dk Pastel Green",
               "Lt Fluorescent Green", "Dk Fluorescent Green",
               "Green", "Dark Green", "Lt Pastel Purple",
               "Dk Pastel Purple", "Blue", "Dark Blue", "Navy
               Blue", "Metallic Blue", "Lt Pastel Blue",
                "Dk Pastel Blue", "Lt Sky Blue", "Dk Sky Blue",
                "Lt Ocean Blue", "Dk Ocean Blue", "Lt Pastel Mauve",
                "Dk Pastel Mauve", "Lt Fluorescent Purple",
                "Dk Fluorescent Purple", "Purple", "Dark Purple",
                "Lt Pastel Yellow", "Dk Pastel Yellow", "Yellow",
                "Mustard", "Lt Olive Green", "Dk Olive Green",
                "Peach", "Lt Orange", "Dk Orange", "Lt Fuchsia",
                "Dk Fuchsia", "Pea Green", "Desert Green",
                "Lt Stone Grey", "Medium Purple", "Dk Stone Grey",
                "Brown", "Chocolate", "Fluorescent Lime Green",
                "Turquois", "Light Blue", "Medium Blue", "Light
                Purple", "Blue-Purple"}

' Have the user select a color from this list

theListSel = MsgBox.ListAsString(theColorList,
                                 "Select A Color:", Scriptname)
if (theListSel = Nil) then
  MsgBox.Error("No color was selected.  Exiting...", Scriptname)
  exit
end

' Get the index number for that color string in theColorList

index = theColorList.Find(theListSel)

' Use that index to grab the corresponding color in Color Palette

theColorPaletteList = thePalette.GetList(#PALETTE_LIST_COLOR)
'MsgBox.Info(theColorPaletteList.Count.AsString, Scriptname)
chosenColor = theColorPaletteList.Get(index)

' Return the chosen color object

Return chosenColor
```

```
'*****************************************************************

' Scriptname:    CWFlowTheme.Make

' Filename:      cwflowth.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a Flow theme based on the CW
'                Flow field in the DC-to-Region Flow theme table.
'                Zero value flows are made invisible using null
'                value and symbol.

' Requires:      DC-to-Region flow theme must exist

' Called by:     View menu item click event
'                ("Display Flows: DC-to-Region by CW Only")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "CWFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("DC-to-Region Flow")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called DC-to-Region Flow
              does not exist.", Scriptname)
  exit
end

catString = "CW Flow"

checkTheme = theView.FindTheme(catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of"++catString)
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the DC-to-Region Flow theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend
newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)

' Make zero the null value

newLegend.SetNullValue(catString, 0)

' Select a color from the color palette to be used
' in drawing the new line theme

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with into five natural breaks
' and weight the line thickness by the flow volume

newLegend.Natural(newTheme, catString, 5)
theSymbolList = newLegend.GetSymbols
thickness = 1
count = 0
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
  end
theSymbolList.UniformColor(theColor)

' Make the null symbol transparent

nullSym = Symbol.Make(#SYMBOL_PEN )
theNullColor = Color.GetBlue
theNullColor.SetTransparent(TRUE)
nullSym.SetColor(theNullColor)
newLegend.SetNullSymbol(nullSym)
newLegend.DisplayNoDataClass(FALSE)
newTheme.SetLegend(newLegend)
newTheme.SetName (catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme (newTheme)
newTheme.UpdateLegend
theView.Invalidate
theNullColor.SetTransparent(FALSE)

return Nil
```

```
'*************************************************************

' Scriptname:      CWStatistics.Generate

' Filename:        cwstatis.ave

' Author:          Kenneth Bennett

' Date:            May 6, 1998

' Description:     Script sums each of the fields in the
'                  CW Logistics Costs table and reports
'                  it to the user.

' Requires:        CW Logistics Costs table exists

' Called by:       View menu item click
'                  ("Trace Costs: Chain-wide CW Statistics")

' Calls:           Nil

' SELF:            Nil

' Returns:         Nil

'*************************************************************
Scriptname = "CWStatistics.Generate"

' Ensure two decimal places in each number

Script.The.SetNumberFormat("d.dd")

' Get the table

theTable = av.GetProject.FindDoc("CW Logistics Costs")
if (theTable = Nil) then
  MsgBox.Error("CW Logistics Costs table not found.",
               Scriptname)
  exit
end

' Get the VTab for the table

theVTab = theTable.GetVTab

' Get the number of records

num = theVTab.GetNumRecords

' Get the list of fields in the VTab

theFieldList = theVTab.GetFields

' Set the field variables

shipFld = theFieldList.Get(1)
```

```
pickFld = theFieldList.Get(2)
tranFld = theFieldList.Get(3)
xdocFld = theFieldList.Get(4)
totlFld = theFieldList.Get(5)

' Initialize summing variables

shipSum = 0
pickSum = 0
tranSum = 0
xdocSum = 0
totlSum = 0

' Loop through the VTab and sum each fld

for each rec in theVTab
  shipSum = shipSum + theVTab.ReturnValue(shipFld, rec)
  pickSum = pickSum + theVtab.ReturnValue(pickFld, rec)
  tranSum = tranSum + theVTab.ReturnValue(tranFld, rec)
  xdocSum = xdocSum + theVTab.ReturnValue(xdocFld, rec)
  totlSum = totlSum + theVTab.ReturnValue(totlFld, rec)
end

' Calculate the per store average for each field

shipAvg = shipSum / num
pickAvg = pickSum / num
tranAvg = tranSum / num
xdocAvg = xdocSum / num
totlAvg = totlSum / num

' Issue the report

reportString = "Total CW Shipping Cost:"++shipSum.AsString+nl+
               "Average Per Region:"++shipAvg.AsString+nl+
               "Total CW Pick Cost:"++pickSum.AsString+nl+
               "Average Per Region:"++pickAvg.AsString+nl+
               "Total CW Transhipment Cost:"++tranSum.AsString
               +nl+"Average Per Region:"++tranAvg.AsString+nl+
               "Total CW Crossdock Cost:"++xdocSum.AsString+nl+
               "Average Per Region:"++xdocAvg.AsString+nl+nl+
               "Chain-wide Total CW Logistics Cost:"++
               totlSum.AsString+nl+"Average Total Cost Per
               Region:"++totlAvg.AsString+nl

MsgBox.Report(reportString, "Chain-wide CW Statistics")

return Nil
```

```
'*****************************************************************

' Scriptname:   DC.HistPct

' Filename:     dc_histp.ave

' Description:  Generates a histogram for the DC Range field in
'               the stores theme table.  A new Chart document
'               is created to display the histogram.  A temporary
'               file is created to store interval counts and
'               other information used to create the histogram.

' Requires:     View is the active document and stores theme is
'               the active theme.  DC Range field has been
'               added to the stores theme table and has been
'               populated using the DC-Store Range menu click
'               event under the Location Strategy menu set.

' Called by:    Menu click event ("Make Histogram")

' Calls:        Nil
'
' SELF:         Nil
'
' Returns:      Nil

'*****************************************************************
Scriptname = "DC.HistPct"

' Get the view and the stores theme, and verify

theView = av.GetProject.FindDoc("Location Strategy")
if (theView = Nil) then
  MsgBox.Error("ERROR: Location Strategy view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  msgBox.Error("Selected document is not a view.",Scriptname)
  exit
end

theThemeList = theView.GetActiveThemes
thePointThemeList = {}
storeTheme = Nil
for each t in theThemeList
  if (t.GetFTab.GetSrcName.GetSubName = "Point") then
    thePointThemeList.Add(t)
   end
end

numThemes = thePointThemeList.Count
if (numThemes = 1) then
  storeTheme = thePointThemeList.Get(0)
elseif (numThemes > 1) then
  storeTheme = MsgBox.List(thePointThemeList,
                           "Select a stores theme:", Scriptname)
else
  MsgBox.Error("Store theme not selected.  Exiting...",
               Scriptname)
  exit
end

' check if a store theme was selected

if (storeTheme = nil) then
  msgBox.Error("Store theme not selected.  Exiting...",
               Scriptname)
  exit
end

storeFTab = storeTheme.GetFTab

'  Verify that the FTab has the required DC Range field

theField = storeFTab.FindField("DC Range")
if (theField = Nil) then
  MsgBox.Error("DC Range field not found.  Make sure that the"
               +NL+"correct store theme has been selected and
               that"+NL+"the DC Range field has been added and
               calculated.", Scriptname)
  exit
end

' Set the number of intervals to six

numIntervals = 6

' Create a temporary storeFTabab to hold interval counts

histoFilePath = "c:\cvs\cvsac\thesis\charts".AsFileName
histoFilePath.SetCWD
histVTab    = VTab.MakeNew(Filename.GetCWD.MakeTmp
                           ("histo","dbf"), dBASE)
labelField = Field.Make("Range", #FIELD_CHAR, 28, 0)
countField = Field.Make("Count", #FIELD_LONG, 12, 0)
percentField = Field.Make("Percent", #FIELD_DECIMAL, 6, 2)
histVTab.AddFields( {labelField, countField, percentField} )


numReads = 0

' Determine number of records to process

if (storeFTab.GetSelection.Count = 0) then
  iter = storeFTab
  n = storeFTab.GetNumRecords
else
  iter = storeFTab.GetSelection
  n = iter.Count
end
maxNumReads = n
```

```
' Set the minimum and maximum values and the interval size

minimum = 0
maximum = 251
intervalSize = 50


' Populate the histogram VTab with interval labels,
' initializing interval counts to 0

oldlow = -1
for each i in 0..(numIntervals - 1)
  rec = histVTab.AddRecord
  low = oldlow + 1
  high = (minimum + ((i + 1) * intervalSize))
  if (i = 0) then
    histVTab.SetValueString(labelField, rec,
              ("within"++intervalSize.AsString++"miles"))
  elseif (i = (numIntervals - 1)) then
    histVTab.SetValueString(labelField, rec,
              (">"++(low - 1).AsString++"miles"))
  else
    histVTab.SetValueString(labelField, rec,
              (low.AsString++" - "++high.AsString++"miles"))
  end
  oldlow = high
  histVTab.SetValueNumber(countField, rec, 0)
  histVTab.SetValueNumber(percentField, rec, 0)
end ' for loop


' Loop through records again, incrementing
' the appropriate counter based on the
' interval in which the value falls

for each rec in iter
  numReads = numReads + 1
  av.SetStatus(numReads / maxNumReads * 100)
  curval = storeFTab.ReturnValueNumber(theField, rec)
  index = ((curval - minimum) / intervalSize) - 1
  if (index = -1) then
     MsgBox.Info("ERROR:  Record(s) still have DC Range
                 value of zero!", Scriptname)
     exit
  end
  if ((index mod 1) <> 0) then 'curval is 251
         (ie, not a multiple of 50), assign to last interval
    index = index.Ceiling
  end
  if (index = numIntervals) then
    index = numIntervals - 1
  end
  histVTab.SetValueNumber(countField, index,
                        histVTab.ReturnValueNumber
```

```
                          (countField, index) + 1)
end 'for loop

' Loop through histVTab and calculate the percent field

for each j in histVTab
  cnt = histVTab.ReturnValue(countField, j)
  p = ( cnt / maxNumReads ) * 100
  histVTab.SetValueNumber(percentField, j, p)
end 'for loop


' Make a Chart document and display it

newChart = Chart.make(histVTab, {percentField})
newChart.SetRecordLabelField(labelField)
storeName = storeTheme.GetName.AsTokens(". ").Get(0)
newChart.GetTitle.SetName("Percent of"++storeName++
                          "Stores to Nearest DC")
newChart.SetName("Histogram of"++storeName++
                   "Store-DC Distances")
xax = newChart.GetXAxis
yax = newChart.GetYAxis
xax.SetName("50 Mile Intervals")
yax.SetName("Percent of Stores")
xax.SetLabelVisible(true)
yax.SetLabelVisible(true)
yax.SetBoundsMin(0)
yax.SetBoundsMax(100)
yax.SetBoundsUsed(true)
yax.SetMajorGridSpacing(10)
yax.SetMajorGridVisible(false)
yax.SetMinorGridSpacing(5)
yax.SetMinorGridVisible(false)
yax.SetTickLabelsVisible(true)
av.GetProject.AddDoc(newChart)
newChart.GetWin.Open

Return Nil
```

```
'**************************************************************

' Scriptname:  DC.Range

' Filename:    dc_range.ave

' Description: Script grabs the selected records of the
'              stores theme and populates the DC Range field
'              selected records with the service range
'              distance.  This script is used to set up the
'              stores attribute tables for charting.

' Requires:    Service area theme has been created and stores
'              with selected ranges have been selected.

' Called by:   Stores.SelByRange

' Calls:       Nil

' SELF:        the stores FTab, the range field, and
'              the range name

' Returns:     Nil

'***************************************************************
Scriptname = "DC.Range"
theFTab = SELF.Get(0)
dcrange = SELF.Get(1)
rangeName = SELF.Get(2)
storeTheme = SELF.Get(3)

' Get the select set of the stores theme

theSelSet = theFTab.GetSelection
dcrange = theFTab.FindField("DC Range")

av.ShowStopButton
av.ShowMsg("Editing DC Range field...")
numSelRecs = theFTab.GetNumSelRecords


n = 0

edit_state = theFTab.IsEditable
theFTab.SetEditable(true)

' Loop through the selected records and calculate
' the selected records to the range Name if it has
' not already been done so.

for each rec in theSelSet
   rangeval = theFTab.ReturnValue(dcrange, rec)
   if (rangeval = 0) then
     theFTab.SetValue(dcrange, rec, rangeName)
     n = n + 1
   else
```

```
    n = n + 1
    continue
  end 'if

  progress = (n/numSelRecs) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
    break
  end 'if

end 'for loop

theFTab.SetEditable(FALSE)
theFTab.SetEditable(edit_state)
storeTheme.ClearSelection

return Nil
```

```
'*****************************************************************

' Scriptname:    DCFixedCostTheme.Make

' Filename:      dcfixedc.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a CVS DCs theme based
'                on the Fixed Cost field in the CVS DCs theme
'                table.  The theme is classified into three sizes
'                based on fixed cost value and uses the triangle
'                as the symbol

' Requires:      CVS DCs theme must exist

' Called by:     View menu item click event
'                ("Display DC: by Fixed Cost")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "DCFixedCostTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theTheme = theView.FindTheme("CVS DCs")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called CVS DCs does not exist.",
               Scriptname)
  exit
end

checkTheme = theView.FindTheme("CVS DCs by Fixed Cost")
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of CVS
                                   DCs by Fixed Cost")
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
```

```
end

' Clone the CVS DCs theme

fixTheme = theTheme.Clone
fixTheme.SetName("CVS DCs by Fixed Cost")

' Change the legend to weight the symbol size
' by the fixed cost  and classify into 3
' groups using a  natural break

fixLegend = fixTheme.GetLegend
fixLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
fixLegend.Natural(fixTheme, "FixedCost", 3)
fixLegend.DisplayNoDataClass(FALSE)
' Get the project working directory
theDir = av.GetProject.GetWorkDir.AsString
thePath = theDir+"\default.avp"
theSymbolList = fixLegend.GetSymbols
index = 0
increment = 0
for each s in theSymbolList
  thePalette = Palette.MakeFromFile(thePath.AsFileName)
  ' Grab the Marker palette and get the outlined
  ' triangle symbol, which is the 9th symbol in the palette
  chosenMarker = thePalette.GetList
                          (#PALETTE_LIST_MARKER).Get(9)
  chosenMarker.SetSize(12 + increment)
  theSymbolList.Set(index, chosenMarker)
  index = index + 1
  increment = increment + 4
end
theSymbolList.UniformColor(Color.GetYellow)

fixTheme.UpdateLegend
fixTheme.SetVisible( TRUE )
theView.AddTheme (fixTheme)
theView.Invalidate

return Nil
```

```
'****************************************************************

' Scriptname:    DCHandCostTheme.Make

' Filename:      dchandco.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a CVS DCs theme based on
'                the total Handling field in the CVS DCs theme
'                table.  The theme is classified into three sizes
'                based on total handling cost and uses the hexagon
'                as the symbol

' Requires:      CVS DCs theme must exist

' Called by:     View menu item click event
'                ("Display DC: by Handling Cost")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'****************************************************************
Scriptname = "DCHandCostTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theTheme = theView.FindTheme("CVS DCs")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called CVS DCs does not exist.",
               Scriptname)
  exit
end

checkTheme = theView.FindTheme("CVS DCs by Handling Cost")
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of CVS
                                    DCs by Handling Cost")
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
```

```
end

' Clone the CVS DCs theme

handTheme = theTheme.Clone
handTheme.SetName("CVS DCs by Handling Cost")

' Change the legend to weight the symbol size
' by the total handling cost  and classify into 3
' groups using a  natural break

handLegend = handTheme.GetLegend
handLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
handLegend.Natural(handTheme, "Total Handling", 3)
handLegend.DisplayNoDataClass(FALSE)
' Get the project working directory
theDir = av.GetProject.GetWorkDir.AsString
thePath = theDir+"\default.avp"
theSymbolList = handLegend.GetSymbols
index = 0
increment = 0
for each s in theSymbolList
  thePalette = Palette.MakeFromFile(thePath.AsFileName)
  ' Grab the Marker palette and get the outlined
  ' hexagon symbol, which is the 11th symbol in the palette
  chosenMarker = thePalette.GetList
                           (#PALETTE_LIST_MARKER).Get(11)
  chosenMarker.SetSize(12 + increment)
  theSymbolList.Set(index, chosenMarker)
  index = index + 1
  increment = increment + 4
end
theSymbolList.UniformColor(Color.GetYellow)

handTheme.UpdateLegend
handTheme.SetVisible( TRUE )
theView.AddTheme (handTheme)
theView.Invalidate

return Nil
```

```
'******************************************************************

' Scriptname:     DCTotDemandTheme.Make

' Filename:       dctotdem.ave

' Author:         Kenneth Bennett

' Date:           May 3, 1998

' Description:    Script generates a CVS DCs theme based on the
'                 OptimizedValue field in the CVS DCs theme table.
'                 The theme is classified into three sizes based
'                 on the optimized value (demand) and uses the
'                 square as the symbol

' Requires:       CVS DCs theme must exist

' Called by:      View menu item click event
                  ("Display DC: by Total Demand")

' Calls:          Nil

' SELF:           Nil

' Returns:        Nil

'******************************************************************
Scriptname = "DCTotDemandTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theTheme = theView.FindTheme("CVS DCs")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called CVS DCs does not exist.",
               Scriptname)
  exit
end

checkTheme = theView.FindTheme("CVS DCs by Total Demand")
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of CVS
                                  DCs by Total Demand")
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
```

```
end

' Clone the CVS DCs theme

demandTheme = theTheme.Clone
demandTheme.SetName("CVS DCs by Total Demand")

' Change the legend to weight the symbol size
' by the optimized value  and classify into 3
' groups using a  natural break

demandLegend = demandTheme.GetLegend
demandLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
demandLegend.Natural(demandTheme, "OptimizedValue", 3)
demandLegend.DisplayNoDataClass(FALSE)
' Get the project working directory
theDir = av.GetProject.GetWorkDir.AsString
thePath = theDir+"\default.avp"
theSymbolList = demandLegend.GetSymbols
index = 0
increment = 0
for each s in theSymbolList
  thePalette = Palette.MakeFromFile(thePath.AsFileName)
  ' Grab the Marker palette and get the outlined
  ' square symbol, which is the 8th symbol in the palette
  chosenMarker = thePalette.GetList
                          (#PALETTE_LIST_MARKER).Get(8)
  chosenMarker.SetSize(12 + increment)
  theSymbolList.Set(index, chosenMarker)
  index = index + 1
  increment = increment + 4
end
theSymbolList.UniformColor(Color.GetYellow)

demandTheme.UpdateLegend
demandTheme.SetVisible( TRUE )
theView.AddTheme (demandTheme)
theView.Invalidate

return Nil
```

```
'*************************************************************

' Scriptname:    DRAllLogTheme.Make

' Filename:      dralllog.ave

' Author:        Kenneth Bennett

' Date:          May 6, 1998

' Description:   Creates a pie chart theme of the Demand
'                Regions where the pie slices represent
'                logistics component costs for all products
'                and the size of the whole pie represents
'                the grand total logistics cost for each
'                demand region.

' Requires:      Demand Regions theme must exist

' Called by:     View menu item click event ("Display Demand
'                Regions: by Total Logistics Cost")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*************************************************************
Scriptname = "DRAllLogTheme.Make"

' Find the view and the Demand Regions theme

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theView.GetWin.Open
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Find the needed fields

shipFld = theFTab.FindField("All Direct Cost")
```

```
pickFld = theFTab.FindField("All Pick Cost")
tranFld = theFTab.FindField("All Tranship Cost")
xdocFld = theFTab.FindField("All Crossdock Cost")
totlFld = theFTab.FindField("All Total Cost")

if ((shipFld = Nil) OR (pickFld = Nil) OR (tranFld = Nil)
     OR (xdocFld = Nil) OR (totlFld = Nil)) then
  MsgBox.Error("ERROR: One or more required fields is missing"
             +nl+"or has been renamed.", Scriptname)
  exit
end

shipFld = shipFld.AsString
pickFld = pickFld.AsString
tranFld = tranFld.AsString
xdocFld = xdocFld.AsString
totlFld = totlFld.AsString

fldStringList = {shipFld, pickFld, tranFld, xdocFld}

' Check to see if the new theme already exists
checkTheme = theView.FindTheme("Demand Regions by Total
                                Logistics Cost")
if (checkTheme <> Nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Demand
                            Regions by Total Logistics Cost")
  if (theTable <> Nil) then
    av.GetProject.RemoveDoc(theTable)
  end
end


' Clone the theme and work with the new theme

demTheme = theTheme.Clone


' Get the new Demand Region theme's legend

demLegend = demTheme.GetLegend

'Create as many fill symbols as you have fieldNames
'and place them in a list.

shipsym = RasterFill.Make
shipsym.SetStyle(#RASTERFILL_STYLE_SOLID)
shipsym.SetColor(Color.GetRed)
picksym = RasterFill.Make
picksym.SetStyle(#RASTERFILL_STYLE_SOLID)
picksym.SetColor(Color.GetWhite)
transym = RasterFill.Make
transym.SetStyle(#RASTERFILL_STYLE_SOLID)
transym.SetColor(Color.GetBlack)
xdocsym= RasterFill.Make
xdocsym.SetStyle(#RASTERFILL_STYLE_SOLID)
xdocsym.SetColor(Color.GetGray)

theSyms = {shipsym, picksym, transym, xdocsym}

' Make a background fill Symbol that is empty

BGsym = RasterFill.Make
BGsym.SetStyle(#RASTERFILL_STYLE_EMPTY)

' Create the New Legend

demLegend.PieChart(demTheme,fldStringList,theSyms,BGSym,totlFld)

' To set a size field:

theSym = demLegend.GetSymbol(demLegend.ReturnFieldNames, false)

theSym.SetMinSize(8)
theSym.SetMaxSize(24)

' Redraw the theme using the PieChart legend.

demTheme.UpdateLegend
demTheme.SetActive(FALSE)
demTheme.SetVisible(TRUE)
demTheme.SetName("Demand Regions by Total Logistics Cost")
theView.AddTheme(demTheme)
theView.Invalidate

return Nil
```

```
'************************************************************

' Scriptname:     DRCWLogTheme.Make

' Filename:       drcwlogt.ave

' Author:         Kenneth Bennett

' Date:           May 6, 1998

' Description:    Creates a pie chart theme of the Demand
'                 Regions where the pie slices represent the
'                 CW logistics component costs and the size of the
'                 whole pie represents the total CW logistics cost.

' Requires:       Demand Regions theme must exist

' Called by:      View menu item click event ("Display Demand
'                 Regions: by CW Logistics Cost")

' Calls:          Nil

' SELF:           Nil

' Returns:        Nil

'************************************************************
Scriptname = "DRCWLogTheme.Make"

' Find the view and the Demand Regions theme

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theView.GetWin.Open
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Find the needed fields

shipFld = theFTab.FindField("CW Direct Cost")
pickFld = theFTab.FindField("CW Pick Cost")
tranFld = theFTab.FindField("CW Tranship Cost")
```

```
xdocFld = theFTab.FindField("CW Crossdock Cost")
totlFld = theFTab.FindField("CW Total Cost")

if ((shipFld = Nil) OR (pickFld = Nil) OR (tranFld = Nil)
     OR (xdocFld = Nil) OR (totlFld = Nil)) then
  MsgBox.Error("ERROR: One or more required fields is missing"+nl+
               "or has been renamed.", Scriptname)
  exit
end

shipFld = shipFld.AsString
pickFld = pickFld.AsString
tranFld = tranFld.AsString
xdocFld = xdocFld.AsString
totlFld = totlFld.AsString

fldStringList = {shipFld, pickFld, tranFld, xdocFld}

' Check to see if the new theme already exists
checkTheme = theView.FindTheme("Demand Regions by CW
                              Logistics Cost")
if (checkTheme <> Nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Demand
                              Regions by CW Logistics Cost")
  if (theTable <> Nil) then
    av.GetProject.RemoveDoc(theTable)
  end
end


' Clone the theme and work with the new theme

demTheme = theTheme.Clone


' Get the new Demand Region theme's legend

demLegend = demTheme.GetLegend

'Create as many fill symbols as you have fieldNames
'and place them in a list.

shipsym = RasterFill.Make
shipsym.SetStyle(#RASTERFILL_STYLE_SOLID)
shipsym.SetColor(Color.GetMagenta)
picksym = RasterFill.Make
picksym.SetStyle(#RASTERFILL_STYLE_SOLID)
picksym.SetColor(Color.GetCyan)
transym = RasterFill.Make
transym.SetStyle(#RASTERFILL_STYLE_SOLID)
transym.SetColor(Color.GetGreen)
xdocsym= RasterFill.Make
xdocsym.SetStyle(#RASTERFILL_STYLE_SOLID)
xdocsym.SetColor(Color.GetWhite)
```

```
theSyms = {shipsym, picksym, transym, xdocsym}

' Make a background fill Symbol that is empty

BGsym = RasterFill.Make
BGsym.SetStyle(#RASTERFILL_STYLE_EMPTY)

' Create the New Legend

demLegend.PieChart(demTheme,fldStringList,theSyms,
                   BGSym,totlFld)

' To set a size field:

theSym = demLegend.GetSymbol(demLegend.ReturnFieldNames,
                             false)

theSym.SetMinSize(8)
theSym.SetMaxSize(24)

' Redraw the theme using the PieChart legend.

demTheme.UpdateLegend
demTheme.SetActive(FALSE)
demTheme.SetVisible(TRUE)
demTheme.SetName("Demand Regions by CW Logistics Cost")
theView.AddTheme(demTheme)
theView.Invalidate

return Nil
```

```
'***************************************************************

' Scriptname:    DROTCLogTheme.Make

' Filename:      drotclog.ave

' Author:        Kenneth Bennett

' Date:          May 6, 1998

' Description:   Creates a pie chart theme of the Demand
'                Regions where the pie slices represent the
'                OTC logistics component costs and the size
'                of the whole pie represents the total OTC
'                logistics cost.

' Requires:      Demand Regions theme must exist

' Called by:     View menu item click event ("Display Demand
'                Regions: by OTC Logistics Cost")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'***************************************************************
Scriptname = "DROTCLogTheme.Make"

' Find the view and the Demand Regions theme

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theView.GetWin.Open
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Find the needed fields

shipFld = theFTab.FindField("OTC Direct Cost")
pickFld = theFTab.FindField("OTC Pick Cost")
```

```
tranFld = theFTab.FindField("OTC Tranship Cost")
xdocFld = theFTab.FindField("OTC Crossdock Cost")
totlFld = theFTab.FindField("OTC Total Cost")

if ((shipFld = Nil) OR (pickFld = Nil) OR (tranFld = Nil)
     OR (xdocFld = Nil) OR (totlFld = Nil)) then
  MsgBox.Error("ERROR: One or more required fields
               is missing"+nl+"or has been renamed.",
               Scriptname)
  exit
end

shipFld = shipFld.AsString
pickFld = pickFld.AsString
tranFld = tranFld.AsString
xdocFld = xdocFld.AsString
totlFld = totlFld.AsString

fldStringList = {shipFld, pickFld, tranFld, xdocFld}

' Check to see if the new theme already exists
checkTheme = theView.FindTheme("Demand Regions by
                               OTC Logistics Cost")
if (checkTheme <> Nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Demand
                              Regions by OTC Logistics Cost")
  if (theTable <> Nil) then
    av.GetProject.RemoveDoc(theTable)
  end
end


' Clone the theme and work with the new theme

demTheme = theTheme.Clone


' Get the new Demand Region theme's legend

demLegend = demTheme.GetLegend

'Create as many fill symbols as you have
'fieldNames and place them in a list.

shipsym = RasterFill.Make
shipsym.SetStyle(#RASTERFILL_STYLE_SOLID)
shipsym.SetColor(Color.GetGreen)
picksym = RasterFill.Make
picksym.SetStyle(#RASTERFILL_STYLE_SOLID)
picksym.SetColor(Color.GetRed)
transym = RasterFill.Make
transym.SetStyle(#RASTERFILL_STYLE_SOLID)
transym.SetColor(Color.GetWhite)
xdocsym= RasterFill.Make
xdocsym.SetStyle(#RASTERFILL_STYLE_SOLID)
xdocsym.SetColor(Color.GetBlack)

theSyms = {shipsym, picksym, transym, xdocsym}

' Make a background fill Symbol that is empty

BGsym = RasterFill.Make
BGsym.SetStyle(#RASTERFILL_STYLE_EMPTY)

' Create the New Legend

demLegend.PieChart(demTheme,fldStringList,theSyms,BGSym,totlFld)

' To set a size field:

theSym = demLegend.GetSymbol(demLegend.ReturnFieldNames, false)

theSym.SetMinSize(8)
theSym.SetMaxSize(24)

' Redraw the theme using the PieChart legend.

demTheme.UpdateLegend
demTheme.SetActive(FALSE)
demTheme.SetVisible(TRUE)
demTheme.SetName("Demand Regions by OTC Logistics Cost")
theView.AddTheme(demTheme)
theView.Invalidate

return Nil
```

```
'*****************************************************************

' Scriptname:    DRRxLogTheme.Make

' Filename:      drrxlogt.ave

' Author:        Kenneth Bennett

' Date:          May 6, 1998

' Description:   Creates a pie chart theme of the Demand
'                Regions where the pie slices represent the
'                Rx logistics component costs and the size of the
'                whole pie represents the total Rx logistics cost.

' Requires:      Demand Regions theme must exist

' Called by:     View menu item click event ("Display Demand
'                Regions: by Rx Logistics Cost")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "DRRxLogTheme.Make"

' Find the view and the Demand Regions theme

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theView.GetWin.Open
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Find the needed fields

shipFld = theFTab.FindField("Rx Direct Cost")
pickFld = theFTab.FindField("Rx Pick Cost")
tranFld = theFTab.FindField("Rx Tranship Cost")
```

```
xdocFld = theFTab.FindField("Rx Crossdock Cost")
totlFld = theFTab.FindField("Rx Total Cost")

if ((shipFld = Nil) OR (pickFld = Nil) OR (tranFld = Nil)
     OR (xdocFld = Nil) OR (totlFld = Nil)) then
  MsgBox.Error("ERROR: One or more required fields is missing"+nl+
               "or has been renamed.", Scriptname)
  exit
end

shipFld = shipFld.AsString
pickFld = pickFld.AsString
tranFld = tranFld.AsString
xdocFld = xdocFld.AsString
totlFld = totlFld.AsString

fldStringList = {shipFld, pickFld, tranFld, xdocFld}

' Check to see if the new theme already exists
checkTheme = theView.FindTheme("Demand Regions by
                               Rx Logistics Cost")
if (checkTheme <> Nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Demand
                                Regions by Rx Logistics Cost")
  if (theTable <> Nil) then
    av.GetProject.RemoveDoc(theTable)
  end
end


' Clone the theme and work with the new theme

demTheme = theTheme.Clone


' Get the new Demand Region theme's legend

demLegend = demTheme.GetLegend

'Create as many fill symbols as you have
'fieldNames and place them in a list.

shipsym = RasterFill.Make
shipsym.SetStyle(#RASTERFILL_STYLE_SOLID)
shipsym.SetColor(Color.GetBlue)
picksym = RasterFill.Make
picksym.SetStyle(#RASTERFILL_STYLE_SOLID)
picksym.SetColor(Color.GetRed)
transym = RasterFill.Make
transym.SetStyle(#RASTERFILL_STYLE_SOLID)
transym.SetColor(Color.GetGreen)
xdocsym= RasterFill.Make
xdocsym.SetStyle(#RASTERFILL_STYLE_SOLID)
xdocsym.SetColor(Color.GetMagenta)
```

```
theSyms = {shipsym, picksym, transym, xdocsym}

' Make a background fill Symbol that is empty

BGsym = RasterFill.Make
BGsym.SetStyle(#RASTERFILL_STYLE_EMPTY)

' Create the New Legend

demLegend.PieChart(demTheme,fldStringList,theSyms,BGSym,totlFld)

' To set a size field:

theSym = demLegend.GetSymbol(demLegend.ReturnFieldNames, false)

theSym.SetMinSize(8)
theSym.SetMaxSize(24)

' Redraw the theme using the PieChart legend.

demTheme.UpdateLegend
demTheme.SetActive(FALSE)
demTheme.SetVisible(TRUE)
demTheme.SetName("Demand Regions by Rx Logistics Cost")
theView.AddTheme(demTheme)
theView.Invalidate

return Nil
```

```
'************************************************************
' Scriptname:  FlowLine.Build

' Filename:    flowline.ave

' Description: Script creates a line theme of all the
'              lines connecting all of the demand regions.
'              FTab contains fields for the DC name and
'              the demand region 3-digit (zip) code.

' Called by:   TransportationLines.Build

' Calls:       SpliceLatLon, FlowValues.Calculate

' SELF:        Nil

' Returns:     an FTab

'************************************************************
Scriptname = "FlowLine.Build"
srcDcsVtab = av.GetProject.FindDoc("inputfac.dbf").GetVTab
if (srcDcsVtab = Nil) then
  MsgBox.Error("The inputfac.dbf table was not found."+NL+
               "Exiting...", Scriptname)
  exit
end
theFTab = av.Run("SpliceLatLon", {srcDcsVtab})
facfld = theFTab.FindField("Facility")
if (facfld = Nil) then
  MsgBox.Error("Facility field not found.  Exiting...",
                Scriptname)
  exit
end

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

strTheme = theView.FindTheme("Demand Regions")
strVTab = strTheme.GetFTab
strfld = strVtab.FindField("Demand Region")
x2fld = strVtab.FindField("X")
y2fld = strVtab.FindField("Y")
if ((strfld = Nil) OR (x2fld = Nil) OR (y2fld = Nil)) then
  MsgBox.Error("Demand Region, X, or Y field in Attributes"+NL+
               "of Demand Regions is missing.  Exiting...",
               Scriptname)
  exit
end
```

```
theView = av.GetActiveDoc
dcsVTab = theView.FindTheme("CVS DCs").GetFTab
origfld = dcsVtab.FindField("Facility")
x1fld = dcsVtab.FindField("X-coord")
y1fld = dcsVtab.FindField("Y-coord")
if ((origfld = Nil) OR (x1fld = Nil) OR (y2fld = Nil)) then
  MsgBox.Error("Facility, X-coord, or Y-coord field"+NL+
               "in CVS DCs theme is missing.  Exiting...",
               Scriptname)
  exit
end

defName = FileName.Make(av.GetProject.GetWorkDir.AsString).MakeTmp
                        ("flolin", "dbf")
theFName = FileDialog.Put(defName, "*.dbf", "Save FTab As")
if (nil <> theFName) then
  lineFTab = FTab.MakeNew( theFName, POLYLINE )
else
  lineFTab = FTab.MakeNew( defName, POLYLINE )
end

'add fields to the new lineFTab

from = Field.Make("DC", #FIELD_CHAR, 20, 0)
to = Field.Make("Store", #FIELD_CHAR, 20, 0)

lineFTab.AddFields({from,to})
shapeF = lineFTab.FindField( "shape" )

av.ShowStopButton
av.ShowMsg("Building Flow Lines...")
numStores = dcsVtab.GetNumRecords
n = 0
for each i in dcsVtab
    Orig1 = dcsVtab.ReturnValue(origfld, i)
    x1 = dcsVtab.ReturnValue(x1fld, i)
    y1 = dcsVtab.ReturnValue(y1fld, i)
  for each j in strVtab
     Dest1 = strVtab.ReturnValue(strfld,j)
     x2 = strVtab.ReturnValue(x2fld,j)
     y2 = strVtab.ReturnValue(y2fld,j)

    ' build the line between each dc-store pair and add
    ' the Shape, DC and Store fields, as well as the CW units
    ' and the Rate and Distance fields, to lineVTab

    newRec = lineFTab.AddRecord
    l = Line.Make(x1@y1,x2@y2).AsPolyLine
    lineFTab.SetValue( shapeF, newRec, l )
    lineFTab.SetValue( from, newRec, Orig1 )
    lineFTab.SetValue( to, newRec, Dest1 )
  end
  n = n + 1
  progress = (n / numStores) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
```

```
        break
  end
end

' make lineFTab into a theme

theTheme = FTheme.Make( lineFTab )
theSymList = theTheme.GetLegend.GetSymbols
theColorPaletteList = av.GetSymbolWin.GetPalette.GetList
                                      (#PALETTE_LIST_COLOR)
theColor = theColorPaletteList.Get(8)
theSymbol = theSymList.Get(0)
theSymbol.SetColor(theColor)
theTheme.SetName("DC-to-Region Flow")
theTheme.UpdateLegend

'  Add the theme to the view

theView.AddTheme( theTheme )
theView.Invalidate


' Run the script to calculate the various DC to
' Region flow values

av.Run("FlowValues.Calculate", {theTheme})

return theFTab
```

```
'*****************************************************************

' Scriptname:    FlowValues.Calculate

' Filename:      flowvalu.ave

' Author:        Kenneth Bennett

' Date:          May 1, 1998

' Description:   Script copies the values of flows for OTC, Rx,
'                and CW products from the dirstore.dbf file to
'                the DC-to-Region FTab using an
'                origin-destination-product type string
'                concatenation.  These three new fields in
'                the FTab are then totalled and the total value
'                is added to fourth new field in the FTab called
'                Total Flow.
'
' Requires:      dirstore.dbf file and DC-to-Region flow
'                theme must exist

' Called by:     Flowlines.Build

' Calls:         Nil

' SELF:          the DC-to-Region Flow theme

' Returns:       Nil

'*****************************************************************
Scriptname = "FlowValues.Calculate"

' Retrieve the theme argument

theTheme = SELF.Get(0)

' Find the dirstore.dbf file and add the new field
' concatenating the facility name, the demand region name,
' and the product category

theDirTable = av.Getproject.FindDoc("dirstore.dbf")
if (theDirTable = Nil) then
  MsgBox.Error("ERROR: dirstore.dbf table does not exist."+NL+
         "DC-to-Region flow values not calculated.", Scriptname)
  exit
end
theVTab = theDirTable.GetVTab
theVTab.SetEditable(TRUE)
odpfld2 = Field.Make("ODP",#FIELD_CHAR,35,0)
theVTab.AddFields({odpfld2})
theval = "[Facility]+[DemandRegion]+[Product]"
theVTab.Calculate(theval,odpfld2)


' Get the Demand-to-Store FTab and add the new flow fields for
' the three product categories, the total flow, and the
' origin-destination-product (ODP) field

theFTab = theTheme.GetFTab
theFTab.SetEditable(TRUE)
cwfld = Field.Make("CW Flow",#FIELD_DECIMAL, 16,2)
otcfld = Field.Make("OTC Flow",#FIELD_DECIMAL, 16,2)
rxfld = Field.Make("Rx Flow",#FIELD_DECIMAL, 16, 2)
totfld = Field.Make("Total Flow",#FIELD_DECIMAL, 16, 2)
odpfld = Field.Make("ODP",#FIELD_CHAR,35, 0)
theFTab.AddFields({cwfld, rxfld, otcfld, totfld, odpfld})

' Calculate the CW flow

newval2 = "[DC]+[Store]+""CW"""
theFTab.Calculate(newval2,odpfld)
theFTab.Join(odpfld,theVTab,odpfld2)
theflowval = "[OptimizedValue]"
theFTab.Calculate(theflowval,cwfld)
theFTab.UnjoinAll

' Calculate the Rx flow

newval2 = "[DC]+[Store]+""Rx"""
theFTab.Calculate(newval2,odpfld)
theFTab.Join(odpfld,theVTab,odpfld2)
theflowval = "[OptimizedValue]"
theFTab.Calculate(theflowval,rxfld)
theFTab.UnjoinAll

' Calculate the OTC flow

newval2 = "[DC]+[Store]+""OTC"""
theFTab.Calculate(newval2,odpfld)
theFTab.Join(odpfld,theVTab,odpfld2)
theflowval = "[OptimizedValue]"
theFTab.Calculate(theflowval,otcfld)
theFTab.UnjoinAll

' Calculate the total flow

newval2 = "[Rx Flow] + [CW Flow] + [OTC Flow]"
theFTab.Calculate(newval2,totfld)

' Remove the ODP field from the Demand-to-Store FTab
' since it is no longer needed

theFTab.RemoveFields({odpfld})
theFTab.SetEditable(FALSE)

' Remove the ODP field from the dirstore.dbf file

theVTab.RemoveFields({odpfld2})
theVTab.SetEditable(FALSE)

return Nil
```

```
'*****************************************************************

' Scriptname:    HandValues.Calculate

' Filename:      handvalu.ave

' Author:        Kenneth Bennett

' Date:          May 1, 1998

' Description:   Script copies the OptimizedValue field for
'                OTC, Rx, and CW products from the handling.dbf
'                file to the CVS DCs FTab using an
'                facility-product type string concatenation.
'                These three new fields in the FTab are then
'                totalled and the total value is added to
'                fourth new field in the FTab called
'                Total Handling.
'
' Requires:      handling.dbf file and CVS DCs theme must exist

' Called by:     SpliceLatLon

' Calls:         Nil

' SELF:          the CVS DCs FTab

' Returns:       Nil

'*****************************************************************
Scriptname = "HandValues.Calculate"

' Retrieve the theme argument

theFTab = SELF.Get(0)

' Find the dirstore.dbf file and add the new field
' concatenating the facility name, the demand region name,
' and the product category

theHandTable = av.Getproject.FindDoc("handling.dbf")
if (theHandTable = Nil) then
  MsgBox.Error("ERROR: handling.dbf table does not exist."+NL+
     "CVS DCs handling cost values not calculated.", Scriptname)
  exit
end
theVTab = theHandTable.GetVTab
theOptFld = theVTab.FindField("OptimizedValue")
if (theOptFld <> Nil) then
  theOptFld.SetAlias("OptVal")
end
theVTab.SetEditable(TRUE)
fpfld2 = Field.Make("FP",#FIELD_CHAR,35,0)
theVTab.AddFields({fpfld2})
theval = "[Facility]+[Product]"
theVTab.Calculate(theval,fpfld2)
```

```
' Add the new handling cost fields for
' the three product categories, the total cost,
' and the Facility-Product field

theFTab.SetEditable(TRUE)
cwfld = Field.Make("CW Handling",#FIELD_DECIMAL, 16,2)
otcfld = Field.Make("OTC Handling",#FIELD_DECIMAL, 16,2)
rxfld = Field.Make("Rx Handling",#FIELD_DECIMAL, 16, 2)
totfld = Field.Make("TotHandVal",#FIELD_DECIMAL, 16, 2)
fpfld = Field.Make("FP",#FIELD_CHAR,35, 0)
theFTab.AddFields({cwfld, rxfld, otcfld, totfld, fpfld})

' Calculate the CW flow

newval2 = "[Facility]+""CW"""
theFTab.Calculate(newval2,fpfld)
theFTab.Join(fpfld,theVTab,fpfld2)
thehandval = "[OptVal]"
theFTab.Calculate(thehandval,cwfld)
theFTab.UnjoinAll

' Calculate the Rx flow

newval2 = "[Facility]+""Rx"""
theFTab.Calculate(newval2,fpfld)
theFTab.Join(fpfld,theVTab,fpfld2)
thehandval = "[OptVal]"
theFTab.Calculate(thehandval,rxfld)
theFTab.UnjoinAll

' Calculate the OTC flow

newval2 = "[Facility]+""OTC"""
theFTab.Calculate(newval2,fpfld)
theFTab.Join(fpfld,theVTab,fpfld2)
thehandval = "[OptVal]"
theFTab.Calculate(thehandval,otcfld)
theFTab.UnjoinAll

' Calculate the total flow

newval2 = "[Rx Handling] + [CW Handling] + [OTC Handling]"
theFTab.Calculate(newval2,totfld)

' Remove the FP field from the CVS DCs FTab
' since it is no longer needed

theFTab.RemoveFields({fpfld})
theFTab.SetEditable(FALSE)

' Remove the FP field from the handling.dbf VTab

theVTab.RemoveFields({fpfld2})
theVTab.SetEditable(FALSE)

return theFTab
```

```
'*****************************************************************

' Scriptname:    HasCWRx

' Filename:      hascwrx.ave

' Description:   Add HasCW and HasRx fields to Attributes
'                of CVS DCs theme table.

' Requires:      Nil

' Called by:     CalcDCs

' Calls:         Nil

' SELF:          the Demand by Regions View and the CVS DCs FTab

' Returns:       Nil

'*****************************************************************
Scriptname = "HasCWRx"

' First get the Demand by Region view and the CVS DCs Ftab
' for Attibutes of DC2S

theView = SELF.Get(0)
theFTab = SELF.Get(1)

' Add two fields to DCs attribute table HasRx and HasCW

edit_state = theFTab.IsEditable

' Make sure table is editable and that fields can be added

if (theFtab.CanEdit) then
  theFTab.SetEditable(true)
  if ((theFTab.CanAddFields).Not) then
    MsgBox.Info("Can't add fields to the table."+NL+
                "Check write permission.",
    "Can't add HasRx and HasCW")
    exit
  end
else
  MsgBox.Info("Can't modify the feature table."+NL+
  "Check write permission.","Can't add HasRx and HasCW")
  exit
end

'Check if fields named "Has Rx" and "Has CW" exist

rx_exists = (theFTab.FindField("HasRx") = NIL).Not
cw_exists = (theFtab.FindField("HasCW") = NIL).Not

' If they do exist, ask if they should be overwritten.
' Otherwise, just make them and add them to CVS DCs FTab
```

```
if (rx_exists or cw_exists) then
  if (MsgBox.YesNo("Overwrite existing fields?",
  "HasRx and HasCW fields already exist", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created
    'from another script).
    if (rx_exists) then
      theFTab.RemoveFields({theFTab.FindField("HasRx")})
    end
    if (cw_exists) then
      theFTab.RemoveFields({theFTab.FindField("HasCW")})
    end
  else
    exit
  end  'if (MsgBox...)
end  'if

rx = Field.Make ("HasRx",#FIELD_DECIMAL,1,0)
cw = Field.Make ("HasCW",#FIELD_DECIMAL,1,0)
theFTab.AddFields({rx,cw})

' Get the Facility name field of the CVS DCs FTab

theDcFld = theFTab.FindField("Facility")

' Get the Facility and Process fields of the
' picking.dbf table's FTab

thePkVTab = av.GetProject.FindDoc("picking.dbf").GetVTab
thePkDcFld = thePkVTab.FindField("Facility")
theProcess = thePkVTab.FindField("Process")

' Loop through the CVS DCs FTab and populate the HasRx
' and the HasCW fields.  Assign zero if it does not
' warehouse these products, one if it does.

for each i in theFTab
  rxval = 0
  cwval = 0
  theFTab.SetValue(rx,i,rxval)
  theFTab.SetValue(cw,i,cwval)
  theDC = theFTab.ReturnValue(theDcFld,i)
  for each j in thePkVTab
    thePk = thePkVTab.ReturnValue(thePkDcFld, j)
    if (thePk <> theDC) then
      continue
    end
    theMkStr = thePkVTab.ReturnValue(theProcess,j)
    pos = theMkStr.IndexOf(" ")
    nchars = theMkStr.Count
    rdchars = nchars - pos - 1
    theActivity = theMkStr.Right(rdchars).Trim.UCase
    if (theActivity = "RX") then
      rxval = 1
      theFTab.SetValue(rx,i,rxval)
    end
    if (theActivity = "CW") then
      cwval = 1
      theFTab.SetValue(cw,i,cwval)
    end
  end
end

' Return the CVS DCs FTab to original edit state.

theFTab.SetEditable(edit_state)

return Nil
```

```
'********************************************************************

' Scriptname:    OTCFlowTheme.Make

' Filename:      otcflowt.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a Flow theme based on the
'                OTC Flow field in the DC-to-Region Flow theme
'                table.  Zero value flows are made invisible
'                using the null value and symbol

' Requires:      DC-to-Region flow theme must exist

' Called by:     View menu item click event ("Display Flows:
'                                   DC-to-Region by OTC Only")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'********************************************************************
Scriptname = "OTCFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("DC-to-Region Flow")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called DC-to-Region Flow
               does not exist.", Scriptname)
  exit
end

catString = "OTC Flow"

checkTheme = theView.FindTheme(catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of"++catString)
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the DC-to-Region Flow theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend
newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)

' Make zero the null value

newLegend.SetNullValue(catString, 0)

' Select a color from the color palette to be used
' in drawing the new line theme

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with into five natural breaks
' and weight the line thickness by the flow volume

newLegend.Natural(newTheme, catString, 5)
theSymbolList = newLegend.GetSymbols
thickness = 1
count = 0
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
  end
theSymbolList.UniformColor(theColor)

' Make the null symbol transparent

nullSym = Symbol.Make(#SYMBOL_PEN )
theNullColor = Color.GetBlue
theNullColor.SetTransparent(TRUE)
nullSym.SetColor(theNullColor)
newLegend.SetNullSymbol(nullSym)
newLegend.DisplayNoDataClass(FALSE)
newTheme.SetLegend(newLegend)
newTheme.SetName (catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme (newTheme)
newTheme.UpdateLegend
theView.Invalidate
theNullColor.SetTransparent(FALSE)

return Nil
```

```
'*****************************************************************

' Scriptname:      OTCStatistics.Generate

' Filename:        otcstati.ave

' Author:          Kenneth Bennett

' Date:            May 6, 1998

' Description:     Script sums each of the fields in the
'                  OTC Logistics Costs table and reports
'                  it to the user.

' Requires:        OTC Logistics Costs table exists

' Called by:       View menu item click
'                  ("Trace Costs: Chain-wide OTC Statistics")

' Calls:           Nil

' SELF:            Nil

' Returns:         Nil

'*****************************************************************
Scriptname = "OTCStatistics.Generate"

' Ensure two decimal places in each number

Script.The.SetNumberFormat("d.dd")

' Get the table

theTable = av.GetProject.FindDoc("OTC Logistics Costs")
if (theTable = Nil) then
  MsgBox.Error("OTC Logistics Costs table not found.", Scriptname)
  exit
end

' Get the VTab for the table

theVTab = theTable.GetVTab

' Get the number of records

num = theVTab.GetNumRecords

' Get the list of fields in the VTab

theFieldList = theVTab.GetFields

' Set the field variables

shipFld = theFieldList.Get(1)
pickFld = theFieldList.Get(2)
```

```
tranFld = theFieldList.Get(3)
xdocFld = theFieldList.Get(4)
totlFld = theFieldList.Get(5)

' Initialize summing variables

shipSum = 0
pickSum = 0
tranSum = 0
xdocSum = 0
totlSum = 0

' Loop through the VTab and sum each fld

for each rec in theVTab
  shipSum = shipSum + theVTab.ReturnValue(shipFld, rec)
  pickSum = pickSum + theVtab.ReturnValue(pickFld, rec)
  tranSum = tranSum + theVTab.ReturnValue(tranFld, rec)
  xdocSum = xdocSum + theVTab.ReturnValue(xdocFld, rec)
  totlSum = totlSum + theVTab.ReturnValue(totlFld, rec)
end

' Calculate the per store average for each field

shipAvg = shipSum / num
pickAvg = pickSum / num
tranAvg = tranSum / num
xdocAvg = xdocSum / num
totlAvg = totlSum / num

' Issue the report

reportString = "Total OTC Shipping Cost:"++shipSum.AsString+nl+
               "Average Per Region:"++shipAvg.AsString+nl+
               "Total OTC Pick Cost:"++pickSum.AsString+nl+
               "Average Per Region:"++pickAvg.AsString+nl+
               "Total OTC Transhipment Cost:"++tranSum.AsString
               +nl+"Average Per Region:"++tranAvg.AsString+nl+
               "Total OTC Crossdock Cost:"++xdocSum.AsString+nl+
               "Average Per Region:"++xdocAvg.AsString+nl+nl+
               "Chain-wide Total OTC Logistics Cost:"++
               totlSum.AsString+nl+"Average Total Cost
               Per Region:"++totlAvg.AsString+nl

MsgBox.Report(reportString, "Chain-wide OTC Statistics")

return Nil
```

```
'*****************************************************************

' Scriptname:    ProdDemRegTheme.Make

' Filename:      proddemr.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Creates a pie chart theme of the Demand
'                Regions where the pie slices represent the
'                three product categories and the size of the
'                whole pie represents the total demand.

' Requires:      Demand Regions theme must exist

' Called by:     View menu item click event ("Display Demand
'                Regions: by Product Volume")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "ProdDemRegTheme.Make"

' Find the view and the Demand Regions theme

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theView.GetWin.Open
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
                Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Find the needed fields

oFld = theFTab.FindField("OTC_Vol")
rFld = theFTab.FindField("Rx_Vol")
cFld = theFTab.FindField("CW_Vol")
```

```
totFld = theFTab.FindField("Total Demand")

if ((oFld = Nil) OR (rFld = Nil) OR (cFld = Nil) OR
                                        (totFld = Nil)) then
  MsgBox.Error("ERROR: Require product fields are missing.",
               Scriptname)
  exit
end

oFld = oFld.AsString
rFld = rFld.AsString
cFld = cFld.AsString

fldStringList = {oFld, rFld, cFld}

' Check to see if the new theme already exists
checkTheme = theView.FindTheme("Demand Regions by Product Volume")
if (checkTheme <> Nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Demand
                                      Regions by Product Volume")
  if (theTable <> Nil) then
    av.GetProject.RemoveDoc(theTable)
  end
end


' Clone the theme and work with the new theme

demTheme = theTheme.Clone


' Get the new Demand Region theme's legend

demLegend = demTheme.GetLegend

'Create as many fill symbols as you have
'fieldNames and place them in a list.

otcsym = RasterFill.Make
otcsym.SetStyle(#RASTERFILL_STYLE_SOLID)
otcsym.SetColor(Color.GetBlue)
rxsym = RasterFill.Make
rxsym.SetStyle(#RASTERFILL_STYLE_SOLID)
rxsym.SetColor(Color.GetRed)
cwsym = RasterFill.Make
cwsym.SetStyle(#RASTERFILL_STYLE_SOLID)
cwsym.SetColor(Color.GetGreen)

theSyms = {otcsym, rxsym, cwsym}

' Make a background fill Symbol that is empty

BGsym = RasterFill.Make
BGsym.SetStyle(#RASTERFILL_STYLE_EMPTY)
```

```
' Create the New Legend

demLegend.PieChart(demTheme,fldStringList,theSyms,BGSym,
                   "Total Demand")

' To set a size field:

theSym = demLegend.GetSymbol(demLegend.ReturnFieldNames, false)

theSym.SetMinSize(8)
theSym.SetMaxSize(24)

' Redraw the theme using the PieChart legend.

demTheme.UpdateLegend
demTheme.SetActive(FALSE)
demTheme.SetVisible(TRUE)
demTheme.SetName("Demand Regions by Product Volume")
theView.AddTheme(demTheme)
theView.Invalidate

return Nil
```

```
'*****************************************************************

' Scriptname:    RxFlowTheme.Make

' Filename:      rxflowth.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a Flow theme based on the
'                Rx Flow field in the DC-to-Region Flow theme
'                table.  Zero value flows are made invisible
'                using the null value and symbol

' Requires:      DC-to-Region flow theme must exist

' Called by:     View menu item click event ("Display Flows:
'                                      DC-to-Region by Rx Only")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "RxFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("DC-to-Region Flow")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called DC-to-Region Flow
               does not exist.", Scriptname)
  exit
end

catString = "Rx Flow"

checkTheme = theView.FindTheme(catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of"++catString)
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the DC-to-Region Flow theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend
newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)

' Make zero the null value

newLegend.SetNullValue(catString, 0)

' Select a color from the color palette to be used
' in drawing the new line theme

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with into five natural breaks
' and weight the line thickness by the flow volume

newLegend.Natural(newTheme, catString, 5)
theSymbolList = newLegend.GetSymbols
thickness = 1
count = 0
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
  end
theSymbolList.UniformColor(theColor)

' Make the null symbol transparent

nullSym = Symbol.Make(#SYMBOL_PEN )
theNullColor = Color.GetBlue
theNullColor.SetTransparent(TRUE)
nullSym.SetColor(theNullColor)
newLegend.SetNullSymbol(nullSym)
newLegend.DisplayNoDataClass(FALSE)
newTheme.SetLegend(newLegend)
newTheme.SetName (catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme (newTheme)
newTheme.UpdateLegend
theView.Invalidate
theNullColor.SetTransparent(FALSE)

return Nil
```

```
'**********************************************************

' Scriptname:      RxStatistics.Generate

' Filename:        rxstatis.ave

' Author:          Kenneth Bennett

' Date:            May 6, 1998

' Description:     Script sums each of the fields in the
'                  Rx Logistics Costs table and reports
'                  it to the user.

' Requires:        Rx Logistics Costs table exists

' Called by:       View menu item click
'                  ("Trace Costs: Chain-wide Rx Statistics")

' Calls:           Nil

' SELF:            Nil

' Returns:         Nil

'**********************************************************
Scriptname = "RxStatistics.Generate"

' Ensure two decimal places in each number

Script.The.SetNumberFormat("d.dd")

' Get the table

theTable = av.GetProject.FindDoc("Rx Logistics Costs")
if (theTable = Nil) then
  MsgBox.Error("Rx Logistics Costs table not found.",
               Scriptname)
  exit
end

' Get the VTab for the table

theVTab = theTable.GetVTab

' Get the number of records

num = theVTab.GetNumRecords

' Get the list of fields in the VTab

theFieldList = theVTab.GetFields

' Set the field variables

shipFld = theFieldList.Get(1)
```

```
pickFld = theFieldList.Get(2)
tranFld = theFieldList.Get(3)
xdocFld = theFieldList.Get(4)
totlFld = theFieldList.Get(5)

' Initialize summing variables

shipSum = 0
pickSum = 0
tranSum = 0
xdocSum = 0
totlSum = 0

' Loop through the VTab and sum each fld

for each rec in theVTab
  shipSum = shipSum + theVTab.ReturnValue(shipFld, rec)
  pickSum = pickSum + theVtab.ReturnValue(pickFld, rec)
  tranSum = tranSum + theVTab.ReturnValue(tranFld, rec)
  xdocSum = xdocSum + theVTab.ReturnValue(xdocFld, rec)
  totlSum = totlSum + theVTab.ReturnValue(totlFld, rec)
end

' Calculate the per store average for each field

shipAvg = shipSum / num
pickAvg = pickSum / num
tranAvg = tranSum / num
xdocAvg = xdocSum / num
totlAvg = totlSum / num

' Issue the report

reportString = "Total Rx Shipping Cost:"++shipSum.AsString+nl+
               "Average Per Region:"++shipAvg.AsString+nl+
               "Total Rx Pick Cost:"++pickSum.AsString+nl+
               "Average Per Region:"++pickAvg.AsString+nl+
               "Total Rx Transhipment Cost:"++tranSum.AsString
                +nl+"Average Per Region:"++tranAvg.AsString+nl+
               "Total Rx Crossdock Cost:"++xdocSum.AsString+nl+
               "Average Per Region:"++xdocAvg.AsString+nl+nl+
               "Chain-wide Total Rx Logistics Cost:"++
               totlSum.AsString+nl+"Average Total Cost
               Per Region:"++totlAvg.AsString+nl

MsgBox.Report(reportString, "Chain-wide Rx Statistics")

return Nil
```

```
'*****************************************************************
' Scriptname:  SpliceLatLon

' Filename:    splicela.ave

' Description: Script takes a single Lat/Lon field and splits
'              them into one Lat field and one Longitude field.
'              It then creates the CVS DCs theme using those
'              lat/lon coordinates.

' Called by:   FlowLine.Build

' Calls:       AddXY

' SELF:        a VTab

' Returns:     an FTab
'*****************************************************************
Scriptname = "SpliceLatLon"
aVTab = SELF.Get(0)
if (aVTab = nil) then
  MsgBox.Info("Error - Table not found","")
  exit
end


' Get the fields to copy from aVTab

theStrFld = aVTab.FindField( "Latlon" )
idV = aVTab.FindField( "Facility" )
fld2 = aVTab.FindField( "Fixedcost" )
fld3 = aVTab.FindField( "Minimum" )
fld4 = aVTab.FindField( "Maximum" )
fld5 = aVTab.FindField( "Optimizedv" )
optstate = aVTab.FindField( "Optimizeds" )
thefldList = { idV, fld2, fld3, fld4, optstate, fld5 }.DeepClone

' Create an FTAB and get its fields

defName = FileName.Make(av.GetProject.GetWorkDir.AsString).MakeTmp
                        ("dcmod", "dbf")
theFName = FileDialog.Put(defName, "*.dbf", "Save FTab As")
if (nil <> theFName) then
  myFTab = FTab.MakeNew( theFName, POINT )
else
  myFTab = FTab.MakeNew( defName, POINT )
end
myFTab.AddFields( thefldList )
shapeF = myFTab.FindField( "shape" )
idF = myFTab.FindField( "Facility" )
fcostfld = myFTab.FindField( "Fixedcost" )
minfld = myFTab.FindField( "Minimum" )
maxfld = myFTab.FindField( "Maximum" )
optstatefld = myFTab.FindField( "Optimizeds" )
optfld = myFTab.FindField( "Optimizedv" )
```

```
' copy each row in the VTab to the new FTab

for each i in aVTab
  ' Get the values from aVTab
     ystr = aVTab.ReturnValue(theStrFld, i )
     thePos = ystr.IndexOf("/")
     y = ystr.Left(thePos).Trim.AsNumber
     xstr = aVTab.ReturnValue(theStrFld , i)
     x = xstr.Right(xstr.IndexOf("/")).Trim.AsNumber
     id = aVTab.ReturnValue( idV, i )
     fcost = aVTab.ReturnValue( fld2, i )
     minim = aVTab.ReturnValue( fld3, i )
     maxim = aVTab.ReturnValue( fld4, i )
     opts = aVTab.ReturnValue( optstate, i )
     opt = aVTab.ReturnValue( fld5, i )
  ' create the next row and add values
     newRec = myFTab.AddRecord
     myFTab.SetValue( shapeF, newRec, x@y)
     myFTab.SetValue( idF, newRec, id )
     myFTab.SetValue( fcostfld, newRec, fcost )
     myFTab.SetValue( minfld, newRec, minim )
     myFTab.SetValue( maxfld, newRec, maxim )
     myFTab.SetValue( optstatefld, newRec, opts )
     myFTab.SetValue( optfld, newRec, opt )
end


' Create a palette using selected palette file

'thePalette = Palette.MakeFromFile(theDefPalFile)
thePalette = av.GetSymbolWin.GetPalette

' Now use myFTab to create a theme and add it to the active view

theTheme = FTheme.Make( myFTab )
theTheme.SetName("CVS DCs")
theLegend = theTheme.GetLegend
theSymList = theLegend.GetSymbols
theDCSymbol = theSymList.Get(0)

' Grab the Marker palette and get the outlined star symbol

theMarkerPaletteList = thePalette.GetList(#PALETTE_LIST_MARKER)
' outlined star is 34th symbol in Marker palette
chosenMarker = theMarkerPaletteList.Get(33)

' Grab the Color palette and get the color gold for the star

theColorPaletteList = thePalette.GetList(#PALETTE_LIST_COLOR)
' color gold is 39th color in Color palette
chosenColor = theColorPaletteList.Get(38)

' change the shape and color the DC symbol

chosenMarker.SetColor(chosenColor)
chosenMarker.SetSize(24)
```

```
theSymList.Set(0, chosenMarker)

' add the theme to the view

theTheme.UpdateLegend
theTheme.SetActive(True)
theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Warning("Demand by Region view does not exist."+NL+
                 "CVS DCs theme not added to that view.",
                 Scriptname)
elseif (not (theView.Is(View))) then
  MsgBox.Warning("Demand by Region view does not exist."+NL+
                 "CVS DCs theme not added to that view.",
                 Scriptname)
else
  theView.AddTheme(theTheme)
end

newFTab = av.Run("AddXY", Nil)

newFTab.SetEditable(FALSE)

' Copy the CVS DCs theme and paste it
' to the Location Strategy view

locView = av.GetProject.FindDoc("Location Strategy")
if (locView = Nil) then
  MsgBox.Warning("Location Strategy view does not exist."+NL+
                 "CVS DCs theme not copied to that view.",
                 Scriptname)
elseif (not (locView.Is(View))) then
  MsgBox.Warning("Location Strategy view does not exist."+NL+
                 "CVS DCs theme not copied to that view.",
                 Scriptname)
else
  checkTheme = locView.FindTheme("CVS DCs")
  if (checkTheme <> Nil) then
    locView.DeleteTheme(checkTheme)
  end
  dcTheme = theTheme.Clone
  locView.AddTheme(dcTheme)
  dcTheme.SetActive(FALSE)
end


return newFTab
```

```
'*****************************************************************

' Scriptname:    SQLTables.Get

' Filename:      sqltable.ave

' Description:   Script launches an SQL connection with MS
'                Access and imports five tables from a selected
'                Access database.  These five tables must have
'                the preset titles agreed upon by Anderson
'                Consulting and the UT team.  Once imported, the
'                SQL tables are exported out again as .dbf files
'                so they can be re-imported into the ArcView
'                project in a read/write state.

' Requires:      MS Access ODBC driver must be activated.  An
'                Ms Access .mdb file containing five tables
'                with the following names must be available.
'                DIRECT TO STORE
'                INPUT - FACILITIES
'                HANDLING
'                PICKING
'                TRANSHIPMENTS

' Called by:     Menu item click event ("SQLTables.Get")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "SQLTables.Get"

' Set up the SQL connection to MS Access and bring SQL tables
' into ArcView

theSQL=SQLCon.Find("MS Access")
if (theSQL = Nil) then
  MsgBox.Error("MS Access database not found."+NL+
               "Try reloading the MS Access ODBC driver.",
               "SQL Connection Error")
  exit
end

' Query the DIRECT TO STORE table

query = "Select * from [DIRECT TO STORE]"
theDTSVTab=VTab.MakeSQL(theSQL, query)
if (theDTSVTab = Nil) then
  exit
end
theDTSTable = Table.Make(theDTSVTab)
av.GetProject.AddDoc(theDTSTable)
theDTSTable.SetName("Direct To Store")
theDTSTable.GetWin.Open

query = "Select * from [INPUT - FACILITIES]"
theDCVTab = VTab.MakeSQL(theSQL, query)
theDCTable = Table.Make(theDCVTab)
av.GetProject.AddDoc(theDCTable)
theDCTable.SetName("Input - Facilities")
theDCTable.GetWin.Open

query = "Select * from PICKING"
thePickVTab = VTab.MakeSQL(theSQL, query)
thePickTable = Table.Make(thePickVTab)
av.GetProject.AddDoc(thePickTable)
thePickTable.SetName("Picking")
thePickTable.GetWin.Open

query = "Select * from HANDLING"
theHandVTab = VTab.MakeSQL(theSQL, query)
theHandTable = Table.Make(theHandVTab)
av.GetProject.AddDoc(theHandTable)
theHandTable.SetName("Handling")
theHandTable.GetWin.Open

query = "Select * from TRANSHIPMENTS"
theTransVTab = VTab.MakeSQL(theSQL, query)
theTransTable = Table.Make(theTransVTab)
av.GetProject.AddDoc(theTransTable)
theTransTable.SetName("Transhipments")
theTransTable.GetWin.Open


' Now convert the SQL tables to readable and writable
' .dbf files by exporting the SQL tables in .dbf format
' and re-importing those .dbf files into ArcView


theProject = av.GetProject
dts = theProject.FindDoc("Direct to Store")
infac = theProject.FindDoc("Input - Facilities")
hand = theProject.FindDoc("Handling")
pick = theProject.FindDoc("Picking")
tran = theProject.FindDoc("Transhipments")

'  Make a list of the tables and make sure they
' exist by looping through and checking
'  for Nil values

tabList = {dts, infac, hand, pick, tran}
for each t in tabList
  if (t = Nil) then
    MsgBox.Error("One or more required tables not
                    available ... Exiting.", Scriptname)
    exit
  end 'if
end' for loop
```

```
'  Get the VTabs for each of the above tables

dtsVTab = dts.GetVTab
infacVTab = infac.GetVTab
handVTab = hand.GetVTab
pickVTab = pick.GetVTab
tranVTab = tran.GetVTab

'  Get the working directory of the project

theDirectory = theProject.GetWorkDir.AsString

' create a filename for each VTab to
' be exported as a dbf file

dtsname = theDirectory + "\dirstore"
infacname = theDirectory + "\inputfac"
handname = theDirectory + "\handling"
pickname = theDirectory + "\picking"
tranname = theDirectory + "\tranship"

'  Export the VTabs

dtsfile = dtsVTab.Export(dtsname.AsFileName, dBase, FALSE)
infacfile = infacVTab.Export(infacname.AsFileName, dBase, FALSE)
handfile = handVTab.Export(handname.AsFileName, dBase, FALSE)
pickfile = pickVTab.Export(pickname.AsFileName, dBase, FALSE)
tranfile = tranVTab.Export(tranname.AsFileName, dBase, FALSE)

'  Add the tables to the project using the new VTabs

dtsTable = Table.Make(dtsfile)
dtsTable.SetName("dirstore.dbf")
infacTable = Table.Make(infacfile)
infacTable.SetName("inputfac.dbf")
handTable = Table.Make(handfile)
handTable.SetName("handling.dbf")
pickTable = Table.Make(pickfile)
pickTable.SetName("picking.dbf")
tranTable = Table.Make(tranfile)
tranTable.SetName("tranship.dbf")

'  Remove the SQL Tables from the project

theProject.RemoveDoc(dts)
theProject.RemoveDoc(infac)
theProject.RemoveDoc(hand)
theProject.RemoveDoc(pick)
theProject.RemoveDoc(tran)

' Disonnect the SQL connection

theSQL.Logout

' Make sure all former themes are removed and all
' Demand Regions table is unlinked and unjoined
```

```
theProject = av.GetProject
theView = theProject.FindDoc("Demand by Region")
if (theView = Nil) then
  return Nil
end
dcTheme = theView.FindTheme("CVS DCs")
if (dcTheme <> Nil) then
  theView.DeleteTheme(dcTheme)
end
dcTable = theProject.FindDoc("Attributes of CVS DCs")
if (dcTable <> Nil) then
  theProject.RemoveDoc(dcTable)
end
flowTheme = theView.FindTheme("DC-to-Region Flow")
if (flowTheme <> Nil) then
  theView.DeleteTheme(flowTheme)
end
flowTable = theProject.FindDoc("Attributes of DC-to-Region Flow")
if (flowTable <> Nil) then
  theProject.RemoveDoc(flowTable)
end
tranTheme = theView.FindTheme("Transhipments")
if (tranTheme <> Nil) then
  theView.DeleteTheme(tranTheme)
end
tranTable = theProject.FindDoc("Attributes of Transhipments")
if (tranTable <> Nil) then
  theProject.RemoveDoc(tranTable)
end

theDemRegTheme = theView.FindTheme("Demand Regions")
if (theDemRegTheme <> Nil) then
  theDemRegTheme.GetFTab.UnjoinAll
  theDemRegTheme.GetFTab.UnlinkAll
else
  if (theProject.FindDoc("Attributes of Demand Regions")
                         <> Nil) then
    theProject.FindDoc("Attributes of Demand Regions").GetVTab.
                         UnjoinAll
    theProject.FindDoc("Attributes of Demand Regions").GetVTab.
                         UnlinkAll
  end
end

MsgBox.Warning("Before proceeding, be sure to delete any"+nl+
               "remaining specialty themes based on the CVS DCs,"
               +nl+"the Demand Regions, the Transhipments, or
               the DC-"+nl+"to-Region Flows theme.",
               "Clean Up The Project")

return Nil
```

```
'************************************************************

' Scriptname:  Stores.SelByRange

' Filename:    stores_s.ave

' Description: Script builds service area polygons around each
'              DC such that each DC forms
'              the center point of five nested polygons
'              representing distance ranges of 50, 100, 150,
'              200, and 250 miles away from DC.  Each
'              distinct range is selected for each DC and
'              the stores that fall within that range are
'              selected.  For each range selection, the
'              script then calls the DC.Range script which
'              calculates a DC_Range field in the stores
'              theme and populates it with the range value
'              if it represents the closest DC to that store.
'              In other words, once the DC_Range field has
'              been populated for that store, it won't be
'              populated for a higher range value.

' Requires:    A line theme representing a reasonably
'              accurate road network and two point themes
'              representing the stores and DCs.

' Called by:   Menu click event ("DC Range")

' Calls:       DC.Range

' SELF:        Nil

' Returns:     Nil

'************************************************************
Scriptname = "Stores.SelByRange"

' Get the view

aView = av.GetProject.FindDoc("Location Strategy")
if (aView = Nil) then
  MsgBox.Error("ERROR: Location Strategy view does not exist.",
               Scriptname)
  exit
end
if (not (aView.Is(View))) then
  msgBox.Error("Selected document is not a view.",Scriptname)
  exit
end

' get the first line theme

aNetTheme = nil
theNetThemeList = {}
for each t in aView.GetThemes
  if (NetDef.CanMakeFromTheme(t)) then
```

```
    theNetThemeList.Add(t)
  end
end

' Have user select a theme

aNetTheme = MsgBox.List(theNetThemeList, "Select a line theme"+NL+
                        "to use as network:", Scriptname)

' check that a proper network theme was selected

if (aNetTheme = nil) then
  msgBox.Error("Network theme not selected.",Scriptname)
  exit
end

' make the NetDef and check it for errors
'
aNetDef = NetDef.Make(aNetTheme.GetFTab)
if (aNetDef.HasError) then
  msgBox.Error("NetDef has error.",Scriptname)
  exit
end

' make the Network object

aNetwork = Network.Make(aNetDef)

' get the point theme (to be used for stops)

aSiteTheme = nil
theSiteThemeList = {}
for each t in aView.GetThemes
  if ((t.GetFTab.GetSrcName.GetSubName = "Point") AND
      (t.GetName.Contains("DCs"))) then
    theSiteThemeList.Add(t)
   end
end

' Ask the user to select a site theme

aSiteTheme = MsgBox.List(theSiteThemeList, "Select a DCs theme:",
                         Scriptname)

' check if a stop theme was selected

if (aSiteTheme = nil) then
  msgBox.Error("Site theme not selected.",Scriptname)
  exit
end

aSiteFTab = aSiteTheme.GetFTab
pointShapeField = aSiteFTab.FindField("Shape")
pointLabelField = aSiteTheme.GetLabelField

' make a point list from the site theme, validate points, and
' set the name of each stop

aPointList = {}
for each rec in aSiteFTab
  p = aSiteFTab.ReturnValue(pointShapeField, rec)
  if (aNetwork.IsPointOnNetwork(p)) then
    p.SetName(aSiteFTab.ReturnValueString(pointLabelField, rec))
    aPointList.Add(p)
  end
end
numPoints = aPointList.Count

' Set the cost field
'
aCostFieldList = aNetDef.GetCostFields
aCostField = aCostFieldList.Get(1)
aCostSetting = aNetwork.SetCostField(aCostField)

' Find the service area

aCost = {50.00, 100.00, 150.00, 200.00, 250.00}

aCostList = {}

'add the list of ranges aCostList
'once for each point in theSiteTheme

for each s in 1..numPoints
aCostList.Add(aCost)
end


aFromPointBool = True
aCompactAreaBool = False
aResultBool = aNetwork.FindServiceArea(aPointList,aCostList,
                                aFromPointBool,aCompactAreaBool)
if (not (aResultBool)) then
  msgBox.Error("Unable to compute the service area",Scriptname)
  exit
end

' Write the results to new shapefiles

theWorkingDir = av.GetProject.GetWorkDir.AsString
aPathFileName1 = "c:\cvs\cvsac\thesis\shapefiles\snetwork"
                 .AsFileName

aPathFileName2 = "c:\cvs\cvsac\thesis\shapefiles\sarea"
                 .AsFileName
aNetwork.WriteServiceArea(aPathFileName1,aPathFileName2)

sName = srcName.Make(aPathFileName2.AsString+".shp")
servAreaTheme = Theme.Make(sName)
aView.AddTheme(servAreaTheme)
servAreaTheme.SetVisible(FALSE)
```

```
' Get the FTab of the Service Area theme

servFTab = servAreaTheme.GetFTab

' Have the user select stores theme

storeTheme = nil
thePointThemeList = {}
for each t in aView.GetThemes
  if (t.GetFTab.GetSrcName.GetSubName = "Point") then
    thePointThemeList.Add(t)
   end
end

' Get the name of the store theme

storeName = aSiteTheme.GetName.AsTokens(" ").Get(0)
storeTheme = aView.FindTheme(storeName++"Stores")

'' Ask the user to select a store theme
'
'storeTheme = MsgBox.List(thePointThemeList,
                         "Select a store theme:", Scriptname)

' check if a store theme was selected

if (storeTheme = nil) then
  msgBox.Error("Store theme not found.",Scriptname)
  exit
end

' Get the store theme FTab

storeFTab = storeTheme.GetFTab

' Add the DC_Range field to the stores FTab
' First get edit state of stores FTab, then
' set it to editable

edit_state = storeFTab.IsEditable
storeFTab.SetEditable(TRUE)

' Make the new DC_Range Field

fld = storeFTab.FindField("DC Range")
if (fld <> Nil) then
  storeFTab.Calculate("0", fld)
else
  fld = Field.Make("DC Range", #FIELD_SHORT, 5, 0)
  storeFTab.AddFields({fld})
  storeFTab.Calculate("0", fld)
end

' Stop editing and restore edit state

storeFTab.SetEditable(FALSE)
storeFTab.SetEditable(edit_state)

' Get the number of records in the DC theme table

numDCs = aSiteFTab.GetNumRecords

' Create a list of starting record numbers (i.e., 0 through 4
' since there are only five service area ranges).

startRecList = {0, 1, 2, 3, 4}
rangeList = {50 , 100, 150, 200, 250}

' Loop through starting Record List and each iteration select
' every fifth record in the service area theme

servBitMap = servFTab.GetSelection
dcSet = numDCs - 1
for each s in startRecList
  index = s
  servBitMap.ClearAll
  servFTab.UpdateSelection
  servBitMap.Set(index)

  'Loop through servBitMap and select every fifth record
  'Number of iterations is number of DCs minus one


  for each d in 1..dcSet ' sets the next n-1 number of records
    index = index + 5
    servBitMap.Set(index)
  end ' for each DC loop

  ' Using the selected ranges in the service area theme
  ' find the stores that intersect with those polygons.

  storeFTab.SelectByFTab(servFTab, #FTAB_RELTYPE_INTERSECTS,
                         0, #VTAB_SELTYPE_NEW)

  ' Get the range value

  rangeName = rangeList.Get(s)

  ' Call the DC.Range script to populate the DC_Range
  ' field with respective range value for the selected set.

  av.Run("DC.Range", {storeFTab, fld, rangeName, storeTheme})

end ' for each range loop
servBitMap.ClearAll
servFTab.UpdateSelection

' Query the stores FTab for records with zero value in
' DC Range field and calculate their values to 251 to
' represent the fact that they are stores that are
' greater than 250 miles away from the nearest DC.
```

```
storeBitMap = storeFTab.GetSelection
storeBitMap.ClearAll
storeFTab.UpdateSelection
queryString = "([DC Range] = 0)"
successful = storeFTab.Query(queryString, storeBitMap,
                             #VTAB_SELTYPE_NEW)
if (NOT successful) then
  MsgBox.Error("Query string did not compile."+NL+
               "See stores theme table.", Scriptname)
  exit
end
edit_state = storeFTab.IsEditable
storeFTab.SetEditable(TRUE)
storeFTab.Calculate("251", fld)
storeFTab.SetEditable(edit_state)
storeBitMap.ClearAll
storeFTab.UpdateSelection
activeThemeList = aView.GetActiveThemes
for each act in activeThemeList
  act.SetActive(FALSE)
end
storeTheme.SetActive(True)

' Delete the service area theme

aView.DeleteTheme(servAreaTheme)

return Nil
```

```
'*****************************************************************

' Scriptname:  SummD2S

' Filename:    summd2s.ave

' Description: Summarizes the dirstore.dbf table over the
'              facility field for each product type and stores
'              them in tables called RxDirect, CWDirect, and
'              OTCDirect

' Requires:    Nil

' Called by:   CalcDCs

' Calls:       Nil

' SELF:        Nil

' Returns:     Nil

'*****************************************************************
Scriptname = "SummD2S"

'First get Direct to Store file and the working directory

theTab = av.GetProject.FindDoc("dirstore.dbf")
theVTab = theTab.GetVTab
theDirectory = av.GetProject.GetWorkDir.AsString

'----------------------------------------------------------------
' Build RxDirect.dbf table
'----------------------------------------------------------------

' Check to see if RxDirect already exists

rx_exists = (av.GetProject.FindDoc("RxDirect.dbf") = NIL).Not
skip = 0
if (rx_exists) then
  thedoc = av.GetProject.FindDoc("RxDirect.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table RxDirect already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (rx_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
     skip = 1
  end  'if (MsgBox...)
end  'if

if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "([Product].UCase = "+"RX".Quote+")"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("Facility")
  flnm = theDirectory + "\RxDirect.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                     theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("RxDirect.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
    'thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Table can't be modified."+NL+
                 "Count field not deleted.", Scriptname)
  end
  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("Rx D2S")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

'-----------------------------------------------------------------
' Build CWDirect.dbf table
'-----------------------------------------------------------------

'Check if the summary for CW picked exists

cw_exists = (av.GetProject.FindDoc("CWDirect.dbf") = NIL).Not
skip = 0
if (cw_exists) then
  thedoc = av.GetProject.FindDoc("CWDirect.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table CWDirect already exists", false)) then
    'if ok to overwrite, delete the fields as
    ' they may not be defined
    'as required by this script (eg., created from
    'another script).
    if (cw_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    'exit
    skip = 1
  end  'if (MsgBox...)
end  'if
if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "([Product].UCase = "+"CW".Quote+")"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("Facility")
  flnm = theDirectory + "\CWDirect.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                        theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("CWDirect.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
    'thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Table can't be modified."+NL+
                   "Count field not deleted.", Scriptname)
  end

  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("CW D2S")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

'-----------------------------------------------------------------
' Build OTCDirect.dbf table
'-----------------------------------------------------------------

'Check if the summary for OTC direct exists

otc_exists = (av.GetProject.FindDoc("OTCDirect.dbf") = NIL).Not
skip = 0
if (otc_exists) then
  thedoc = av.GetProject.FindDoc("OTCDirect.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table OTCDirect already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (otc_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
    else
       skip = 1
      'exit
    end  'if (MsgBox...)
  end  'if
  if (skip = 0) then
    theBitMap = theVTab.GetSelection
    expr = "([Product].UCase = "+"OTC".Quote+")"
    theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    theSummaryField = theVTab.FindField("Facility")
    flnm = theDirectory + "\OTCDirect.dbf"
    fld1 = theVTab.FindField("OptimizedValue")
    sumFldList = {fld1}
    sumList = {#VTAB_SUMMARY_SUM}
    newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                            theSummaryField, sumFldList, sumList)
    newTable = Table.Make(newVTab)
    newTable.SetName("OTCDirect.dbf")
    'newTable.GetWin.Open
    thenewVTab = newTable.GetVtab

    '  Make sure table is editable and if so,
    '  remove the count field

    edit_state = thenewVTab.IsEditable
    if (thenewVTab.CanEdit) then
      thenewVTab.SetEditable(true)
      thefld = thenewVTab.FindField("Count")
      thenewVTab.RemoveFields({thefld})
      'thenewVTab.SetEditable(false)
    else
      MsgBox.Warning("Table can't be modified."+NL+
                    "Count field not deleted.", Scriptname)
    end
    thefld = thenewVTab.FindField("Sum_OptimizedValue")
    thefld.SetAlias("OTC Picked")

  end
  theTab.GetVTab.GetSelection.ClearAll
  theTab.GetVTab.UpdateSelection

  return Nil
```

```
'*****************************************************************

' Scriptname:    SummDems

' Filename:      summdems.ave

' Description:   This script is similar to CalcDCs but summarizes
'                the flows for demand regions instead of DCs.
'                Need to summarize the the dirstore.dbf table
'                over the demand regions for all flows greater
'                than zero

' Requires:      dirstore.dbf must exist

' Called by:     View menu item click event ("Sum Regions")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "SummDems"

'First get direct to store file and the working directory

theTab = av.GetProject.FindDoc("dirstore.dbf")
if (theTab = Nil) then
  MsgBox.Info("Could not find dirstore.dbf file. Exiting...",
             "ERROR")
  exit
end
theVTab = theTab.GetVTab
theDirectory = av.GetProject.GetWorkDir.AsString

'Check if the summary for Rx2Store exists

rx_exists = (av.GetProject.FindDoc("Rx2Store.dbf") = NIL).Not
skip = 0
if (rx_exists) then
  thedoc = av.GetProject.FindDoc("Rx2Store.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table Rx2Store already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (rx_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
     skip = 1
    'exit
  end  'if (MsgBox...)
end  'if
```

```
if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "(([Product].UCase = ""RX"") and
                  ([OptimizedValue] > 0))"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("DemandRegion")
  flnm = theDirectory + "\Rx2Store.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                        theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("Rx2Store.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
'    thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Can't modify the table."+NL+
    "Check write permission.","Can't delete Count field!")
    exit
  end

  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("Rx2Store")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection



'Check if the summary for CW2Store exists

cw_exists = (av.GetProject.FindDoc("CW2Store.dbf") = NIL).Not
skip = 0
if (cw_exists) then
  thedoc = av.GetProject.FindDoc("CW2Store.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table CW2Store already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (cw_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
```

```
    'exit
    skip = 1
  end  'if (MsgBox...)
end  'if
if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "(([Product].UCase = ""CW"") and
                   ([OptimizedValue] > 0))"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("DemandRegion")
  flnm = theDirectory + "\CW2Store.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                        theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("CW2Store.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
'    thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Can't modify the table."+NL+
    "Check write permission.","Can't delete Count field!")
    exit
  end
  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("CW2Store")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

'Check if the summary for OTC d2s exists
otc_exists = (av.GetProject.FindDoc("OTC2Store.dbf") = NIL).Not
skip = 0
if (otc_exists) then
  thedoc = av.GetProject.FindDoc("OTC2Store.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table OTC2Store already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (cw_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
```

```
    'exit
    skip = 1
  end  'if (MsgBox...)
end  'if
if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "(([Product].UCase = ""OTC"") and
                  ([OptimizedValue] > 0))"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("DemandRegion")
  flnm = theDirectory + "\OTC2Store.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                    theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("OTC2Store.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
'    thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Can't modify the table."+NL+
    "Check write permission.","Can't delete Count field!")
    exit
  end
  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("OTC2Store")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

'--------------------------------------------------------------
' Join the Rx2Store, CW2Store, and OTC2Store tables
' to the Demand Regions theme table
'--------------------------------------------------------------

' Get the Demand Regions FTab

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist."+NL+
               "Summaries not joined to Demand Regions theme.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
```

```
  MsgBox.Error("ERROR: Demand by Region doc is not a view."+NL+
              "Summaries not joined to Demand Regions theme.",
              Scriptname)
  exit
end
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called Demand
               Regions does not exist."+NL+
              "Summaries not joined to Demand
                 Regions theme.", Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Check to see if it already has joins

if (theFTab.IsBase.Not) then
  av.GetProject.SetModified(true)
end
theFTab.UnjoinAll

' Get the join field of the Demand Regions table

field1 = theFTab.FindField("Demand Region")

' Get the FTab and join field of the Rx2Store table
' and join it to the Demand Regions table

theVtab2 = av.FindDoc("Rx2Store.dbf").GetVTab
field2 = theVtab2.FindField("DemandRegion")
theFTab.Join(field1, theVtab2, field2)

' Get the FTab and join field of the CW2Store table
' and join it to the Demand Regions table

theVtab2 = av.FindDoc("CW2Store.dbf").GetVTab
field2 = theVtab2.FindField("DemandRegion")
theFTab.Join(field1, theVtab2, field2)

' Get the FTab and join field of the OTC2Store table
' and join it to the Demand Regions table

theVtab2 = av.FindDoc("OTC2Store.dbf").GetVTab
field2 = theVtab2.FindField("DemandRegion")
theFTab.Join(field1, theVtab2, field2)

'-------------------------------------------------------------
' Summarize the demand for all the products
'-------------------------------------------------------------

theFTab.SetEditable(true)

totalField = theFTab.FindField("Total Demand")

' Create the Total Demand field if necessary, and calculate
' its value to zero

if (totalField = Nil) then
  totalField = Field.Make("Total Demand", #FIELD_LONG, 12, 0)
  theFTab.AddFields({totalField})
end
theFTab.Calculate("0", totalField)

' Create the new "_Vol" fields for CW, Rx, and OTC
' if necessary and calculate their values to zero

catList = {"OTC", "Rx", "CW"}
for each cat in catList
  newFldString = cat+"_Vol"
  newFld = theFTab.FindField(newFldString)
  if (newFld = Nil) then
    newFld = Field.Make(newFldString, #FIELD_LONG, 12, 0)
    theFTab.AddFields({newFld})
  end
  theFTab.Calculate("0", newFld)
end

' Get the bitmap of the FTab and select all records in each
' joined sort field that is not null, then populate these
' values to its corresponding new field in the FTab using
' the calculate request

theBitMap = theFTab.GetSelection
theBitMap.ClearAll
theFTab.UpdateSelection
for each cat in catList
  fldName = cat+"2Store"
  theField = theFTab.FindField(fldName)
  if (theField <> Nil) then
    expr = "(["+fldName+"].IsNull.Not)"
    theFTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
    calcexpr = "["+fldName+"]"
    curstring = cat+"_Vol"
    calcfield = theFTab.FindField(curstring)
    theFTab.Calculate(calcexpr, calcfield)
  end
end
theBitMap.ClearAll
theFTab.UnjoinAll

' Get the fields to be totalled and calculate the
' Total Demand field

oFld = theFTab.FindField("OTC_Vol").AsString
rFld = theFTab.FindField("Rx_Vol").AsString
cFld = theFTab.FindField("CW_Vol").AsString

calcexpr = "(["+oFld+"] + ["+rFld+"] + ["+cFld+"])"
theFTab.Calculate(calcexpr, totalField)

' Stop editing and save changes to FTab
```

```
theFTab.SetEditable(FALSE)

MsgBox.Info("Summary of Demand Region Data complete.",
            "NOTICE")

return Nil
```

```
'*****************************************************************

' Scriptname:     SummTS

' Filename:       summts.ave

' Description:    Script summarizes the transhipment information
'                 which is later joined to the CVS DCs theme table

' Requires:       Nil

' Called by:      CalcDCs

' Calls:          Nil

' SELF:           Nil

' Returns:        Nil

'*****************************************************************
Scriptname = "SummTS"

' First get tranship.dbf table's VTab and the working directory

theTab = av.GetProject.FindDoc("tranship.dbf")
theVTab = theTab.GetVTab
theDirectory = av.GetProject.GetWorkDir.AsString

'----------------------------------------------------------------
' First do summaries over OriginFacilities field in picking.dbf
' and make pick tables called RxPicked and CWPicked.
'----------------------------------------------------------------

' Check if the summary for Rx picked exists

rx_exists = (av.GetProject.FindDoc("RxPicked.dbf") = NIL).Not
skip = 0
if (rx_exists) then
  thedoc = av.GetProject.FindDoc("RxPicked.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table RxPicked already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (rx_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
     skip = 1
  end  'if (MsgBox...)
end  'if

if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "([Product].UCase = "+"RX".Quote+")"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("OriginFacility")
  flnm = theDirectory + "\RxPicked.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                      theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("RxPicked.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

'  Make sure table is editable and if so,
'  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
    thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Table can't be modified."+NL+
                "Count field not deleted.", Scriptname)
  end
  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("RxPicked for TS")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

'Check if the summary for CW picked exists

cw_exists = (av.GetProject.FindDoc("CWPicked.dbf") = NIL).Not
skip = 0
if (cw_exists) then
  thedoc = av.GetProject.FindDoc("CWPicked.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table CWPicked already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (cw_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    skip = 1
  end  'if (MsgBox...)
end  'if

if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "([Product].UCase = "+"CW".Quote+")"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
```

```
    theSummaryField = theVTab.FindField("OriginFacility")
    flnm = theDirectory + "\CWPicked.dbf"
    fld1 = theVTab.FindField("OptimizedValue")
    sumFldList = {fld1}
    sumList = {#VTAB_SUMMARY_SUM}
    newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                          theSummaryField, sumFldList, sumList)
    newTable = Table.Make(newVTab)
    newTable.SetName("CWPicked.dbf")
    'newTable.GetWin.Open
    thenewVTab = newTable.GetVtab

    '  Make sure table is editable and if so,
    '  remove the count field

    edit_state = thenewVTab.IsEditable
    if (thenewVTab.CanEdit) then
      thenewVTab.SetEditable(true)
      thefld = thenewVTab.FindField("Count")
      thenewVTab.RemoveFields({thefld})
      thenewVTab.SetEditable(false)
    else
      MsgBox.Warning("Table can't be modified."+NL+
                     "Count field not deleted.", Scriptname)
    end
    thefld = thenewVTab.FindField("Sum_OptimizedValue")
    thefld.SetAlias("CWPicked for TS")
  end
  theTab.GetVTab.GetSelection.ClearAll
  theTab.GetVTab.UpdateSelection


  '-------------------------------------------------------------
  ' Now do summaries over DestFacility field in
  ' picking.dbf table and make crossdock tables called
  ' Rx_X_Doc and CW_X_Doc.
  '-------------------------------------------------------------

  'Check if the summary for Rx cross doc exists

  rx_exists = (av.GetProject.FindDoc("Rx_X_Doc.dbf") = NIL).Not
  skip = 0
  if (rx_exists) then
    thedoc = av.GetProject.FindDoc("Rx_X_Doc.dbf")
    if (MsgBox.YesNo("Overwrite existing table?",
    "The Table Rx_X_Doc already exists", false)) then
      'if ok to overwrite, delete the fields as they
      'may not be defined
      'as required by this script (eg., created from
      'another script).
      if (rx_exists) then
        av.GetProject.RemoveDoc(thedoc)
      end
    else
       skip = 1
      'exit
  end  'if (MsgBox...)
end  'if
if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "([Product].UCase = ""RX"")"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("DestinationFacility")
  flnm = theDirectory + "\Rx_X_Doc.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                        theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("Rx_X_Doc.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
    thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Table can't be modified."+NL+
                   "Count field not deleted.", Scriptname)
  end
  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("Rx_X_Doc")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

'Check if the summary for CW picked exists

cw_exists = (av.GetProject.FindDoc
                          ("CW_X_Doc.dbf") = NIL).Not
skip = 0
if (cw_exists) then
  thedoc = av.GetProject.FindDoc("CW_X_Doc.dbf")
  if (MsgBox.YesNo("Overwrite existing table?",
  "The Table CW_X_Doc already exists", false)) then
    'if ok to overwrite, delete the fields as they
    'may not be defined
    'as required by this script (eg., created from
    'another script).
    if (cw_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    'exit
    skip = 1
  end  'if (MsgBox...)
end  'if
if (skip = 0) then
  theBitMap = theVTab.GetSelection
  expr = "([Product].UCase = ""CW"")"
  theVTab.Query(expr, theBitMap, #VTAB_SELTYPE_NEW)
  theSummaryField = theVTab.FindField("DestinationFacility")
  flnm = theDirectory + "\CW_X_Doc.dbf"
  fld1 = theVTab.FindField("OptimizedValue")
  sumFldList = {fld1}
  sumList = {#VTAB_SUMMARY_SUM}
  newVTab = theVTab.Summarize(flnm.AsFileName, dBase,
                      theSummaryField, sumFldList, sumList)
  newTable = Table.Make(newVTab)
  newTable.SetName("CW_X_Doc.dbf")
  'newTable.GetWin.Open
  thenewVTab = newTable.GetVtab

  '  Make sure table is editable and if so,
  '  remove the count field

  edit_state = thenewVTab.IsEditable
  if (thenewVTab.CanEdit) then
    thenewVTab.SetEditable(true)
    thefld = thenewVTab.FindField("Count")
    thenewVTab.RemoveFields({thefld})
    thenewVTab.SetEditable(false)
  else
    MsgBox.Warning("Table can't be modified."+NL+
                   "Count field not deleted.", Scriptname)
    'exit
  end
  thefld = thenewVTab.FindField("Sum_OptimizedValue")
  thefld.SetAlias("CW_X_Doc")
end
theTab.GetVTab.GetSelection.ClearAll
theTab.GetVTab.UpdateSelection

return Nil
```

```
'*****************************************************************

' Scriptname:  Table.Convert

' Filename:    table_co.ave

' Description: Script requires that certain tables be brought
'              into the project from Anderson's MS Access
'              database via an SQL connection.  Script selects
'              each table, exports it as a dbf file, then adds
'              the new dbf file to the project.

' Called by:   Menu Click event ("Convert Tables")

' Calls:       Nil

' SELF:        Nil

' Returns:     Nil

'*****************************************************************

'  First locate the SQL tables in the project
theProject = av.GetProject
dts = theProject.FindDoc("Direct to Store")
infac = theProject.FindDoc("Input - Facilities")
hand = theProject.FindDoc("Handling")
pick = theProject.FindDoc("Picking")
tran = theProject.FindDoc("Transhipments")

'  Make a list of the tables and make sure they
'  exist by looping through and checking
'  for Nil values

tabList = {dts, infac, hand, pick, tran}
for each t in tabList
  if (t = Nil) then
    MsgBox.Error("One or more required tables not
                 available ... Exiting.", "Table.Convert")
    exit
  end 'if
end' for loop

'  Get the VTabs for each of the above tables

dtsVTab = dts.GetVTab
infacVTab = infac.GetVTab
handVTab = hand.GetVTab
pickVTab = pick.GetVTab
tranVTab = tran.GetVTab

'  Get the working directory of the project

theDirectory = theProject.GetWorkDir.AsString

'  create a filename for each VTab to be exported as a dbf file
```

```
dtsname = theDirectory + "\dirstore"
infacname = theDirectory + "\inputfac"
handname = theDirectory + "\handling"
pickname = theDirectory + "\picking"
tranname = theDirectory + "\tranship"

'  Export the VTabs

dtsfile = dtsVTab.Export(dtsname.AsFileName, dBase, FALSE)
infacfile = infacVTab.Export(infacname.AsFileName, dBase, FALSE)
handfile = handVTab.Export(handname.AsFileName, dBase, FALSE)
pickfile = pickVTab.Export(pickname.AsFileName, dBase, FALSE)
tranfile = tranVTab.Export(tranname.AsFileName, dBase, FALSE)

'  Add the tables to the project using the new VTabs

dtsTable = Table.Make(dtsfile)
dtsTable.SetName("dirstore.dbf")
infacTable = Table.Make(infacfile)
infacTable.SetName("inputfac.dbf")
handTable = Table.Make(handfile)
handTable.SetName("handling.dbf")
pickTable = Table.Make(pickfile)
pickTable.SetName("picking.dbf")
tranTable = Table.Make(tranfile)
tranTable.SetName("tranship.dbf")

'  Remove the SQL Tables from the project

theProject.RemoveDoc(dts)
theProject.RemoveDoc(infac)
theProject.RemoveDoc(hand)
theProject.RemoveDoc(pick)
theProject.RemoveDoc(tran)

'  Get the SQL connection and disconnect it

theCon = SQLCon.Find("MS Access 97 Database")
theCon.Logout
if (theCon = Nil) then
  MsgBox.Info("SQL connection closed.", "Table.Convert")
else
  MsgBox.Info("SQL connection not closed.", "Table.Convert")
end

return Nil
```

```
'*****************************************************************

' Scriptname:      Tables.Link

' Filename:        tables_1.ave

' Description:     Script links the Demand Regions FTab to the
'                  dirstore.dbf VTab, the dirstore.dbf VTab to
'                  the DCs FTab, and the DCs FTab to the
'                  tranship.dbf VTab.  These links are needed to
'                  run the trace scripts.

' Requires:        dirstore.dbf file, tranship.dbf file,
'                  Demand Regions theme, CVS DCs theme exist

' Called by:       Any of the Trace scripts

' Calls:           Nil

' SELF:            Nil

' Returns:         Nil

'*****************************************************************
Scriptname = "Tables.Link"

'  Get the tables to be linked

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
                Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
                Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab


'  Get the common fields and link Attributes of
'  Demand Regions to dirstore.dbf

theStoreFld1 = theStoreVTab.FindField("Demand Region")
theStoreFld2 = theD2SVTab.FindField("DemandRegion")
theStoreVTab.Link(theStoreFld1, theD2SVTab, theStoreFld2)
if (theSToreVTab.IsLinked.Not) then
  MsgBox.Warning("Link was unsuccessful...exiting.",
                  "Tables.Link")
  exit
end 'if

'  Get the common fields and link
'  dirstore.dbf to Attributes of DCs

theDCFld1 = theD2SVTab.FindField("Facility")
theDCFld2 = theDCVTab.FindField("Facility")
theD2SVTab.Link(theDCFld1, theDCVTab, theDCFld2)
if (theD2SVTab.IsLinked.Not) then
  MsgBox.Warning("Link was unsuccessful...exiting.",
                  "Tables.Link")
  theStoreVTab.UnlinkAll
  exit
end 'if

'  Get the common fields and link
'  Attributes of DCs to tranship.dbf

theFacFld1 = theDCFld2
theFacFld2 = theTSVTab.FindField("DestinationFacility")
theDCVTab.Link(theFacFld1, theTSVTab, theFacFld2)
if (theDCVTab.IsLinked.Not) then
  MsgBox.Warning("Link was unsuccessful...exiting.",
                  "Tables.Link")
  theStoreVTab.UnlinkAll
  theD2SVTab.UnlinkAll
  exit
end 'if

return Nil
```

```
'*****************************************************************

' Scriptname:       Tables.Unlink

' Filename:         tables_u.ave

' Description:      Script unlinks the Demand Regions FTab, the
'                   dirstore.dbf VTab, and the CVS DCs FTab. These
'                   links are needed to run the Trace scripts

' Requires:         dirstore.dbf file, tranship.dbf file, Demand
'                   Regions theme, CVS DCs theme exist

' Called by:        Any of the Trace scripts

' Calls:            Nil

' SELF:             Nil

' Returns:          Nil

'*****************************************************************
Scriptname = "Tables.Unlink"

' Get the tables to be linked

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
```

```
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

theStoreVTab.UnlinkAll
theD2SVTab.UnlinkAll
theDCVTab.UnlinkAll
theTSVTab.UnlinkAll

return Nil
```

```
'*****************************************************************

' Scriptname:    TotalFlowTheme.Make

' Filename:      totalflo.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a Flow theme based on the Total
'                Flow field in the DC-to-Region Flow theme table.
'                Zero value flows are made invisible using the
'                null value and symbol.

' Requires:      DC-to-Region flow theme must exist

' Called by:     View menu item click event ("Display Flows:
'                                    DC-to-Region by Total Flow")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "TotalFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("DC-to-Region Flow")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called DC-to-Region
                         Flow does not exist.", Scriptname)
  exit
end

catString = "Total Flow"

checkTheme = theView.FindTheme(catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of"++catString)
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the DC-to-Region Flow theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend
newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)

' Make zero the null value

newLegend.SetNullValue(catString, 0)

' Select a color from the color palette to be used
' in drawing the new line theme

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with into five natural breaks
' and weight the line thickness by the flow volume

newLegend.Natural(newTheme, catString, 5)
theSymbolList = newLegend.GetSymbols
thickness = 1
count = 0
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
  end
theSymbolList.UniformColor(theColor)

' Make the null symbol transparent

nullSym = Symbol.Make(#SYMBOL_PEN )
theNullColor = Color.GetBlue
theNullColor.SetTransparent(TRUE)
nullSym.SetColor(theNullColor)
newLegend.SetNullSymbol(nullSym)
newLegend.DisplayNoDataClass(FALSE)
newTheme.SetLegend(newLegend)
newTheme.SetName (catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme (newTheme)
newTheme.UpdateLegend
theView.Invalidate
theNullColor.SetTransparent(FALSE)

return Nil
```

```
'******************************************************************

' Scriptname:     TotDemRegTheme.Make

' Filename:       totdemre.ave

' Author:         Kenneth Bennett

' Date:           May 3, 1998

' Description:    Script generates a Demand Region theme based on
'                 the Total Demand field in the Demand Regions
'                 theme table.  The theme is classified into five
'                 sizes based on total demand and uses the outlined
'                 round symbol

' Requires:       DC-to-Region flow theme must exist

' Called by:      View menu item click event
'                 ("Display Flows: by Total Flow")

' Calls:          Nil

' SELF:           Nil

' Returns:        Nil

'******************************************************************
Scriptname = "TotalFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called Demand Regions
               does not exist.", Scriptname)
  exit
end

checkTheme = theView.FindTheme("Demand Regions by Total Demand")
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc
                ("Attributes of Demand Regions by Total Demand")
  if (theTable <> NIL) then
    av.GetProject.RemoveDoc(theTable)
  end
```

```
end

' Clone the Demand-to-Store theme

totTheme = theTheme.Clone
totTheme.SetName("Demand Regions by Total Demand")

' Change the legend to weight the symbol size
' by the total demand and classify into five
' groups using a  natural break

totLegend = totTheme.GetLegend
totLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
totLegend.Natural(totTheme, "Total Demand", 5)
totLegend.DisplayNoDataClass(FALSE)
' Get the project working directory
theDir = av.GetProject.GetWorkDir.AsString
thePath = theDir+"\default.avp"
theSymbolList = totLegend.GetSymbols
index = 0
increment = 0
for each s in theSymbolList
  thePalette = Palette.MakeFromFile(thePath.AsFileName)
  ' Grab the Marker palette and get the outlined round marker
  chosenMarker = thePalette.GetList(#PALETTE_LIST_MARKER).Get(7)
  chosenMarker.SetSize(10 + increment)
  theSymbolList.Set(index, chosenMarker)
  index = index + 1
  increment = increment + 2
end
theSymbolList.UniformColor(Color.GetBlue)

totTheme.UpdateLegend
totTheme.SetVisible( TRUE )
theView.AddTheme (totTheme)
theView.Invalidate

return Nil
```

```
'****************************************************************

' Scriptname:    TotLogTheme.Make

' Filename:      totlogth.ave

' Author:        Kenneth Bennett

' Date:          May 6, 1998

' Description:   Creates a pie chart theme of the Demand
'                Regions where the pie slices represent the
'                shipping, pick, transhipment, and crossdock
'                costs for all products, and the size of the
'                whole pie represents the total logistics cost.

' Requires:      Demand Regions theme and the respective logistics
'                cost table must exist.

' Called by:     View menu item click event ("Display Demand
'                Regions: by Total Logisics Cost")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'****************************************************************
Scriptname = "TotLogTheme.Make"

' Find the view and the Demand Regions theme

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theView.GetWin.Open
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
                Scriptname)
  exit
end
theFTab = theTheme.GetFTab

' Find the needed fields

oFld = theFTab.FindField("OTC_to_Store")
```

```
rFld = theFTab.FindField("Rx_to_Store")
cFld = theFTab.FindField("CW_to_Store")
totFld = theFTab.FindField("Total Demand")

if ((oFld = Nil) OR (rFld = Nil) OR (cFld = Nil) OR
                                (totFld = Nil)) then
  MsgBox.Error("ERROR: Require product fields are missing.",
               Scriptname)
  exit
end

oFld = oFld.AsString
rFld = rFld.AsString
cFld = cFld.AsString

fldStringList = {oFld, rFld, cFld}

' Check to see if the new theme already exists
checkTheme = theView.FindTheme
                         ("Demand Regions by Product Volume")
if (checkTheme <> Nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc
           ("Attributes of Demand Regions by Product Volume")
  if (theTable <> Nil) then
    av.GetProject.RemoveDoc(theTable)
  end
end


' Clone the theme and work with the new theme

demTheme = theTheme.Clone


' Get the new Demand Region theme's legend

demLegend = demTheme.GetLegend

'Create as many fill symbols as you have
'fieldNames and place them in a list.

otcsym = RasterFill.Make
otcsym.SetStyle(#RASTERFILL_STYLE_SOLID)
otcsym.SetColor(Color.GetBlue)
rxsym = RasterFill.Make
rxsym.SetStyle(#RASTERFILL_STYLE_SOLID)
rxsym.SetColor(Color.GetRed)
cwsym = RasterFill.Make
cwsym.SetStyle(#RASTERFILL_STYLE_SOLID)
cwsym.SetColor(Color.GetGreen)

theSyms = {otcsym, rxsym, cwsym}

' Make a background fill Symbol that is empty
```

```
BGsym = RasterFill.Make
BGsym.SetStyle(#RASTERFILL_STYLE_EMPTY)

' Create the New Legend

demLegend.PieChart(demTheme,fldStringList,
                   theSyms,BGSym,"Total Demand")

' To set a size field:

theSym = demLegend.GetSymbol(demLegend.ReturnFieldNames,
                             false)

theSym.SetMinSize(8)
theSym.SetMaxSize(24)

' Redraw the theme using the PieChart legend.

demTheme.UpdateLegend
demTheme.SetActive(FALSE)
demTheme.SetVisible(TRUE)
demTheme.SetName("Demand Regions by Product Volume")
theView.AddTheme(demTheme)
theView.Invalidate

return Nil
```

```
'*****************************************************************

' Scriptname:       TraceAll

' Filename:         traceall.ave

' Description:      Traces the total logistics costs for all
'                   product types (OTC + Rx + CW) in the
'                   demand region selected by the user.  The
'                   demand region is selected by clicking the
'                   mouse pointer on a demand region feature
'                   immediately after selecting the "A" tool
'                   button or the Demand Regions All Products
'                   item under the Trace Costs view menu.  The
'                   costs are shown to the user via a pop up
'                   dialog box.

' Requires:         dirstore.dbf file, tranship.dbf file, Demand
'                   Regions theme, CVS DCs theme must exist.

' Called by:        View menu item click event
'                   ("Trace Cost: Demand Regions All Products")
'                   or by a tool button apply event
'                   (button with the "A" icon in the toolbar)

' Calls:            Tables.Link, Tables.Unlink

' SELF:             Nil

' Returns:          Nil

'*****************************************************************
Scriptname = "TraceAll"

' Set the number format for all numbers in the script

Script.The.SetNumberFormat("d.dd")

' Next, link the necessary tables

av.Run("Tables.Link","")

' Now get the necessary tables

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theThemeList = theView.GetThemes
for each t in theThemeList
```

```
  t.SetActive(FALSE)
end
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theTheme.SetActive(true)
av.Run("View.SelectPoint","")
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

'-----------------------------------------------------------------
' First Get the OTC costs
'-----------------------------------------------------------------

' Get the bitmap for the Demand Regions VTab

theBStMap = theStoreVTab.GetSelection
if (theBStMap.Count = 0) then
  exit
end
thestorefld = theStoreVTab.FindField("Demand Region")
selList = theBStMap.AsList
theBStMap.ClearAll
theStoreVTab.UpdateSelection
ii = -1
for each jj in selList
  ii = ii + 1
  if (jj.Not) then
    continue
  end
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt1 = 0
  thetsflow1 = 0
  d2scost1 = 0
  pickcost1 = 0
  xdoccost1 = 0
  tscost1 = 0
  totalCost1 = 0
    if (theBStMap.Get(ii)) then
      store = theStoreVTab.ReturnValue(thestorefld,ii)
    end
  theBMap = theD2SVTab.GetSelection
  expr = "([OptimizedValue] > 0)" ++ "and" ++
                              "([Product] = ""OTC"")"
  theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
  theratefld = theD2SVTab.FindField("ActualRate")
  theamtfld = theD2SVTab.FindField("OptimizedValue")
  thed2sdcfld = theD2SVTab.FindField("Facility")
  for each i in theBMap
    if (theBMap.Get(i)) then
      theD2SVTab.UpdateSelection
      thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
      theamt1 = theD2SVTab.ReturnValue(theamtfld,i)
      d2scost1 = theD2SVTab.ReturnValue(theratefld,i) * theamt1
    end
    'for this flow, see what is happening at the DC
    theDCBMap = theDCVTab.GetSelection
    dcratefld = theDCVTab.FindField("OTC Rate")
    dcfld = theDCVTab.FindField("Facility")
    for each j in theDCBMap
      if (theDCBMap.Get(j)) then
            'a pick cost
          thedc = theDCVTab.ReturnValue(dcfld,j)
         if (thedc = thefacil) then
             pickcost1 = theamt1 * theDCVTab.ReturnValue
                                                (dcratefld,j)
         end
      end
    end
  end
  thetsflow1 = 0  'No transhipments of OTC products
  totalCost1 = d2scost1 + xdoccost1 + tscost1 + pickcost1
  theBStMap.Clear(ii)
  theStoreVTab.UpdateSelection
  theD2SVTab.UpdateSelection
  theDCBMap = theDCVTab.GetSelection
  theDCBMap.ClearAll
  theDCVTab.UpdateSelection
  theTSBMap = theTSVTab.GetSelection
  theTSBMap = theTSVTab.GetSelection
  theTSBMap.ClearAll
  theTSVTab.UpdateSelection
end ' on jj

'-----------------------------------------------------------------
' Next get the Rx costs
'-----------------------------------------------------------------
```

```
ii = -1
for each jj in selList
  ii = ii + 1
  if (jj.Not) then
    continue
  end
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt2 = 0
  thetsflow2 = 0
  d2scost2 = 0
  pickcost2 = 0
  xdoccost2 = 0
  tscost2 = 0
  totalCost2 = 0
  if (theBStMap.Get(ii)) then
     store = theStoreVTab.ReturnValue(thestorefld,ii)
  end
  theBMap = theD2SVTab.GetSelection
  expr = "([OptimizedValue] > 0)" ++ "and" ++
                                       "([Product] = ""Rx"")"
  theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
  theratefld = theD2SVTab.FindField("ActualRate")
  theamtfld = theD2SVTab.FindField("OptimizedValue")
  thed2sdcfld = theD2SVTab.FindField("Facility")
  for each i in theBMap
    if (theBMap.Get(i)) then
      theD2SVTab.UpdateSelection
      thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
      theamt2 = theD2SVTab.ReturnValue(theamtfld,i)
      d2scost2 = theD2SVTab.ReturnValue(theratefld,i) * theamt2
    end
    'for this flow, see what is happening at the DC
    theDCBMap = theDCVTab.GetSelection
    hasrx = theDCVTab.FindField("HasRx")
    dcratefld = theDCVTab.FindField("Rx Rate")
    dcfld = theDCVtab.FindField("Facility")
    for each j in theDCBMap
      if (theDCBMap.Get(j)) then
        thedc = theDCVTab.ReturnValue(dcfld,j)
        if(thedc <> thefacil) then
           continue
        end
            rxthere = theDCVTab.ReturnValue(hasrx,j)
        if (rxthere = 0) then
             xdoccost2 = theamt2 * theDCVTab.ReturnValue
                                                (dcratefld,j)
           'put traceback to tranship here
             theTSBMap = theTSVTab.GetSelection
             expr = "([OptimizedValue] > 0)" ++ "and" ++
                                          "([Product] = ""Rx"")"
              theTSVTab.Query(expr, theTSBMap, #VTAB_SELTYPE_AND)
              theTSVTab.UpdateSelection
              thetsratefld = theTSVTab.FindField("ActualRate")
              theorigfld = theTSVTab.FindField("OriginFacility")
              theflowfld = theTSVTab.FindField("OptimizedValue")
              thedcfld = theDCVTab.FindField("Facility")
              'get total flow into xdoc dc
              thetsflow2 = 0
              for each k in theTSBMap
                if (theTSBMap.Get(k)) then
                  thetsflow2 = thetsflow2 +
                               theTSVTab.ReturnValue(theflowfld,k)
                end
              end
              'get average cost per unit
              therate = 0
              for each k in theTSBMap
               if (theTSBMap.Get(k)) then
                  theratio = theTSVTab.ReturnValue(theflowfld,k)
                                                      /thetsflow2
                 therate = therate + theTSVTab.ReturnValue
                           (thetsratefld,k) * theratio
                  theorig = theTSVTab.ReturnValue(theorigfld,k)
               end
              end
              tscost2 = theamt2 * therate
              pickcostrate = 0
              for each k in theTSBMap
                if (theTSBMap.Get(k)) then
                 theorig = theTSVTab.ReturnValue(theorigfld,k)
                 theratio = theTSVTab.ReturnValue
                                     (theflowfld,k)/thetsflow2
              for each m in theDCVTab
                   thefacil2 = theDCVTab.ReturnValue
                                                (thedcfld,m)
                   if (thefacil2 = theorig) then
                       pickcostrate = pickcostrate +
                       theDCVTab.ReturnValue(dcratefld,m)
                       * theratio
                     end
              end
               end
          end
              pickcost2 = theamt2 * pickcostrate
           else 'a pick cost
             pickcost2 = theamt2 * theDCVTab.ReturnValue
                                          (dcratefld,j)
           end
      end
    end
  end
  totalCost2 = d2scost2 + xdoccost2 + tscost2 + pickcost2
  theBStMap.Clear(ii)
  theStoreVTab.UpdateSelection
  theBMap.ClearAll
  theD2SVTab.UpdateSelection
  theDCBMap = theDCVTab.GetSelection
  theDCBMap.ClearAll
  theDCVTab.UpdateSelection
  theTSBMap = theTSVTab.GetSelection
  theTSBMap = theTSVTab.GetSelection
  theTSBMap.ClearAll
  theTSVTab.UpdateSelection
end ' on jj

'-----------------------------------------------------------------
' Finally get the CW costs
'-----------------------------------------------------------------

ii = -1
for each jj in selList
  ii = ii + 1
  if (jj.Not) then
    continue
  end
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt3 = 0
  thetsflow3 = 0
  d2scost3 = 0
  pickcost3 = 0
  xdoccost3 = 0
  tscost3 = 0
  totalCost3 = 0
  if (theBStMap.Get(ii)) then
    store = theStoreVTab.ReturnValue(thestorefld,ii)
  end
  theBMap = theD2SVTab.GetSelection
  expr = "([OptimizedValue] > 0)" ++ "and" ++
                                  "([Product] = ""CW"")"
  theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
  theratefld = theD2SVTab.FindField("ActualRate")
  theamtfld = theD2SVTab.FindField("OptimizedValue")
  thed2sdcfld = theD2SVTab.FindField("Facility")
  for each i in theBMap
    if (theBMap.Get(i)) then
      theD2SVTab.UpdateSelection
      thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
      theamt3 = theD2SVTab.ReturnValue(theamtfld,i)
      d2scost3 = theD2SVTab.ReturnValue(theratefld,i) * theamt3
    end
    'for this flow, see what is happening at the DC
    theDCBMap = theDCVTab.GetSelection
    hascw = theDCVTab.FindField("HasCW")
    dcratefld = theDCVTab.FindField("CW Rate")
    otcratefld = theDCVTab.FindField("OTC Rate")
    dcfld = theDCVTab.FindField("Facility")
    for each j in theDCBMap
      if (theDCBMap.Get(j)) then
        thedc = theDCVTab.ReturnValue(dcfld,j)
        if (thedc <> thefacil) then
           continue
        end
            cwthere = theDCVTab.ReturnValue(hascw,j)
            if (cwthere = 0) then
              xdoccost3 = theamt3 * theDCVTab.ReturnValue
                                                   (dcratefld,j)
        'put traceback to tranship here
          theTSBMap = theTSVTab.GetSelection
          expr = "([OptimizedValue] > 0)" ++ "and" ++
                                      "([Product] = ""CW"")"
           theTSVTab.Query(expr, theTSBMap, #VTAB_SELTYPE_AND)
           theTSVTab.UpdateSelection
           thetsratefld = theTSVTab.FindField("ActualRate")
           theorigfld = theTSVTab.FindField("OriginFacility")
           theflowfld = theTSVTab.FindField("OptimizedValue")
           thedcfld = theDCVTab.FindField("Facility")
           'get total flow into xdoc dc
           thetsflow3 = 0
           for each k in theTSBMap
             if (theTSBMap.Get(k)) then
               thetsflow3 = thetsflow3 +
                           theTSVTab.ReturnValue(theflowfld,k)
             end
           end
           'get average cost per unit
           therate = 0
           for each k in theTSBMap
            if (theTSBMap.Get(k)) then
               theratio = theTSVTab.ReturnValue(theflowfld,k)
                                                /thetsflow3
              therate = therate + theTSVTab.ReturnValue
                                (thetsratefld,k) * theratio
               theorig = theTSVTab.ReturnValue(theorigfld,k)
            end
           end
           tscost3 = theamt3 * therate
           pickcostrate = 0
           for each k in theTSBMap
             if (theTSBMap.Get(k)) then
              theorig = theTSVTab.ReturnValue(theorigfld,k)
              theratio = theTSVTab.ReturnValue(theflowfld,k)
                                                /thetsflow3
           for each m in theDCVTab
                     thefacil2 = theDCVTab.ReturnValue
                                                (thedcfld,m)
                     if (thefacil2 = theorig) then
                         pickcostrate = pickcostrate +
                         theDCVTab.ReturnValue(otcratefld,m)
                         * theratio
                       end
           end
            end
       end
           pickcost3 = theamt3 * pickcostrate
        else 'a pick cost
          pickcost3 = theamt3 * theDCVTab.ReturnValue
                                           (otcratefld,j)
        end
    end
  end
end
totalCost3 = d2scost3 + xdoccost3 + tscost3 + pickcost3
```

```
    theBStMap.Clear(ii)
    theStoreVTab.UpdateSelection
    theBMap.ClearAll
    theD2SVTab.UpdateSelection
    theDCBMap = theDCVTab.GetSelection
    theDCBMap.ClearAll
    theDCVTab.UpdateSelection
    theTSBMap = theTSVTab.GetSelection
    theTSBMap = theTSVTab.GetSelection
    theTSBMap.ClearAll
    theTSVTab.UpdateSelection
  end ' on jj
  av.Run("Tables.Unlink","")

  '-----------------------------------------------------------------
  ' Sum the DC-to-Region flow and the transhipment
  ' flow amounts, and also the cost amount for each
  ' component and issue the report.
  '-----------------------------------------------------------------

  theamt4 = theamt1 + theamt2 + theamt3
  thetsflow4 = thetsflow1 + thetsflow2 + thetsflow3
  d2scost4 = d2scost1 + d2scost2 + d2scost3
  xdoccost4 = xdoccost1 + xdoccost2 + xdoccost3
  tscost4 = tscost1 + tscost2 + tscost3
  pickcost4 = pickcost1 + pickcost2 + pickcost3
  totalCost4 = totalCost1 + totalCost2 + totalCost3

  ' Make the report string

  therepstr = "Trace type: All Products"+nl+"Demand Region:"++
              store+nl+"Demand for all products:"++theamt4.AsString
              +nl+"Shipping cost from"++thefacil++":"++
              d2scost4.AsString+nl+"Total transhipment flow:"
              ++thetsflow4.AsString+nl+"Total transhipment cost:"
              ++tscost4.AsString+nl+"Total crossdock cost:"++
              xdoccost4.AsString+nl+"Total pick cost:"++
              pickcost4.AsString+nl+nl+"Total Logistics cost for all
              products:"++totalCost4.AsString+nl

  ' Call up the report

  MsgBox.Report(therepstr,
                "Total Logistics Cost To Serve Demand Region"++
                store.AsString)

  return Nil
```

```
'*****************************************************************

' Scriptname:       TraceCW

' Filename:         tracecw.ave

' Description:      Traces the total CW logistics costs for a
'                   demand region selected by the user.  The
'                   demand region is selected by clicking the
'                   mouse pointer on a demand region feature
'                   immediately after selecting the "C" tool
'                   button or the Demand Regions CW Only item
'                   under the Trace Costs view menu.  The costs
'                   are shown to the user via a pop up dialog
'                   box.

' Requires:         dirstore.dbf file, tranship.dbf file, Demand
'                   Regions theme, CVS DCs theme must exist.

' Called by:        View menu item click event
'                   ("Trace Cost: Demand Regions CW Only")
'                   or by a tool button apply event
'                   (button with the "C" icon in the toolbar)

' Calls:            Tables.Link, Tables.Unlink

' SELF:             Nil

' Returns:          Nil

'*****************************************************************
Scriptname = "TraceCW"

' Set the number format for all numbers in the script

Script.The.SetNumberFormat("d.dd")

' Next, link the necessary tables

av.Run("Tables.Link","")

' Now get the necessary tables

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theThemeList = theView.GetThemes
for each t in theThemeList
  t.SetActive(FALSE)
```

```
end
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theTheme.SetActive(true)
av.Run("View.SelectPoint","")
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

' Get the bitmap for the Demand Regions VTab

theBStMap = theStoreVTab.GetSelection
if (theBStMap.Count = 0) then
  exit
end
thestorefld = theStoreVTab.FindField("Demand Region")
selList = theBStMap.AsList
theBStMap.ClearAll
theStoreVTab.UpdateSelection
ii = -1
for each jj in selList
  ii = ii + 1
  if (jj.Not) then
    continue
  end
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt = 0
  thetsflow = 0
  d2scost = 0
  pickcost = 0
  xdoccost = 0
  tscost = 0
```

```
totalCost = 0
therepstr = "Trace type: CW Products Only"+nl
if (theBStMap.Get(ii)) then
  store = theStoreVTab.ReturnValue(thestorefld,ii)
  therepstr = therepstr + "Demand Region:" ++ store + nl
end
theBMap = theD2SVTab.GetSelection
expr = "([OptimizedValue] > 0)" ++ "and" ++
                     "([Product] = ""CW"")"
theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
theratefld = theD2SVTab.FindField("ActualRate")
theamtfld = theD2SVTab.FindField("OptimizedValue")
thed2sdcfld = theD2SVTab.FindField("Facility")
for each i in theBMap
  if (theBMap.Get(i)) then
    theD2SVTab.UpdateSelection
    thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
    theamt = theD2SVTab.ReturnValue(theamtfld,i)
    therepstr = therepstr+ "Demand for CW products:" ++
                theamt.AsString+nl
    d2scost = theD2SVTab.ReturnValue(theratefld,i) * theamt
    therepstr = therepstr + "Shipping cost from" ++ thefacil ++
                ":"++d2scost.AsString+nl
  end
  'for this flow, see what is happening at the DC
  theDCBMap = theDCVTab.GetSelection
  hascw = theDCVTab.FindField("HasCW")
  dcratefld = theDCVTab.FindField("CW Rate")
  dcfld = theDCVTab.FindField("Facility")
  for each j in theDCBMap
    if (theDCBMap.Get(j)) then
      thedc = theDCVTab.ReturnValue(dcfld,j)
      if (thedc <> thefacil) then
         continue
      end
          cwthere = theDCVTab.ReturnValue(hascw,j)
          if (cwthere = 0) then
            xdoccost = theamt * theDCVTab.ReturnValue
                                          (dcratefld,j)
        therepstr = therepstr + "Crossdock cost at" ++
                    thefacil ++ ":"++
                    xdoccost.AsString +nl
         'put traceback to tranship here
           theTSBMap = theTSVTab.GetSelection
           expr = "([OptimizedValue] > 0)" ++ "and" ++
                  "([Product] = ""CW"")"
            theTSVTab.Query(expr, theTSBMap, #VTAB_SELTYPE_AND)
            theTSVTab.UpdateSelection
            thetsratefld = theTSVTab.FindField("ActualRate")
            theorigfld = theTSVTab.FindField("OriginFacility")
            theflowfld = theTSVTab.FindField("OptimizedValue")
            thedcfld = theDCVTab.FindField("Facility")
            'get total flow into xdoc dc
            thetsflow = 0
            for each k in theTSBMap
              if (theTSBMap.Get(k)) then
            thetsflow = thetsflow +
                        theTSVTab.ReturnValue(theflowfld,k)
          end
        end
        'get average cost per unit
        therate = 0
        for each k in theTSBMap
         if (theTSBMap.Get(k)) then
            theratio = theTSVTab.ReturnValue(theflowfld,k)
                                               /thetsflow
           therate = therate + theTSVTab.ReturnValue
                     (thetsratefld,k) * theratio
            theorig = theTSVTab.ReturnValue(theorigfld,k)
         end
        end
        tscost = theamt * therate
        if (theTSBMap.Count > 1) then
       therepstr = therepstr+
                   "Weighted Average Transship Cost from"++
                   theTSBMap.count.AsString++
                   "origins to"++thefacil++
                   ":"++tscost.AsString+nl
        else
       therepstr = therepstr+"Transshipment cost from"++
                   theorig++"to"++thefacil++":"++
                   tscost.AsString+nl
    end
        pickcostrate = 0
        for each k in theTSBMap
          if (theTSBMap.Get(k)) then
           theorig = theTSVTab.ReturnValue(theorigfld,k)
           theratio = theTSVTab.ReturnValue(theflowfld,k)
                                              /thetsflow
        for each m in theDCVTab
                  thefacil2 = theDCVTab.ReturnValue
                                              (thedcfld,m)
                  if (thefacil2 = theorig) then
                      pickcostrate = pickcostrate +
                      theDCVTab.ReturnValue(dcratefld,m)
                      * theratio
                    end
        end
         end
    end
        pickcost = theamt*pickcostrate
       if (theTSBMap.Count > 1) then
      therepstr = therepstr+
                  "Weighted average pick cost at"++
                  theTSBMap.count.AsString++
                  "origins:"++pickcost.AsString+nl
       else
      therepstr = therepstr+"Pick cost at origin DC:"++
                  pickcost.AsString+nl
       end
     else 'a pick cost
       pickcost = theamt * theDCVTab.ReturnValue
```

```
                                                         (dcratefld,j)
             therepstr = therepstr+"Pick cost at "+thefacil++":"++
                         pickcost.AsString+nl
             end
         end
       end
     end
     totalCost = d2scost + xdoccost + tscost + pickcost
     therepstr = therepstr+nl+"Total Logistics Cost for CW Products:"
                 ++totalCost.AsString+nl
     MsgBox.Report(therepstr,
                  "Total Logisitic Costs to Serve Demand Region"
                  ++store.AsString)
     av.Run("Tables.Unlink","")
     theBStMap.Clear(ii)
     theStoreVTab.UpdateSelection
     theBMap.ClearAll
     theD2SVTab.UpdateSelection
     theDCBMap = theDCVTab.GetSelection
     theDCBMap.ClearAll
     theDCVTab.UpdateSelection
     theTSBMap = theTSVTab.GetSelection
     theTSBMap = theTSVTab.GetSelection
     theTSBMap.ClearAll
     theTSVTab.UpdateSelection
   end ' on jj

   return Nil
```

```
'*******************************************************************

' Scriptname:     TraceCWAll

' Filename:       tracecwa.ave

' Description:    Script finds the total CW logistics cost
'                 for each demand region and writes it to a
'                 new dBase file called CWLgCst.dbf.  This
'                 file is then joined to the Demand Regions
'                 table.

' Requires:       dirstore.dbf, tranship.dbf, Demand Regions
'                 theme, CVS DCs theme exists

' Called by:      View menu click event
                  ("Trace Costs: Chain-wide CW Only")

' Calls:           Tables.Link, Tables.Unlink

' SELF:           Nil

' Returns:        Nil

'*******************************************************************
Scriptname = "TraceCWAll"

' Warn user about time to complete this script

resume = MsgBox.YesNo("This trace takes approximately 10 minutes."
                     +nl+"Do you want to continue?",
                     "Trace Costs: Chain-wide CW Only", FALSE)
if (resume = false) then
  exit
end

' Set the number format for the script

Script.The.SetNumberFormat("d.dd")

'  Get the VTabs to be used and get the working directory
av.Run("Tables.Link","")
theDirectory = av.GetProject.GetWorkDir.AsString

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Demand Regions")
```

```
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

' Get the bitmap for the Demand Regions VTab

theBStMap = theStoreVTab.GetSelection

'check if table exists
sumcwcst_exists = (av.GetProject.FindDoc("CWLgCst.dbf")
                   = NIL).Not
skip = 0
if (sumcwcst_exists) then
  thedoc = av.GetProject.FindDoc("CWLgCst.dbf")
  if (MsgBox.YesNo("Overwrite existing logistics cost table?",
    "The Table CWLgCst already exists",false)) then
    if (sumcwcst_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    skip = 1
  end
end
'create a newtable
if (skip = 0) then
  flnm = theDirectory + "/CWLgCst.dbf"
  newVTab    = VTab.MakeNew( flnm.AsFileName, dBase )
  storefld   = Field.Make ("DemRegion",#FIELD_CHAR,16, 0)
  directfld = Field.Make ("CWDrctCst",#FIELD_DECIMAL,16,2)
  pickfld    = Field.Make ("CWPickCst",#FIELD_DECIMAL,16,2)
  transfld   = Field.Make ("CWTranCst",#FIELD_DECIMAL,16,2)
  xdocfld    = Field.Make ("CWXdocCst",#FIELD_DECIMAL,16,2)
```

```
totfld    = Field.Make ("CWTotlCst",#FIELD_DECIMAL,16,2)
newVTab.AddFields({storefld, directfld,pickfld,
                  transfld,xdocfld,totfld})
storefld.SetAlias("Demand Region")
directfld.SetAlias("CW Direct Cost")
pickfld.SetAlias("CW Pick Cost")
transfld.SetAlias("CW Tranship Cost")
xdocfld.SetAlias("CW Crossdock Cost")
totfld.SetAlias("CW Total Cost")
theBStMap.ClearAll
thestorefld = theStoreVTab.FindField("Demand Region")
numrecs = theStoreVTab.GetNumRecords
for each ii in theStoreVTab
  'Clear the bitmap
  theBSTMap.ClearAll
  ' Set the record in the demand regions table
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt = 0
  thetsflow = 0
  d2scost = 0
  pickcost = 0
  xdoccost = 0
  tscost = 0
  if (theBStMap.Get(ii)) then
      store = theStoreVTab.ReturnValue(thestorefld,ii)
  end
  ' Get the dirstore.dbf record selected by the link
  theBMap = theD2SVTab.GetSelection
  ' Reselect those record with CW flow greater than zero
  expr = "([OptimizedValue] > 0)" ++ "and" ++
         "([Product] = ""CW"")"
  theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
  theratefld = theD2SVTab.FindField("ActualRate")
  theamtfld = theD2SVTab.FindField("OptimizedValue")
  thed2sdcfld = theD2SVTab.FindField("Facility")
  ' Loop through selected set in dirstore.dbf
  for each i in theBMap
    if (theBMap.Get(i)) then
      theD2SVTab.UpdateSelection
      thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
      theamt = theD2SVTab.ReturnValue(theamtfld,i)
      d2scost = theD2SVTab.ReturnValue(theratefld,i) * theamt
    end
    'for this flow, see what is happening at the DC by
    'getting the selected set in the CVS DCs table
    theDCBMap = theDCVTab.GetSelection
    hascw = theDCVTab.FindField("HasCW")
    dcratefld = theDCVTab.FindField("CW Rate")
    dcfld = theDCVTab.FindField("Facility")
    for each j in theDCBMap
      if (theDCBMap.Get(j)) then
        thedc = theDCVTab.ReturnValue(dcfld,j)
        if (thedc <> thefacil) then
           continue
        end
```

```
 cwthere = theDCVTab.ReturnValue(hascw,j)
 if (cwthere = 0) then
   xdoccost = theamt * theDCVTab.ReturnValue
              (dcratefld,j)
      'put traceback to tranship here
  theTSBMap = theTSVTab.GetSelection
  expr = "([OptimizedValue] > 0)" ++ "and" ++
         "([Product] = ""CW"")"
   theTSVTab.Query(expr, theTSBMap, #VTAB_SELTYPE_AND)
   theTSVTab.UpdateSelection
   thetsratefld = theTSVTab.FindField("ActualRate")
   theorigfld = theTSVTab.FindField("OriginFacility")
   theflowfld = theTSVTab.FindField("OptimizedValue")
   thedcfld = theDCVTab.FindField("Facility")
   'get total flow into xdoc dc
   thetsflow = 0
   for each k in theTSBMap
     if (theTSBMap.Get(k)) then
       thetsflow = thetsflow +
                   theTSVTab.ReturnValue(theflowfld,k)
     end
   end
   'get average cost per unit
   therate = 0
   for each k in theTSBMap
    if (theTSBMap.Get(k)) then
       theratio = theTSVTab.ReturnValue(theflowfld,k)
                                       /thetsflow
      therate = therate + theTSVTab.ReturnValue
               (thetsratefld,k) * theratio
       theorig = theTSVTab.ReturnValue(theorigfld,k)
    end
   end
   tscost = theamt * therate
   pickcostrate = 0
   for each k in theTSBMap
     if (theTSBMap.Get(k)) then
      theorig = theTSVTab.ReturnValue(theorigfld,k)
      theratio = theTSVTab.ReturnValue(theflowfld,k)
                                     /thetsflow
         for each m in theDCVTab
             thefacil2 = theDCVTab.ReturnValue
                                        (thedcfld,m)
             if (thefacil2 = theorig) then
                 pickcostrate = pickcostrate +
                 theDCVTab.ReturnValue(dcratefld,m)
                 * theratio
               end
         end
    end
 end
   pickcost = theamt * pickcostrate
else 'a pick cost
  pickcost = theamt *
             theDCVTab.ReturnValue(dcratefld,j)
end
```

```
            end
          end
        end
        theBStMap.Clear(ii)
        theStoreVTab.UpdateSelection
        newrec = newVtab.AddRecord
        newVTab.SetValue(storefld,newrec,store)
        newVTab.SetValue(directfld,newrec,d2scost)
        newVTab.SetValue(pickfld,newrec,pickcost)
        newVTab.SetValue(transfld,newrec,tscost)
        newVTab.SetValue(xdocfld,newrec,xdoccost)
        totcost = d2scost+pickcost+tscost+xdoccost
        newVTab.SetValue(totfld,newrec,totcost)
      end ' on ii
    end 'if skip = 0
    theBStMap.ClearAll
    theStoreVTab.UpdateSelection
    theBMap = theD2SVTab.GetSelection
    theBMap.ClearAll
    theD2SVTab.UpdateSelection
    theBMap = theDCVTab.GetSelection
    theBMap.ClearAll
    theDCVTab.UpdateSelection
    theBMap = theTSVTab.GetSelection
    theBMap.ClearAll
    theTSVTab.UpdateSelection
    av.Run("Tables.Unlink","")

    ' Set the new table to uneditable and write to file

    newVTab.SetEditable(FALSE)
    newVTab.Flush

    checkTable = av.GetProject.FindDoc("CW Logistics Costs")
    if (checkTable <> Nil) then
      av.GetProject.RemoveDoc(checkTable)
    end

    ' Bring the new table into the project

    newTable = Table.Make(newVTab)
    newTable.SetName("CW Logistics Costs")
    av.GetProject.AddDoc(newTable)

    ' Join the newTable to the Demand Regions table

    theStoreVTab.Join(thestorefld, newVTab, storefld)

    MsgBox.Info("Tracing of CW logistics cost for"+nl+
                "each demand region is complete.",
                "Trace Costs: Chain-wide CW Only")

    return Nil
```

```
'*****************************************************************

' Scriptname:    TraceOTCAll

' Filename:      traceotc.ave

' Description:   Script finds the total OTC logistics cost
'                for each demand region and writes it to a
'                new dBase file called OTCLgCst.dbf.  This
'                file is then joined to the Demand Regions
'                table.

' Requires:      dirstore.dbf, tranship.dbf, Demand Regions
'                theme, CVS DCs theme exists

' Called by:     View menu click event
'                ("Trace Costs: Chain-wide OTC Only")

' Calls:          Tables.Link, Tables.Unlink

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "TraceOTCAll"

' Warn user about time to complete this script

resume = MsgBox.YesNo("This trace takes approximately 10 minutes."
                      +nl+"Do you want to continue?",
                      "Trace Costs: Chain-wide OTC Only", FALSE)
if (resume = false) then
  exit
end

' Set the number format for the script

Script.The.SetNumberFormat("d.dd")


'  Get the VTabs to be used and get the working directory
av.Run("Tables.Link","")
theDirectory = av.GetProject.GetWorkDir.AsString

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
```

```
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

' Get the bitmap from the Demand Regions VTab

theBStMap = theStoreVTab.GetSelection
sumotccst_exists = (av.GetProject.FindDoc("OTCLgCst.dbf")
                    = NIL).Not
skip = 0
if (sumotccst_exists) then
  thedoc = av.GetProject.FindDoc("OTCLgCst.dbf")
  if (MsgBox.YesNo("Overwrite existing logistics cost table?",
    "The Table OTCLgCst already exists",false)) then
    if (sumotccst_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    skip = 1
  end
end
'create a newtable
if (skip = 0) then
  flnm = theDirectory + "/OTCLgCst.dbf"
  newVTab    = VTab.MakeNew( flnm.AsFileName, dBase )
  storefld   = Field.Make ("DemRegion",#FIELD_CHAR,5, 0)
  directfld  = Field.Make ("OTCDrctCst",#FIELD_DECIMAL,16,2)
  pickfld    = Field.Make ("OTCPickCst",#FIELD_DECIMAL,16,2)
  transfld   = Field.Make ("OTCTranCst",#FIELD_DECIMAL,16,2)
  xdocfld    = Field.Make ("OTCXdocCst",#FIELD_DECIMAL,16,2)
  totfld     = Field.Make ("OTCTotlCst",#FIELD_DECIMAL,16,2)
```

```
newVTab.AddFields({storefld, directfld,pickfld,
                   transfld,xdocfld,totfld})
storefld.SetAlias("Demand Region")
directfld.SetAlias("OTC Direct Cost")
pickfld.SetAlias("OTC Pick Cost")
transfld.SetAlias("OTC Tranship Cost")
xdocfld.SetAlias("OTC Crossdock Cost")
totfld.SetAlias("OTC Total Cost")
theBStMap.ClearAll
thestorefld = theStoreVTab.FindField("Demand Region")
numrecs = theStoreVTab.GetNumRecords
for each ii in theStoreVTab
  theBStMap.ClearAll
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt = 0
  thetsflow = 0
  d2scost = 0
  pickcost = 0
  xdoccost = 0
  tscost = 0
  if (theBStMap.Get(ii)) then
      store = theStoreVTab.ReturnValue(thestorefld,ii)
  end
  theBMap = theD2SVTab.GetSelection
  expr = "([OptimizedValue] > 0)" ++ "and" ++
         "([Product] = ""OTC"")"
  theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
  theratefld = theD2SVTab.FindField("ActualRate")
  theamtfld = theD2SVTab.FindField("OptimizedValue")
  thed2sdcfld = theD2SVTab.FindField("Facility")
  for each i in theBMap
    if (theBMap.Get(i)) then
      theD2SVTab.UpdateSelection
      thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
      theamt = theD2SVTab.ReturnValue(theamtfld,i)
      d2scost = theD2SVTab.ReturnValue(theratefld,i) * theamt
    end
    'for this flow, see what is happening at the DC
    theDCBMap = theDCVTab.GetSelection
    dcratefld = theDCVTab.FindField("OTC Rate")
    dcfld = theDCVTab.FindField("Facility")
    for each j in theDCBMap
      if (theDCBMap.Get(j)) then
          'a pick cost
          thedc = theDCVTab.ReturnValue(dcfld,j)
          if (thedc <> thefacil) then
             continue
          end
           pickcost = theamt *
                      theDCVTab.ReturnValue(dcratefld,j)
      end
    end
  end
  theBStMap.Clear(ii)
  theStoreVTab.UpdateSelection
```

```
    newrec = newVTab.AddRecord
      newVTab.SetValue(storefld,newrec,store)
      newVTab.SetValue(directfld,newrec,d2scost)
      newVTab.SetValue(pickfld,newrec,pickcost)
      newVTab.SetValue(transfld,newrec,tscost)
      newVTab.SetValue(xdocfld,newrec,xdoccost)
      totcost = d2scost+pickcost+tscost+xdoccost
      newVTab.SetValue(totfld,newrec,totcost)
  end ' on ii
end ' on if skip = 0
theBStMap.ClearAll
theStoreVTab.UpdateSelection
theBMap = theD2SVTab.GetSelection
theBMap.ClearAll
theD2SVTab.UpdateSelection
theBMap = theDCVTab.GetSelection
theBMap.ClearAll
theDCVTab.UpdateSelection
theBMap = theTSVTab.GetSelection
theBMap.ClearAll
theTSVTab.UpdateSelection
av.Run("Tables.Unlink","")

' Set the new table to uneditable and write to file

newVTab.SetEditable(FALSE)
newVTab.Flush

checkTable = av.GetProject.FindDoc("OTC Logistics Costs")
if (checkTable <> Nil) then
  av.GetProject.RemoveDoc(checkTable)
end

' Bring the new table into the project

newTable = Table.Make(newVTab)
newTable.SetName("OTC Logistics Costs")
av.GetProject.AddDoc(newTable)

' Join the newTable to the Demand Regions table

theStoreVTab.Join(thestorefld, newVTab, storefld)

MsgBox.Info("Tracing of OTC logistics cost for"+nl+
            "each demand region is complete.",
            "Trace Costs: Chain-wide OTC Only")

return Nil
```

```
'*****************************************************************

' Scriptname:      TraceRx

' Filename:        tracerx.ave

' Description:     Traces the total Rx logistics costs for a
'                  demand region selected by the user.  The
'                  demand region is selected by clicking the
'                  mouse pointer on a demand region feature
'                  immediately after selecting the "R" tool
'                  button or the Demand Regions Rx Only item
'                  under the Trace Costs view menu.  The costs
'                  are shown to the user via a pop up dialog
'                  box.

' Requires:        dirstore.dbf file, tranship.dbf file, Demand
'                  Regions theme, CVS DCs theme must exist.

' Called by:       View menu item click event
'                  ("Trace Cost: Demand Regions Rx Only")
'                  or by a tool button apply event
'                  (button with the "R" icon in the toolbar)

' Calls:           Tables.Link, Tables.Unlink

' SELF:            Nil

' Returns:         Nil

'*****************************************************************
Scriptname = "TraceRx"

' Set the number format for all numbers in the script

Script.The.SetNumberFormat("d.dd")

' Next, link the necessary tables

av.Run("Tables.Link","")

' Now get the necessary tables

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end
theThemeList = theView.GetThemes
for each t in theThemeList
  t.SetActive(FALSE)
```

```
end
theTheme = theView.FindTheme("Demand Regions")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theTheme.SetActive(true)
av.Run("View.SelectPoint","")
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

' Get the bitmap for the Demand Regions VTab

theBStMap = theStoreVTab.GetSelection
if (theBStMap.Count = 0) then
  exit
end
thestorefld = theStoreVTab.FindField("Demand Region")
selList = theBStMap.AsList
theBStMap.ClearAll
theStoreVTab.UpdateSelection
ii = -1
for each jj in selList
  ii = ii + 1
  if (jj.Not) then
    continue
  end
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt = 0
  thetsflow = 0
  d2scost = 0
  pickcost = 0
  xdoccost = 0
  tscost = 0
```

```
totalCost = 0
therepstr = "Trace type: Rx Products Only"+nl
if (theBStMap.Get(ii)) then
   store = theStoreVTab.ReturnValue(thestorefld,ii)
  therepstr = therepstr + "Demand Region:" ++ store + nl
end
theBMap = theD2SVTab.GetSelection
expr = "([OptimizedValue] > 0)" ++ "and" ++ "([Product]
         = ""Rx"")"
theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
theratefld = theD2SVTab.FindField("ActualRate")
theamtfld = theD2SVTab.FindField("OptimizedValue")
thed2sdcfld = theD2SVTab.FindField("Facility")
for each i in theBMap
  if (theBMap.Get(i)) then
    theD2SVTab.UpdateSelection
    thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
    theamt = theD2SVTab.ReturnValue(theamtfld,i)
    therepstr = therepstr+ "Demand for Rx products:" ++
                theamt.AsString+nl
    d2scost = theD2SVTab.ReturnValue(theratefld,i) * theamt
    therepstr = therepstr + "Shipping cost from" ++ thefacil
                ++":"++d2scost.AsString+nl
  end
  'for this flow, see what is happening at the DC
  theDCBMap = theDCVTab.GetSelection
  hasrx = theDCVTab.FindField("HasRx")
  dcratefld = theDCVTab.FindField("Rx Rate")
  dcfld = theDCVtab.FindField("Facility")
  for each j in theDCBMap
    if (theDCBMap.Get(j)) then
      thedc = theDCVTab.ReturnValue(dcfld,j)
      if(thedc <> thefacil) then
         continue
      end
          rxthere = theDCVTab.ReturnValue(hasrx,j)
      if (rxthere = 0) then
            xdoccost = theamt * theDCVTab.ReturnValue
                                           (dcratefld,j)
        therepstr = therepstr + "Crossdock cost at" ++
                    thefacil ++ ":"++xdoccost.AsString +nl
         'put traceback to tranship here
           theTSBMap = theTSVTab.GetSelection
           expr = "([OptimizedValue] > 0)" ++ "and" ++
                  "([Product] = ""Rx"")"
            theTSVTab.Query(expr, theTSBMap, #VTAB_SELTYPE_AND)
            theTSVTab.UpdateSelection
            thetsratefld = theTSVTab.FindField("ActualRate")
            theorigfld = theTSVTab.FindField("OriginFacility")
            theflowfld = theTSVTab.FindField("OptimizedValue")
            thedcfld = theDCVTab.FindField("Facility")
            'get total flow into xdoc dc
            thetsflow = 0
            for each k in theTSBMap
              if (theTSBMap.Get(k)) then
                thetsflow = thetsflow +
                       theTSVTab.ReturnValue(theflowfld,k)
          end
        end
        'get average cost per unit
        therate = 0
        for each k in theTSBMap
         if (theTSBMap.Get(k)) then
            theratio = theTSVTab.ReturnValue(theflowfld,k)
                                               /thetsflow
           therate = therate +
                     theTSVTab.ReturnValue(thetsratefld,k)
                     * theratio
            theorig = theTSVTab.ReturnValue(theorigfld,k)
         end
        end
        tscost = theamt * therate
        if (theTSBMap.Count > 1) then
       therepstr = therepstr+
                   "Weighted Average Transship Cost from"
                   ++theTSBMap.count.AsString++"origins to"
                   ++thefacil++":"++tscost.AsString+nl
        else
      therepstr = therepstr+"Transshipment cost from"
                  ++theorig++"to"++
                  thefacil++":"++tscost.AsString+nl
    end
    pickcostrate = 0
    for each k in theTSBMap
       if (theTSBMap.Get(k)) then
         theorig = theTSVTab.ReturnValue(theorigfld,k)
         theratio = theTSVTab.ReturnValue(theflowfld,k)
                                              /thetsflow
         for each m in theDCVTab
           thefacil2 = theDCVTab.ReturnValue
                                 (thedcfld,m)
           if (thefacil2 = theorig) then
              pickcostrate = pickcostrate +
                             theDCVTab.ReturnValue
                             (dcratefld,m)* theratio
           end
          end
        end
     end
     pickcost = theamt*pickcostrate
     if (theTSBMap.Count > 1) then
      therepstr = therepstr+"Weighted average pick cost at"
                  ++theTSBMap.count.AsString++"origins:"
                  ++pickcost.AsString+nl
     else
      therepstr = therepstr+"Pick cost at origin DC:"
                  ++pickcost.AsString+nl
       end
     else 'a pick cost
       pickcost = theamt * theDCVTab.ReturnValue
                                    (dcratefld,j)
    therepstr = therepstr+"Pick cost at "+thefacil++
                    ":"++pickcost.AsString+nl
           end
        end
      end
    end
    totalCost = d2scost + xdoccost + tscost + pickcost
    therepstr = therepstr+nl+
                "Total Logistics Cost for Rx Products:"++
                totalCost.AsString+nl
    MsgBox.Report(therepstr,
                 "Total Logisitic Costs to Serve Demand Region"++
                 store.AsString)
    av.Run("Tables.Unlink","")
    theBStMap.Clear(ii)
    theStoreVTab.UpdateSelection
    theBMap.ClearAll
    theD2SVTab.UpdateSelection
    theDCBMap = theDCVTab.GetSelection
    theDCBMap.ClearAll
    theDCVTab.UpdateSelection
    theTSBMap = theTSVTab.GetSelection
    theTSBMap = theTSVTab.GetSelection
    theTSBMap.ClearAll
    theTSVTab.UpdateSelection
  end ' on jj

  return Nil
```

```
'*****************************************************************

' Scriptname:    TraceRxAll

' Filename:      tracerxa.ave

' Description:   Script finds the total Rx logistics cost
'                for each demand region and writes it to a
'                new dBase file called RxLgCst.dbf.  This
'                file is then joined to the Demand Regions
'                table.

' Requires:      dirstore.dbf, tranship.dbf, Demand Regions
'                theme, CVS DCs theme exists

' Called by:     View menu click event
                 ("Trace Costs: Chain-wide Rx Only")

' Calls:          Tables.Link, Tables.Unlink

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "TraceRxAll"

' Warn user about time to complete this script

resume = MsgBox.YesNo("This trace takes approximately 10 minutes.
                      "+nl+"Do you want to continue?",
                      "Trace Costs: Chain-wide Rx Only", FALSE)
if (resume = false) then
  exit
end

' Set the number format for the script

Script.The.SetNumberFormat("d.dd")

'  Get the VTabs to be used and get the working directory
av.Run("Tables.Link","")
theDirectory = av.GetProject.GetWorkDir.AsString

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Demand Regions")
```

```
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Demand Regions theme does not exist.",
               Scriptname)
  exit
end
theStoreVTab = theTheme.GetFTab
theD2STable = av.GetProject.FindDoc("dirstore.dbf")
if(theD2STable = Nil) then
  MsgBox.Info("ERROR: dirstore.dbf table does not exist.",
              Scriptname)
  exit
end
theD2SVTab = theD2STable.GetVTab
theDCTheme = theView.FindTheme("CVS DCs")
if (theDCTheme = Nil) then
  MsgBox.Error("ERROR: CVS DCs theme does not exist.",
               Scriptname)
  exit
end
theDCVTab = theDCTheme.GetFTab
theTSTable = av.GetProject.FindDoc("tranship.dbf")
if (theTSTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist.",
               Scriptname)
  exit
end
theTSVTab = theTSTable.GetVTab

' Get the bitmap for the Demand Regions VTab

theBStMap = theStoreVTab.GetSelection

'check if table exists
sumrxcst_exists = (av.GetProject.FindDoc("RxLgCst.dbf")
                   = NIL).Not
skip = 0
if (sumrxcst_exists) then
  thedoc = av.GetProject.FindDoc("RxLgCst.dbf")
  if (MsgBox.YesNo("Overwrite existing logistics cost table?",
    "The Table RxLgCst already exists",false)) then
    if (sumrxcst_exists) then
      av.GetProject.RemoveDoc(thedoc)
    end
  else
    skip = 1
  end
end
'create a newtable
if (skip = 0) then
  flnm = theDirectory + "/RxLgCst.dbf"
  newVTab    = VTab.MakeNew( flnm.AsFileName, dBase )
  storefld   = Field.Make ("DemRegion",#FIELD_CHAR,16, 0)
  directfld = Field.Make ("RxDrctCst",#FIELD_DECIMAL,16,2)
  pickfld    = Field.Make ("RxPickCst",#FIELD_DECIMAL,16,2)
  transfld   = Field.Make ("RxTranCst",#FIELD_DECIMAL,16,2)
  xdocfld    = Field.Make ("RxXdocCst",#FIELD_DECIMAL,16,2)
```

```
totfld    = Field.Make ("RxTotlCst",#FIELD_DECIMAL,16,2)
newVTab.AddFields({storefld, directfld,pickfld,
                   transfld,xdocfld,totfld})
storefld.SetAlias("Demand Region")
directfld.SetAlias("Rx Direct Cost")
pickfld.SetAlias("Rx Pick Cost")
transfld.SetAlias("Rx Tranship Cost")
xdocfld.SetAlias("Rx Crossdock Cost")
totfld.SetAlias("Rx Total Cost")
theBStMap.ClearAll
thestorefld = theStoreVTab.FindField("Demand Region")
numrecs = theStoreVTab.GetNumRecords
for each ii in theStoreVTab
  'Clear the bitmap
  theBSTMap.ClearAll
  ' Set the record in the demand regions table
  theBStMap.Set(ii)
  theStoreVTab.UpdateSelection
  theamt = 0
  thetsflow = 0
  d2scost = 0
  pickcost = 0
  xdoccost = 0
  tscost = 0
  if (theBStMap.Get(ii)) then
      store = theStoreVTab.ReturnValue(thestorefld,ii)
  end
  ' Get the dirstore.dbf record selected by the link
  theBMap = theD2SVTab.GetSelection
  ' Reselect those record with Rx flow greater than zero
  expr = "([OptimizedValue] > 0)" ++ "and" ++
         "([Product] = ""Rx"")"
  theD2SVTab.Query(expr, theBMap, #VTAB_SELTYPE_AND)
  theratefld = theD2SVTab.FindField("ActualRate")
  theamtfld = theD2SVTab.FindField("OptimizedValue")
  thed2sdcfld = theD2SVTab.FindField("Facility")
  ' Loop through selected set in dirstore.dbf
  for each i in theBMap
    if (theBMap.Get(i)) then
      theD2SVTab.UpdateSelection
      thefacil = theD2SVTab.ReturnValue(thed2sdcfld,i)
      theamt = theD2SVTab.ReturnValue(theamtfld,i)
      d2scost = theD2SVTab.ReturnValue(theratefld,i) * theamt
    end
    'for this flow, see what is happening at the DC by
    'getting the selected set in the CVS DCs table
    theDCBMap = theDCVTab.GetSelection
    hasrx = theDCVTab.FindField("HasRx")
    dcratefld = theDCVTab.FindField("Rx Rate")
    dcfld = theDCVTab.FindField("Facility")
    for each j in theDCBMap
      if (theDCBMap.Get(j)) then
        thedc = theDCVTab.ReturnValue(dcfld,j)
        if (thedc <> thefacil) then
           continue
        end
```

```
 rxthere = theDCVTab.ReturnValue(hasrx,j)
 if (rxthere = 0) then
   xdoccost = theamt *
              theDCVTab.ReturnValue(dcratefld,j)
      'put traceback to tranship here
  theTSBMap = theTSVTab.GetSelection
  expr = "([OptimizedValue] > 0)" ++ "and" ++
         "([Product] = ""Rx"")"
   theTSVTab.Query(expr, theTSBMap, #VTAB_SELTYPE_AND)
   theTSVTab.UpdateSelection
   thetsratefld = theTSVTab.FindField("ActualRate")
   theorigfld = theTSVTab.FindField("OriginFacility")
   theflowfld = theTSVTab.FindField("OptimizedValue")
   thedcfld = theDCVTab.FindField("Facility")
   'get total flow into xdoc dc
   thetsflow = 0
   for each k in theTSBMap
     if (theTSBMap.Get(k)) then
       thetsflow = thetsflow +
                   theTSVTab.ReturnValue(theflowfld,k)
     end
   end
   'get average cost per unit
   therate = 0
   for each k in theTSBMap
    if (theTSBMap.Get(k)) then
       theratio = theTSVTab.ReturnValue(theflowfld,k)
                                      /thetsflow
      therate = therate +
                theTSVTab.ReturnValue(thetsratefld,k)
                * theratio
       theorig = theTSVTab.ReturnValue(theorigfld,k)
    end
   end
   tscost = theamt * therate
   pickcostrate = 0
   for each k in theTSBMap
     if (theTSBMap.Get(k)) then
      theorig = theTSVTab.ReturnValue(theorigfld,k)
      theratio = theTSVTab.ReturnValue(theflowfld,k)
                /thetsflow
         for each m in theDCVTab
             thefacil2 = theDCVTab.ReturnValue
                                      (thedcfld,m)
             if (thefacil2 = theorig) then
                 pickcostrate = pickcostrate +
                 theDCVTab.ReturnValue(dcratefld,m)
                 * theratio
               end
         end
    end
 end
   pickcost = theamt * pickcostrate
else 'a pick cost
  pickcost = theamt *
             theDCVTab.ReturnValue(dcratefld,j)
```

```
                end
             end
           end
         end
         theBStMap.Clear(ii)
         theStoreVTab.UpdateSelection
         newrec = newVtab.AddRecord
         newVTab.SetValue(storefld,newrec,store)
         newVTab.SetValue(directfld,newrec,d2scost)
         newVTab.SetValue(pickfld,newrec,pickcost)
         newVTab.SetValue(transfld,newrec,tscost)
         newVTab.SetValue(xdocfld,newrec,xdoccost)
         totcost = d2scost+pickcost+tscost+xdoccost
         newVTab.SetValue(totfld,newrec,totcost)
       end ' on ii
     end 'if skip = 0
     theBStMap.ClearAll
     theStoreVTab.UpdateSelection
     theBMap = theD2SVTab.GetSelection
     theBMap.ClearAll
     theD2SVTab.UpdateSelection
     theBMap = theDCVTab.GetSelection
     theBMap.ClearAll
     theDCVTab.UpdateSelection
     theBMap = theTSVTab.GetSelection
     theBMap.ClearAll
     theTSVTab.UpdateSelection
     av.Run("Tables.Unlink","")

     ' Set the new table to uneditable and write to file

     newVTab.SetEditable(FALSE)
     newVTab.Flush

     checkTable = av.GetProject.FindDoc("Rx Logistics Costs")
     if (checkTable <> Nil) then
       av.GetProject.RemoveDoc(checkTable)
     end

     ' Bring the new table into the project

     newTable = Table.Make(newVTab)
     newTable.SetName("Rx Logistics Costs")
     av.GetProject.AddDoc(newTable)

     ' Join the newTable to the Demand Regions table

     theStoreVTab.Join(thestorefld, newVTab, storefld)

     MsgBox.Info("Tracing of Rx logistics cost for"+nl+
                 "each demand region is complete.",
                 "Trace Costs: Chain-wide Rx Only")

     return Nil
```

```
'*****************************************************************

' Scriptname:    TransCWFlowTheme.Make

' Filename:      transcwf.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a Flow theme based on the CW
'                Flow field in Transshipment theme table.  Zero
'                value transshipment flows are made invisible
'                using the null value and symbol

' Requires:      Transshipment theme must exist

' Called by:     View menu item click event
'                ("Display Flows:Transshipments by CW Flow")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*****************************************************************
Scriptname = "TransCWFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Transshipments")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called Transshipments
               does not exist.", Scriptname)
  exit
end

catString = "CW Flow"

checkTheme = theView.FindTheme("Transshipments"++catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Transshipments"
                                  ++catString)
  if (theTable <> NIL) then
```

```
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the Transshipment theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend

' Select a color from the color palette to
' be used in drawing the transhipment lines

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with three natural
' breaks and size the lines according to
' the flow volume

newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
newLegend.SetNullValue(catString, 0)
newLegend.DisplayNoDataClass(FALSE)
newLegend.Natural(newTheme, catString, 3)
theSymbolList = newLegend.GetSymbols
thickness = 1
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
end
theSymbolList.UniformColor(theColor)

' Create a null symbol for the legend
' and set it

nullSym = Symbol.Make(#SYMBOL_PEN)
nullColor = Color.GetBlue
nullColor.SetTransparent(TRUE)
nullSym.SetColor(nullColor)
newLegend.SetNullSymbol(nullSym)
newTheme.SetLegend(newLegend)
newTheme.SetName ("Transshipments"++catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme(newTheme)
newTheme.UpdateLegend
theView.Invalidate
nullColor.SetTransparent(FALSE)

return Nil
```

```
'*****************************************************************

' Scriptname:    TransFlowValues.Calculate

' Filename:      transflo.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script copies the values of transhipment
'                flows for Rx and CW products from the
'                dirstore.dbf file to the Transhipments FTab
'                using an origin-destination-product string
'                concatenation.  These three new fields in the
'                FTab are then totalled and the total value is
'                added to the fourth new field in the FTab
'                called Total Flow.
'
' Requires:      tranship.dbf file and Transhipments
'                flow theme exist

' Called by:     TranshipLine.Build

' Calls:         Nil

' SELF:          the Transhipments theme

' Returns:       Nil

'*****************************************************************
Scriptname = "TransFlowValues.Calculate"

' Retrieve the theme argument
theTheme = SELF.Get(0)

' Find the tranship.dbf file and add the new field concatenating
' the facility name, the demand region name,
' and the product category

theTranTable = av.Getproject.FindDoc("tranship.dbf")
if (theTranTable = Nil) then
  MsgBox.Error("ERROR: tranship.dbf table does not exist."+NL+
          "Transhipment flow values not calculated.", Scriptname)
  exit
end
theVTab = theTranTable.GetVTab
theVTab.SetEditable(TRUE)
odpfld2 = Field.Make("ODP",#FIELD_CHAR,35,0)
theVTab.AddFields({odpfld2})
theval = "[OriginFacility]+[DestinationFacility]+[Product]"
theVTab.Calculate(theval,odpfld2)

' Get the Transhipment FTab and add the new flow fields for
' the two product categories, the total flow, and the
' origin-destination-product (ODP) field
```

```
theFTab = theTheme.GetFTab
theFTab.SetEditable(TRUE)
cwfld = Field.Make("CW Flow",#FIELD_DECIMAL, 16,2)
rxfld = Field.Make("Rx Flow",#FIELD_DECIMAL, 16, 2)
totfld = Field.Make("Total Flow",#FIELD_DECIMAL, 16, 2)
odpfld = Field.Make("ODP",#FIELD_CHAR,35, 0)
theFTab.AddFields({cwfld, rxfld, totfld, odpfld})

' Calculate the CW flow

newval2 = "[Origin]+[Destination]+""CW"""
theFTab.Calculate(newval2,odpfld)
theFTab.Join(odpfld,theVTab,odpfld2)
theflowval = "[OptimizedValue]"
theFTab.Calculate(theflowval,cwfld)
theFTab.UnjoinAll

' Calculate the Rx flow

newval2 = "[Origin]+[Destination]+""Rx"""
theFTab.Calculate(newval2,odpfld)
theFTab.Join(odpfld,theVTab,odpfld2)
theflowval = "[OptimizedValue]"
theFTab.Calculate(theflowval,rxfld)
theFTab.UnjoinAll

' For each record in FTab, set null values to zero

for each rec in theFTab
  cwvalue = theFTab.ReturnValue(cwfld, rec)
  if (cwvalue.IsNull) then
    theFTab.SetValue(cwfld, rec, 0)
  end
  rxvalue = theFTab.ReturnValue(rxfld, rec)
  if (rxvalue.IsNull) then
    theFTab.SetValue(rxfld, rec, 0)
  end
end

' Calcualte the total flow

newval2 = "[Rx Flow] + [CW Flow]"
theFTab.Calculate(newval2,totfld)

' Remove the ODP field from the Transhipments FTab
' since it is no longer needed

theFTab.RemoveFields({odpfld})
theFTab.SetEditable(FALSE)

' Remove the ODP field from the tranship.dbf file
theVTab.RemoveFields({odpfld2})
theVTab.SetEditable(FALSE)

return Nil
```

```
'*****************************************************************

' Scriptname:    TranshipLine.Build

' Filename:      tranship.ave

' Author:        Kenneth Bennett

' Date           May 3, 1998 (Updated)

' Description:   Script receives the FTab of the DCs theme from
'                FlowLine.Build and uses it o get X and Y
'                coordinates in order to build the transhipment
'                lines.  It then calls TransFlowValues.Calculate
'                script to join the flow values to the
'                Transhipments FTab.

' Requires:      Demand by Region view must exist

' Called by:     TransportationLines.Build

' Calls:         TransFlowValues.Calculate

' SELF:          the CVS DCs FTab

' Returns:       Nil

'*****************************************************************
Scriptname = "TranshipLine.Build"

dcFTab = SELF.Get(0)

'dcFTab = av.GetActiveDoc.GetActiveThemes.Get(0).GetFTab

if (dcFTab = nil) then
  MsgBox.Info("Error - Table not found","")
  exit
end

' Get the facility name from the dcFTab

facfld = dcFTab.FindField("Facility")

' Give a name and path to the new transhipment FTab and create it

defName = FileName.Make(av.GetProject.GetWorkDir.AsString)
          .MakeTmp("trnshp", "dbf")
theFName = FileDialog.Put(defName, "*.dbf", "Save FTab As")
if (nil <> theFName) then
  transFTab = FTab.MakeNew( theFName, POLYLINE )
else
  transFTab = FTab.MakeNew( defName, POLYLINE )
end

'Add fields to the new transFTab
```

```
orig = Field.Make("Origin", #FIELD_CHAR, 20, 0)
dest = Field.Make("Destination", #FIELD_CHAR, 20, 0)
od = Field.Make("O-D", #FIELD_CHAR, 40, 0)
transFTab.AddFields({orig, dest, od})

' Now create a variable for the shape field in the transFTAB

shapeF = transFTab.FindField( "Shape" )

' Loop through the dcFTab and generate a line
' for each DC pair combination except when a
' DC is paired with itself

av.ShowStopButton
av.ShowMsg("Building Transhipment Lines...")
numDC = dcFTab.GetNumRecords
n = 0

for each o in dcFTab
  origstring = dcFTab.ReturnValue(facfld, o)
  origpnt = dcFTab.GetLabelPoint(o)
  X1 = origpnt.GetX
  Y1 = origpnt.GetY
  for each d in dcFTab
    deststring = dcFTab.ReturnValue(facfld, d)
    if (origstring <> deststring) then
      destpnt = dcFTab.GetLabelPoint(d)
      X2 = destpnt.GetX
      Y2 = destpnt.GetY
      newstring = origstring+deststring
      newRec = transFTab.AddRecord
      l = Line.Make(X1@Y1,X2@Y2).AsPolyLine
      transFTab.SetValue(shapeF, newRec, l)
      transFTab.SetValue(orig, newRec, origstring)
      transFTab.SetValue(dest, newRec, deststring)
      transFTab.SetValue(od, newRec, newstring)
    end 'if
  end 'internal for loop
  n = n + 1
  progress = (n / numDC) * 100
  doMore = av.SetStatus( progress )
  if (not doMore) then
       break
  end 'if
end ' external for loop

' make transFTab into a theme and add to the view

theTheme = FTheme.Make( transFTab )
theSymList = theTheme.GetLegend.GetSymbols
theColorPaletteList = av.GetSymbolWin.GetPalette
                      .GetList(#PALETTE_LIST_COLOR)
'Get the color green for the transhipment lines
theColor = theColorPaletteList.Get(15)
theSymbol = theSymList.Get(0)
theSymbol.SetColor(theColor)
```

```
theTheme.SetName("Transshipments")

'  Add the theme to the view

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist."+NL+
               "Transhipments theme not added to it.",
               Scriptname)
  return Nil
elseif (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view."+NL+
               "Transhipments theme not added to it.",
               Scriptname)
  return Nil
else
  theView.AddTheme( theTheme )
  theView.Invalidate
end

' Run the script to calculate the
' various transhipment flow values

av.Run("TransFlowValues.Calculate", {theTheme})

return Nil
```

```
'******************************************************************

' Scriptname:    TransportationLines.Build

' Filename:      transpor.ave

' Description:   This script is the master script for generating
'                a DCs theme and the DC-to-Region and Transhipment
'                transportation themes. The script first calls the
'                FlowLines.Build script. Then the FlowLine.Build
'                script calls the SpliceLatLon script, which in
'                turn calls the AddXY script.  Together, these
'                scripts generate the DCs theme.  FlowLine.Build
'                then generates all the DC-to-Region Flow
'                transporation lines FlowLine.Build then returns
'                the DC theme FTab to this script, and this script
'                calls the TranshipLine.Build script which then
'                builds the Transhipments theme.  At the end, the
'                CVS DCs theme is shuffled to the top of the TOC.

' Called by:     Menu click event ("Build Transport Lines")

' Calls:         FlowLine.Build, TranshipLine.Build

' SELF:          Nil

' Returns:       Nil

'******************************************************************

theFTab = av.Run("FlowLine.Build", Nil)

av.Run("TranshipLine.Build", {theFTab})

theView = av.GetActiveDoc
theThemes = theView.GetThemes
dcTheme = theView.FindTheme("CVS DCs")
theThemes.Shuffle(dcTheme, 0)
dcTheme.SetActive(TRUE)
theView.InvalidateTOC(Nil)

return Nil
```

```
'*******************************************************************

' Scriptname:    TransRxFlowTheme.Make

' Filename:      transrxf.ave

' Author:        Kenneth Bennett

' Date:          May 3, 1998

' Description:   Script generates a Flow theme based on the
'                Rx Flow field in Transshipment theme table.
'                Zero value transshipment flows are made
'                invisible using the null value and symbol.

' Requires:      Transshipment theme must exist

' Called by:     View menu item click event
'                ("Display Flows: Transshipments by Rx Flow")

' Calls:         Nil

' SELF:          Nil

' Returns:       Nil

'*******************************************************************
Scriptname = "TransRxFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Transshipments")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called Transshipments
               does not exist.", Scriptname)
  exit
end

catString = "Rx Flow"

checkTheme = theView.FindTheme("Transshipments"++catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Transshipments"
                                   ++catString)
  if (theTable <> NIL) then
```

```
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the Transshipment theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend

' Select a color from the color palette to
' be used in drawing the transhipment lines

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with three natural
' breaks and size the lines according to
' the flow volume

newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
newLegend.SetNullValue(catString, 0)
newLegend.DisplayNoDataClass(FALSE)
newLegend.Natural(newTheme, catString, 3)
theSymbolList = newLegend.GetSymbols
thickness = 1
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
end
theSymbolList.UniformColor(theColor)

' Create a null symbol for the legend
' and set it

nullSym = Symbol.Make(#SYMBOL_PEN)
nullColor = Color.GetBlue
nullColor.SetTransparent(TRUE)
nullSym.SetColor(nullColor)
newLegend.SetNullSymbol(nullSym)
newTheme.SetLegend(newLegend)
newTheme.SetName ("Transshipments"++catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme(newTheme)
newTheme.UpdateLegend
theView.Invalidate
nullColor.SetTransparent(FALSE)

return Nil
```

```
'*****************************************************************

' Scriptname:     TransTotalFlowTheme.Make

' Filename:       transtot.ave

' Author:         Kenneth Bennett

' Date:           May 3, 1998

' Description:    Script generates a Flow theme based on the
'                 Total Flow field in Transshipment theme table.
'                 Zero value transshipment flows are made
'                 invisible using null value and symbol.

' Requires:       Transshipment theme must exist

' Called by:      View menu item click event
'                 ("Display Flows: Transshipments by Total Flow")

' Calls:          Nil

' SELF:           Nil

' Returns:        Nil

'*****************************************************************
Scriptname = "TransTotalFlowTheme.Make"

theView = av.GetProject.FindDoc("Demand by Region")
if (theView = Nil) then
  MsgBox.Error("ERROR: Demand by Region view does not exist.",
               Scriptname)
  exit
end
if (not (theView.Is(View))) then
  MsgBox.Error("ERROR: Demand by Region doc is not a view.",
               Scriptname)
  exit
end

theTheme = theView.FindTheme("Transshipments")
if (theTheme = Nil) then
  MsgBox.Error("ERROR: Theme called Transshipments
                does not exist.", Scriptname)
  exit
end

catString = "Total Flow"

checkTheme = theView.FindTheme("Transshipments"++catString)
if (checkTheme <> nil) then
  theView.DeleteTheme(checkTheme)
  theTable = av.GetProject.FindDoc("Attributes of Transshipments"
                                   ++catString)
  if (theTable <> NIL) then
```

```
    av.GetProject.RemoveDoc(theTable)
  end
end

' Clone the Transshipment theme

newTheme = theTheme.Clone
newLegend = newTheme.GetLegend

' Select a color from the color palette to
' be used in drawing the transhipment lines

theColor = av.Run("ColorPalette.SelectColor", Nil)

' Classify the legend with three natural
' breaks and size the lines according to
' the flow volume

newLegend.SetLegendType(#LEGEND_TYPE_SYMBOL)
newLegend.SetNullValue(catString, 0)
newLegend.DisplayNoDataClass(FALSE)
newLegend.Natural(newTheme, catString, 3)
theSymbolList = newLegend.GetSymbols
thickness = 1
for each s in theSymbolList
  s.SetSize(thickness)
  thickness = thickness + 1
end
theSymbolList.UniformColor(theColor)

' Create a null symbol for the legend
' and set it

nullSym = Symbol.Make(#SYMBOL_PEN)
nullColor = Color.GetBlue
nullColor.SetTransparent(TRUE)
nullSym.SetColor(nullColor)
newLegend.SetNullSymbol(nullSym)
newTheme.SetLegend(newLegend)
newTheme.SetName ("Transshipments"++catString)
newTheme.SetActive(FALSE)
newTheme.SetVisible(TRUE)
theView.AddTheme(newTheme)
newTheme.UpdateLegend
theView.Invalidate
nullColor.SetTransparent(FALSE)

return Nil
```

Vita

Kenneth Bennett was born in Chula Vista, a suburb of San Diego, California, on February 19, 1966. He attended Hilltop High School in Chula Vista where he graduated in 1984. He received his Bachelor of Arts degree from the University of California, Los Angeles, in 1989, with a major in English and World Literature. After graduating from UCLA, he spent one year traveling in Mexico and the Southwest of the United States. From 1990 until 1993, he worked for an airfreight forwarding company, and from 1994 to 1996 as a trade journalist in such fields as import and export trade, and the private sector financing of international energy, transportation, and communications infrastructure. These experiences led him to an interest in the spatial distribution of economic activity, particularly in the areas of energy and transportation. In 1996, he entered the Department of Geography at the University of Tennessee in pursuit of a Master of Science degree. Upon successful defense of this thesis, Kenneth Bennett will receive his M.S. in Geography with an emphasis in the use of geographic information systems for transportation and logistics analysis.