HIGH-PERFORMANCE COMPUTING AND STATISTICAL ANALYSIS

Chaired by Mark L. Davison, *University of Minnesota*

Application of high-performance computing to the management of social science and demographic data

ALBERT F. ANDERSON

*Population Studies Center, University of Michigan, Ann Arbor, Michigan
and Public Data Queries, Inc., Chelsea, Michigan*

The costs of conducting censuses and surveys have always been high. Nonetheless, demographic, social, environmental, and health scientists have found it possible to gather quantities of data that far exceed the capabilities of any contemporary computing technology to extract fully the information contained in the data. Recent developments in computing and information science technology offer the promise of supporting innovative approaches to managing and analyzing data that can dramatically increase the ease of access to the information hidden within these data. This presentation describes a highly integrated system of data, metadata, instructional tutorials, hardware, and software designed to minimize the technical barriers to working with large and complex data sets. The objective is to provide easy, low-cost, fast, and meaningful access to these data for a broad spectrum of users.

Acquiring and managing data, and extracting information from those data, are uniquely human activities. Information is important at every level of human inquiry today, from personal to corporate, local to international. As a consequence, significant commercial as well as academic effort has been directed toward developing computer-based systems that facilitate the organization, storage, retrieval, analysis, and interpretation of information. The project described in this paper is developing an interactive information-retrieval system, PDQ-Explore, optimized to provide fast, easy, low-cost, and meaningful access to data sets ranging up to the size of full national censuses.

Extracting information from data is often an iterative process throughout which the parameters for the next cycle are determined from the results of the previous cycle. The system presented here has the potential to dramatically impact this process by minimizing the costs and time associated with each iteration. It is an especially valuable tool when used by investigators who have the substantive knowledge needed to make effective use of a large data

set, but not the skills, resources, or technical understanding of the data set usually required to use the data.

The system consists of four general components: data, metadata, software, and hardware. The system runs on parallel high-performance computing systems. It makes very efficient use of high-performance processors, especially of the random-access memory (RAM) and the primary instruction caches associated with those processors to minimize execution times. Test implementations of the system are currently running on a four-processor symmetric multiple-processor (SMP) system and a single-processor PC. A preproduction implementation of the system will be undertaken in early 1997. The system will be accessible via local and wide-area networks to allow a large base of users to share the cost of a relatively expensive system.

The project reflects the author's 30 years of experience in assisting colleagues and students with the extraction of information from a wide variety of large census and survey data sets and his son's expertise in computer engineering (Anderson & Anderson, 1994; Anderson, Hanss, & Hermalin, 1991). Recent work on the project has been supported in part by Small Business Technology Innovation Research and Small Business Technology Transfer Research grants to a family-owned company, Public Data Queries, Inc. Following a discussion of the problem being addressed by the project and background on efforts directed toward solving the problem, this paper focuses on the technical aspects of the solution we are developing.

The work on the project described in this paper has been funded in part by Small Business Technology Transfer Research and Small Business Innovation Research grants from the National Institute for Child Health and Human Development—1R41HD32222-01 and 1R43HD33633-01, respectively. For further information about the availability of PDQ-Explore, please contact the author at Population Studies Center, 1225 South University, Ann Arbor, MI 48104-2590 (e-mail: afa@umich.edu or afa@pdq.com).

The Problem

Although the costs of gathering data are high, social, behavioral, and health scientists have often found it possible to gather more data than can be readily managed and analyzed by using existing technology. As a consequence, many important data sets are underutilized relative to the value of the information they contain. For the first time, technology exists that, when creatively applied to data management and analysis, can dramatically increase the ease of access to data sets of almost any size and decrease the costs associated with using those data—costs measured in terms of the time, dollars, effort, and technical and substantive expertise needed to make effective use of the data. Tasks that once required weeks, sometimes months, or that often would not be attempted at all, can now be accomplished in a few seconds.

One specific data set, the 5% Public Use Microdata Sample (PUMS) from the 1990 U.S. Census (Bureau of the Census, 1992), will be used to provide a context for presenting the problem and the solution being developed. Following each of the last four censuses, the Bureau of the Census has released to the public one or more data sets containing housing- and person-level information from the long form of the census questionnaire. In 1990, 1% and 5% samples from the full population and a 3% sample from all housing units having one or more persons aged 60 or older were made available. The 5% data set was released on 26 reels of 9-track magnetic tape—4.5 GB of ASCII data representing more than 18 million housing and person records. Had a data set of this size been available in the days of punched cards, the data set would have filled 27,000 boxes of cards. Today, the data can be stored on a single 4- or 8-mm tape cartridge or a single hard disk drive.

The PUMS is one of many data sets of sufficient size and/or complexity to present major data-management problems to those who would use the data. The difficulty of working with these data sets has, in effect, limited their use in the past to persons with access to research facilities well endowed with computing hardware and data-management expertise. A number of federal agencies gather data of this nature that are now available to the public. These include more than 25 years of mortality-detail files from the National Center for Health Statistics (NCHS), Current Population Surveys (CPS) from the Bureau of Labor Statistics and the Census, and the Consumer Expenditure Surveys from Labor Statistics. The Social History Research Laboratory at the University of Minnesota is compiling the Integrated Public Use Microdata Series (IPUMS) with sample data sets from 11 censuses and a total of approximately 65 million records compiled to date. Smaller federally funded longitudinal studies are also becoming available to the public. Some of these accumulate masses of data over time, but the complexity of the data sets is often a more significant barrier to their use. The Survey of Income and Program Participation (SIPP) from the Bureau of the Census and

the National Election Studies (NES), Panel Study of Income Dynamics (PSID), and Health and Retirement Survey (HRS) conducted by the Survey Research Center at the University of Michigan are examples of such data sets.

In 1990, 15.9% of all housing units in the United States received the long form of the census questionnaire that requested information on a broad range of demographic, social, and economic characteristics of the housing unit and its inhabitants. Thus, the 5% sample includes data from slightly less than one third of the long-form records. The 1990 samples are stratified with housing and person weights included on the records to allow population estimates to be calculated. Earlier PUMS were self-weighting—multiplying counts based on a 1% file by 100 would yield population estimates, for example.

Figure 1 displays portions of a few records from the 1990 5% PUMS. The data set is structured as a hierarchy of records. The first column of each record designates with an “H” or a “P” whether the record is a housing record or a person record. Other than the record-type designator, all data in the records are numeric with a leading minus sign (–) used to denote negative values in the case of some income items. The records are in a fixed format with specified columns containing responses to specific census items or recodes of items as provided by the Bureau of the Census. Technical documentation for the data set includes the “codebook” that presents the correspondence between items and the columns in the data record representing the item and also the correspondence between the response categories for an item and the numeric codes assigned to those responses (see Figure 2). For example, “age” is coded in columns 15–16 of the person record, with values ranging from “00” through “90.” As the documentation notes, “00” indicates an age of less than one year, “01”–“89” the age in years, and “90” an age of 90 or more.

The PUMS includes a broad range of housing and individual information. Thus, it is an especially rich source of information about the nation at a cross-section in time. The PUMS series provides a basis for identifying trends over time. To protect individual confidentiality, only limited geographical detail is available in the PUMS files, with identification to levels that include a minimum of approximately 100,000 persons. Nonetheless, the files contain much information about the U.S. population, information that is of potential relevance to a broad range of issues of local, regional, and national concern. Since the 1900 census, research monographs have been commissioned, initially by the Bureau of the Census and more recently by public foundations, to report significant research findings based on analyses of the then current census data. For the 1990 census, the two-volume work sponsored by the Russell Sage Foundation, *State of the Union: America in the 1990s*, reports on economic and social trends as identified by analyses of the 1990 census data (Farley, 1995). The topics covered in these volumes reflect in part the range of information gathered

```

H000000113260010080999900041003403000002006222150000000641115622621004800002...
P000000100000157000300000000221000000000501055000110050022100000001723340000...
P00000010110015600031000000024104000000050103000010905008910000002000040000...
P00000010210011940026000000010604000000050102600021105008910000002000040000...
H000000213260010080999900093002701000002006222130000000541112111621003000800...
P00000020010019010030000000010304000000016902600010505199910000002000040000...
H000000313260010080999900054003401000003004322001320000331111311521009000002...
P00000030010018820031000000021404000000043502300011005099910000002000040000...
H0000004132600100809999000700271000002009122110000000261113111721009001500...
P00000040000013700011000000010100000000163026000110088999226003002000040000...
P00000040110013400012000000010103000000163026000110088999226003002000040000...
P00000040200011440007000011310010100000163026000205088999226003002000000000...
P00000040210011140012000011310014100000163026000204088999226003002000000000...
P000000403100104400070000113100141000001630260001019999990000000000000000...
P00000040210010540008000011310014100000163026000302999999226003002000000000...
P00000040200011040008000011310010100000163026000204999999226003002000000000...
P000000408100121400110000000106040000000102026000110931999226001002000040000...
P000000408100119400080000000106040000000076026000209026999226001002000040000...
P000000408100169400110000000226040000000110017000101051999226001002000040000...
H000000513260010080999900040003001000002005122020000000431112411821003360300...
P000000500100161200270000000224040000000348018000110089022100000002000020800...
...
...
...

```

Figure 1. Example records from the 1990 5% PUMS.

in the census—for example, structure of the family, distribution of wealth, characteristics of the older population, the new immigrants, educational attainment, and economic roles of women and men.

Analyzing a data set, such as the PUMS, to produce the meaningful results, as in the monographs, requires that a sequence of operations be performed. These typically include: (1) reading (inputting) the data set record by record; (2) transforming/recoding the items to be used as filters for selecting cases for analysis; (3) filtering cases by selecting/excluding records to generate the subset of interest; (4) transforming/recoding items to be analyzed; and (5) analyzing the items of interest.

The first and last of these operations, reading the data and executing the analysis, are always performed. Item transformations and filtering of the records may be required for specific tasks. A complicated task may require that information from multiple records within a household be used to calculate such indexes as a socioeconomic status index or the number of bedrooms required for the sex–age composition of a given household as required by a given housing standard. For example, a housing code may allow a 3-year-old boy to share a bedroom with a 2-year-old sister, but not allow a 14-year-old boy to share a bedroom with a 13-year-old sister or a 7-year-old brother.

The difficulty of managing the full data set often leads to a first step of extracting from the data set only those cases and items needed for the desired analysis. The extract is then analyzed at a second step. The extract may still be a relatively large subset of the file if, for example, one is interested in analyzing several characteristics of the labor force using data from the 5% PUMS. Until just a few years ago, a sizable budget, access to a mainframe computing environment, a team of assistants with data management expertise and computing skills, and several months of time were needed to accomplish even modest

research aims using the PUMS or similar data sets. Although the revolution in computing and information technology has reduced these demands, working with these data still presents a major challenge to most users and requires hours of processing time.

The primary objective of the current project is to reduce to a minimum the barriers to using these data. While

Data	RELAT1	Size	2	Begin	9
	Relationship				
	00	Householder			
	01	Husband/wife			
	02	Son/daughter			
	03	Stepson/stepdaughter			
	04	Brother/sister			
	05	Father/mother			
	06	Grandchild			
	07	Other relative			
		Not related			
	08	Roomer/boarder/foster child			
	09	Housemate/roommate			
	10	Unmarried partner			
	11	Other nonrelative			
		Group quarters			
	12	Institutionalized person			
	13	Other persons in group quarters			
D SEX	Sex	1	11		
	0	Male			
	1	Female			
D RACE	Race	3	12		
	001-037	(See appendix C)			
	301-327	American Indian Tribes			
D AGE	Age	2	15		
	00	Less than 1 year			
	01-.89	Age in years			
	90	90 or more years old			
D MARITAL	Marital status	1	17		
	0	Now married, except separated			
	1	Widowed			
	2	Divorced			
	3	Separated			
	4	Never married or under 15 years old			

Figure 2. Example codebook entries from the 1990 5% PUMS.

the focus in this paper is specifically on aspects of the project related to overcoming the technical barriers, the project also addresses such other barriers as the complexity of the data and substantive concerns in using and understanding the data and interpreting results.

Historically, the primary technical problem associated with using social science data has been input/output (I/O). Tasks that at one time used punched cards, then magnetic tape, and eventually multiple-reel tape data files were usually I/O bound, as opposed to compute bound; that is, the time required to read the data usually overwhelmed the processing time required for the analysis. Time-shared mainframe systems allowed effective use to be made of processing power by handling a number of I/O-bound tasks "simultaneously." The recent development of high-capacity tape-and-disk storage devices has significantly eased the problems associated with the physical handling of the data, but processing times still range to hours for even simple analyses based on a full data set the size of the PUMS.

With the advent of low-cost, relatively high-speed disk storage, high-performance processors, large quantities of low-cost RAM, and high-performance linking networks, a hardware system can be designed to deliver levels of performance well beyond existing expectations for the execution of any given task with any given data set. However, expectations quickly adjust to new levels of performance. Colleagues who once waited months for results are now aware of the difference between 5- and 10-sec turnaround times. When this project began 4 years ago, the goal was to generate tabulations and summary statistics from data sets as large as the 1990 1% PUMS (3.5 million records) in 5 sec at the server level. The target is now for processing times of 1 sec for data sets as large as full national censuses—250 million records and more. From the user perspective, this translates to response times on the order of 5 sec under favorable network conditions. Designing a cost-effective system that can meet these targets requires that a balance in performance be maintained across the hardware elements of the system: disks, controllers, RAM, memory bandwidth, processors, and networks. Unfortunately, the optimal balance is task dependent. Tasks can be readily defined that turn any one of the hardware elements into the bottleneck. Consequently, the hardware system that best meets our needs will likely be the one that best meets the demands of the most common tasks.

A simple query might request a two-way tabulation between items from the PUMS. For example, one might be interested in the number of older children living at home. A cross-tabulation of two items, the relationship to the household head and the age of the person—with age grouped into 5-year intervals—is easy to specify and simple to execute. Relationship to head has 14 categories ranging from 0 to 13 and age ranges from 0 to 90, becoming 18 five-year intervals plus another for those 90 and over. Thus, a 14 by 19 table is defined. If we want to generate this table from the 5% PUMS, we need to access each of approximately 12.5 million values of rela-

tionship to head and the associated age of the person. The task is basically simple except for accessing the 12.5 million pairs of values. Depending upon how one approaches the problem, this can easily require several hours in most contemporary computing environments. We want to do it in 1 sec or less.

Another rather simple query, but one that can load the system in a different way, could be a request for the percentiles for wage and salary income for the labor force from the 1% PUMS. This requires that, in effect, a multimillion-record file be sorted. We handle this quickly by tabulating income into a 200,000 element vector—the range of values for wage and salary income. A system based on parallel processors can handle the tabulation very efficiently, but the partial tabulations that exist in the RAM associated with each of the individual processors in a parallel system must be moved to a common processor for aggregation. For a 32-processor system and assuming 4-byte elements, each tabulation vector would be 800,000 bytes long. Thus, approximately 25 MB of data—32 times 800,000—must be moved so that one processor can aggregate the results, identify the percentiles, and pass the results back to the user. If the percentiles for each of four racial categories are desired, tabulating into RAM is still feasible, but now 100 MB of data must be moved. One might also be interested in the accumulated aggregate income at each percentile to provide the information needed to plot Lorenz curves, tripling the intermediate storage and results that must be brought together and summed—assuming double-precision arithmetic is required to aggregate income (Kakwani, 1980). The capabilities of the communication links between the individual processors and the aggregating processor become critical at some point as the quantity of data that must be moved over the communication link increases.

A complex query might request that a socioeconomic-status (SES) item be created from items of information on the housing record (e.g., value of housing or rent) and the household head and spouse records (e.g., occupation, education, and income). Look-up tables might be used to transform occupation to a prestige score. Value of housing or rent might be adjusted to contribute equally to the SES score. The higher of the head or spouse scores on each of prestige, education, and income might be used in the determination. The final SES scores would be held as a housing-level characteristic and then perhaps included in the calculation of a variance/covariance matrix for 10 items at the individual level. This would likely become a compute-intensive task unlike the prior two examples that are computationally trivial.

The PDQ-Explore system was designed to assist demographers and other social scientists in their efforts to become familiar with data sets such as the PUMS. The difficulty of using the data sets tends to discourage their use by anyone other than well-funded researchers. The intent behind the development of PDQ-Explore has been to encourage greater use of the data by significantly reducing the investment in time, dollars, and effort required to access the data sets and to develop the familiarity with

the data needed to make informed use of the data sets. PDQ-Explore is a system with the potential to fulfill this intent. Before presenting the design of the PDQ-Explore system, we will review some of the efforts that have been made by others to facilitate the handling of large data sets.

Background

Large data sets are not unique to the social, behavioral, and health sciences. Commercial database products such as Oracle, Ingres, Sybase, Informix, FoxPro, and Paradox are available for general-purpose data management. Because of the needs of the corporate world, most of these products are oriented toward the handling of transactions—adding an entry to a database or retrieving an entry or subset of entries from a database. Organization of the data using pretabulations and indexes to facilitate access can allow some forms of information to be aggregated very quickly; other information may be essentially inaccessible.

While very efficient methods have been developed for handling transaction-oriented activity, significantly less attention has been given to developing efficient methods of access to information in data sets when every record in a data set, or significant subsets of the data that have not been pretabulated, must be handled to extract the desired information. This is generally true for analytic operations that generate summary statistics over the data or partitions of the data—means, medians, percentiles, and error estimates for income, for example, over a data set partitioned by race, age, and sex. The researchers who work with census and survey data often use data in exactly that way, applying selection criteria and filters to each record in a data set to create an extract of manageable size, which is then handled on a record-by-record basis by a statistical package such as SAS or SPSS. These packages and similar products provide for varying degrees of data management integrated with their extensive analytic capabilities.

Few people have developed the range of skills and expertise, along with the patience, to handle the full research process from beginning to end for large data sets such as the PUMS, even with the availability of the sophisticated statistical packages. More typically, a project is divided into a number of tasks which are shared among individuals, each with the expertise appropriate to a given task.

Data mining and data warehousing have recently attracted attention as interest in extending the analytic capabilities of database systems has increased. See, for example, the articles on data warehousing (Raden, 1996a, 1996b). Vendors that have developed excellent solutions for handling transactional data are having difficulty in adapting traditional database engines to handle analytic tasks such as locating the percentiles in the distribution of income for employed persons in the 5% PUMS or calculating mean income by race, sex, and education for selected clusters of occupations. By optimizing for queries of this nature, procedures that require hours when using typical database systems can be designed that are capa-

ble of generating responses in a few seconds. These procedures are not yet being used in commercially available products, however.

Data-mining systems automate the process of searching for patterns in data and subsets of data. These systems provide the most useful results when they identify middle-range patterns and clusters in the data—the most general patterns are probably already known and the most specific are too numerous and idiosyncratic to be of value, at least in commercial settings. Users need to know their data well to interpret and understand the midrange information, however. But, if they do know their data well, one can reasonably argue that a hybrid system that allows the user to interact with the system as they search for meaningful information may be more efficient than a fully automated one. This hybrid approach is more amenable to the traditional methods employed in scientific research. When based on an interactive system with very short response times to queries, the search for information is of an entirely different character from that experienced when more conventional tools are used.

Ian Taylor (1992) and his company in London, CHISO, Ltd., adapted a parallel system to generate tables from data sets such as the PUMS. His software, Tablemaker, ran on massively parallel single instruction, multiple data (SIMD) machines, 4,096- and 16,384-processor MasPar systems. It was designed to be a single-user system capable of serving the needs of research and commercial institutions requiring access to a single large proprietary data set.

Stonebraker (Stonebraker & Dozier, 1991) recognized the difficulty of handling every record in massive data sets in his work on the Sequoia 2000 project. His Postgres system, which evolved from the Sequoia project into a commercial product, Illustra, is described as an object-relational database-management system, ORDBMS (Wiegner, 1994). It focuses on allowing complex queries to be made of complex data structures that include audio, video, and satellite data. Like Tablemaker, Illustra appears to be oriented toward single users with a specific data set to be handled, but with sufficient flexibility to allow any data set to be adapted for use with the system.

Proprietary systems have been developed, but little publicized, for managing commercial databases in a manner similar to that of PDQ-Explore. Teradata and HP have developed a system used by Wal-Mart for inventory management in response to daily sales that has been described as functionally similar to PDQ-Explore (DePompa, 1996). Tucker has described the ShopKo system that adapts and integrates commercial packages into an on-line transaction processing system (Tucker, 1996). Companies that manage credit card transactions, geological databases in the oil exploration industry, health insurance and medical records, and other insurance data also use similar systems to generate summary reports over the full data sets or substantial subsets of the data, but at costs amounting to tens of millions of dollars for the very fast systems or response times in hours, rather

than seconds and fractions of seconds, for the more modest systems.

The Consortium for International Earth Science Information Network (CIESIN) has undertaken a project with funding from the National Aeronautics and Space Administration (NASA) and other sources to develop an information system similar to PDQ-Explore in function, but tailored to meet the needs of their users in the global environmental change research and policy-making communities. The system, Ulysses, is a redesign of the prototype Explore system that was implemented on eight loosely coupled HP-735 workstations at CIESIN in 1993 (Meij, 1996). Ulysses currently runs on two HP-735 workstations and provides access to the 1970, 1980, and 1990 1% PUMS data and the 1982 People's Republic of China 1% census data.

The Conversational Survey Analysis system developed and supported by the Computer-Assisted Survey Methods Program at the University of California, Berkeley, provides interactive access to survey data sets of modest size and allows a variety of statistical and analytic procedures to be applied to the data (CSM, 1993; Shanks, 1989). It has similarities to the earlier Michigan Interactive Data Analysis System (MIDAS), but with a stronger emphasis on integrating technical documentation and codebooks with data access (Fox & Guire, 1976).

The Great American History Machine (GAHM), developed originally by David Miller, Professor of History at Carnegie Mellon University, and now under further development and distribution by Chad McDaniel and the Software Development Group at the University of Maryland, is an example of a well-designed single-user interactive data system (McDaniel, 1994; Miller, 1988). GAHM allows a user to work very easily with 150 years of U.S. county-level data, selecting items for thematic display on maps of the United States. Flexible data recoding and transformation features are incorporated. Although statistical summaries of the selected items are available, most of the analytic power of the system rests in the ability of the user to discern visual patterns in the thematic display. The database is of modest size, 100 MB or so, consisting largely of census and election data since 1840. Even though the units for analysis are the relatively small number of counties in existence at any given time (3,143 counties and county equivalents in 1990), desktop computers have only recently become powerful enough to give reasonable response times—a few seconds to a minute or so. The most impressive feature of GAHM is the simplicity of the user interface. Anyone with an interest in seeing a thematic display, at the county level, of the value of farmland over time, mean incomes in 1980, or voting patterns in the 1940 presidential election can readily do so.

Developers of spreadsheets, graphic information system packages, and statistical packages are increasingly taking advantage of the power of desktop machines to allow users of their products to handle modest amounts of data interactively. Graphical user interfaces (GUIs) in conjunction with the processing power to handle tens of

thousands of records interactively are enticing some users away from the batch-oriented behavior that has characterized data analysis since the introduction of computers. Even so, considerable technical and substantive expertise is needed to use these systems. The primary strength of these systems is the overall power they offer the user to manage and analyze data, rather than simplicity and speed.

PDQ-Explore is not intended to fully duplicate the features of such existing analytic tools as SAS and SPSS, but rather to complement those tools in areas where PDQ-Explore can make a dramatic difference in how easily and how quickly tasks can be executed. It is intended to do simple things, such as tabulations, very simply and very quickly. Where more difficult things, such as recoding and data transformation, can also be executed simply and quickly, those capabilities are being incorporated into PDQ-Explore. While PDQ-Explore has been viewed as being especially valuable at the early, exploratory stages of becoming familiar with a data set and in situations where information is needed quickly, it has the potential to significantly reduce much of the effort involved in getting to the point where the power of a standard statistical package can be easily applied to an extract of the data. The system will allow researchers to experiment with the creation of ad hoc indices, recodes, and transformations much more easily than if a standard package was used from the beginning of the project. Upon determining the appropriate data operations to be performed, PDQ-Explore can be instructed to generate an extract consisting of exactly the data needed for the analysis. In the preproduction version of PDQ-Explore, scheduled for implementation in 1997, requests for extracts will be passed to a separate extraction system which will generate the extracts and move them to temporary file space from which users will move the extracts to their own facilities for subsequent analyses. Given sufficient computing resources for PDQ-Explore, later versions will integrate the extraction process within PDQ-Explore and assign free processors to execute subsequent analyses on the extracts. Only the output from those runs will need to be moved over the Internet to the user. Meanwhile, online users would continue to be served via the processors dedicated to the basic PDQ-Explore functions.

PDQ-Explore

PDQ-Explore is currently a work in progress. It is a highly optimized social science information-retrieval system. The high level of performance achieved by the system has been attained by combining very efficient program code with powerful processors linked to large quantities of RAM. The system is designed to allow users to work interactively with a variety of large census and survey data sets. Users are expected to include persons with relatively limited substantive and analytic expertise as well as the most sophisticated researchers. In its fully developed form, the system will allow users to extract information quickly and easily in the form of tabulations, summary statistics, simple multivariate analyses, and extracts from data sets containing millions of records. Re-

sults will be returned to users in a variety of forms, including textual, graphical, and geographical displays. The user interface will also incorporate interactive access to metadata, documentation, and multimedia tutorials and help to assist with the use of the system and the data.

Four-processor hardware systems such as the IBM J30 SMP machine allow us to attain the targeted performance level—processing times of less than 1 sec by the back-end server—for simple queries made of the 1% PUMS. Moderately complex queries can increase processing times by a factor of 10 or more. Extending the system to 32 higher performance processors will allow complex queries to be made of the 1% data and simple queries to be made of the 5% PUMS and larger data sets within the 1-sec time frame.

The design model is client-server with the user interface running on a local desktop machine. For smaller data sets, all of the components of the system can be integrated on a single machine, but the system is designed to be distributed across as many processors as required to deliver the performance desired. Queries are passed from the user interface to a front-end subsystem that handles a variety of tasks, including queuing queries for execution, compiling queries and distributing the code to the back-end execution engine, aggregating results, returning results to users, and controlling the data-management subsystem. More mundane tasks, such as user authentication and accounting, are also handled by the front end.

Processing time is the primary performance criterion for the PDQ-Explore system. The overall speed of the system determines, first of all, whether true interaction with the data is possible. It also determines, in large part, how many simultaneous users can be handled while maintaining interactive performance and the size of the largest data set that can be accessed at interactive speeds. Each component and each subsystem of PDQ-Explore is a potential obstacle to attaining the performance target we have set. System latencies of 1 or 2 sec, for example, are of no consequence in a task that requires 10 min to execute, but fatal for a task that is intended to execute in 1 sec.

The original Explore software was programmed in Fortran with some of the early code translated to C. PDQ-Explore is now being written in C++, resulting in significant performance increases due, in large part, to the more efficient data structures that can be implemented in C and the translation of queries to C source code that is then compiled for execution. We transpose the data matrix and store the data items, one to a file, on the mass storage system. These are, in turn, divided into subfiles that are stored on disks local to each of the parallel processors and then cached to RAM upon demand. Thus, 32 processors might be assigned to the task, with each processor having $1/32$ of each item's data on a disk local to it. The data-management subsystem handles the subsetting of the items and the distribution of the item subfiles across the disks on the available processors. If hardware components fail, the data and tasks are automatically assigned to the available processors.

In the following sections, the design issues and the balancing of cost and performance in developing the PDQ-Explore system are considered. Figures 3 and 4 present, respectively, a schematic view of the hardware base and an example display of the GUI for the system. The interface shown is for tabulations. Similar interfaces are being developed for other system functions—summary statistics, correlations, n-tiles, and extracts. The figures will help convey a sense of the nature of the system.

Design Considerations

The technical design challenge in creating PDQ-Explore has been largely one of maximizing overall system performance while achieving balance in performance among the subsystems and components that comprise the hardware platform. Cost effectiveness and the budget constraints of the small business research programs that are funding the project dictate that the best performing mass market components be used as the hardware base for the system. While a Silicon Graphics Challenge SMP system configured with 36 processors, 16 GB of RAM, and 256 GB of high-performance parallel disk would be an ideal hardware platform for an integration of PDQ-Explore with one or more standard data-analysis packages, the cost would be several times that of the funds

Figure 3. PDQ-Explore hardware platform. The basic components are: parallel processors consisting of dual Pentium Pro boards housed two to a cabinet; the J30 SMP, which serves three primary roles in the configuration—front end to the parallel processors, mass storage controller, and communication hub/concentrator; the mass storage disk system; and the fast Ethernet network that links the components. Users access the system via the Internet using either the graphical or command line interfaces. The graphical interface is linked to the front end through sockets when the query is sent and the response received. The query can be sent to any compatible system. Fast Ethernet links the components of the platform. Each processor in the parallel server is connected to the front end through a hub, with each hub initially handling eight processors and having a direct connection to a network card in the front-end machine. The storage subsystem is, in effect, a three-level hierarchical storage system with data distributed on demand from the mass storage server to the local disks linked to each processor and then to the RAM cache associated with each processor.

Figure 4. Prototype user interface to PDQ-Explore. Example windows from the PDQ-Explore prototype graphical user interface (GUI). The interface is coded in TCL/Tk—Tool Control Language and Tool Kit for X-Windows—which is available for UNIX-based systems, Windows 95, Windows NT, and Macintosh operating systems. An HTML viewer is integrated to provide access to documentation and reference materials. Multimedia instructional and tutorial materials are being developed for integration with the interface. The example displays output from the tabulation of a *relationship to head* recode by *age* scaled to 5-year cohorts for the selection *race* equal to "1" (whites).

available for the preproduction system to be implemented in 1997 and well beyond a cost level that can be supported by the users currently envisioned for the system. However, a hardware platform with those features may be a perfectly practical choice 4 or 5 years from now if the size of the user base grows sufficiently to support it.

Generating a response to a query in PDQ-Explore involves a sequence of 10 steps. The performance target for any given query is to complete these steps within an overall 5-sec time frame so that the user receives the response within 5 sec of sending the query. Pipelining the steps through two or more subsystems can allow throughput rates approaching the one-query-per-second target for the back-end execution engine. Simple queries for even multimillion record data sets can be easily handled within this time frame given enough processors and RAM capacity for the execution engine. However, any one of four of these steps can, under certain circumstances, consume

the whole 5 sec or more. The *critical steps* are denoted in **bold** in the following list.

1. Receive the query
2. Queue the query for execution
3. Translate the query to C++ code
- 4. Compile the code**
5. Broadcast the executable code to the query engine processors
- 6. Map the required data to RAM for each processor**
- 7. Execute the query on each processor and the corresponding portion of data**
- 8. Retrieve the partial results from each processor**
9. Aggregate the results
10. Return the results to the user

These steps can be distributed across multiple processors to reduce response times. The fundamental I/O prob-

lem lends itself well to parallel solutions. The system scales well. Demonstrations have been run on hardware platforms ranging in size from a single processor to as many as 32. The present model uses a front-end processor to handle everything except the mapping of the data to RAM and the execution of the query, tasks that are shared at the back end by as many processors as are available for the parallel-execution engine. Multiple processors or an SMP system can be used as the front end to reduce or eliminate any bottlenecks that arise there.

The original Explore in effect interpreted a query as it was executed. Compiled code generally executes significantly faster than interpreted code. For PDQ-Explore, compiling the code results in 10-fold or better increases in performance over interpretation at the execution stage. To attain maximum performance, the C++ code must be compiled with certain optimization parameters set. As a consequence, for some complex queries, the compilation phase can require several seconds when using any of the currently available Pentium and PowerPC systems. Faster processors, more efficient compilers, and parallel compilation will, if necessary, allow the compilation time to be reduced to under 1 sec. In early 1997, we will adapt an existing compiler or write a custom compiler optimized for the needs of PDQ-Explore to further reduce the compilation time. Because the target system is limited to executing a narrow range of tasks, significant increases in speed will be achieved over those attained with a general-purpose compiler. We expect eventually to achieve negligible compilation times.

The performance issues for managing the data and executing the queries in Steps 6 and 7 arise from the size and number of data sets to be handled, usage patterns, and the complexity of the query posed. Optimizing for performance requires that all the data needed for responding to a query be cached in RAM at the beginning of the query and that the inner most executable code be held within the on-chip instruction cache of each processor during the pass through the full data set. With single-disk systems, best performance is achieved when a single vector of data at a time can be retrieved sequentially from disk and cached to RAM. This minimizes the disk seeks that occur when multiple items are being accessed simultaneously at different locations on the disk. The disk I/O subsystems we have tested to date do not fully meet our needs. We expect to obtain or write I/O routines in early 1997 that will meet our needs.

Given the cost of RAM and/or the limitations on the amount of RAM that can be installed on processor boards along with the size of the data sets we wish to handle, not all of the data can be cached in RAM. Dynamic caching algorithms are used in PDQ-Explore to keep the most recently used items in the cache. The long-term likelihood for finding any given item already cached depends upon the size of the cache and the patterns of usage of the data sets and items within data sets. For any given query, the likelihood depends upon the data set, items referenced in the query, and the most recent history of usage. Among users, some data sets and some items within data sets will

be much more popular than others. But the possibility always exists for a query to require that extensive quantities of data (megabytes per processor) be retrieved from disk and for a combination of users to be competing over a short period of time for more cache space than is available. Thrashing behavior on mainframe systems was a serious—but fortunately infrequent—problem when competition for scarce resources resulted in most of the processing time's being devoted to resource management rather than task execution. PDQ-Explore does not face exactly that problem, because task queuing eliminates the most direct competition for system resources, but execution times could be increased from 1 sec to many seconds if extensive and repeated input of data from disk is required. We do not want that to happen.

Paralleling disks through disk striping (spreading data across two or more disks in such a manner that data flow from one disk overlaps the seeks on another, thus providing continuous data flow from the disks) can increase the overall disk throughput. But providing parallel disks for each processor, especially in a loosely coupled system of processors, greatly increases the number of disks required and does not obviate the dependence upon RAM. We expect to install two high-performance disks and one disk controller per processor in the preproduction system to be implemented in 1997.

RAM is also essential for accumulating and storing intermediate results. Extensive RAM can be required for tables. For example, a state-by-state tabulation of occupation by industry, each with approximately 1,000 categories, for four racial groupings defines a 200-million-cell table requiring 800 MB of RAM per processor to accumulate. Even though such a tabulation may be easy to specify, no attempt will be made to execute it interactively. The user may resubmit it in the form of partitions of the full table or, perhaps more reasonably, collapse occupation and industry into fewer categories to define a manageable problem. We can also provide off-line consultant services to handle special needs.

Complex data transformations and recodes can also place heavy demands upon the system. Some may require more than one pass of the data from beginning to end—in calculating standard scores, for example. Others, such as computing a household-level SES score as described earlier, may require cycling through the data from all of the records in a housing unit twice, once to create an index based on information from several or all records in the housing unit and then, the second time, to use the index while executing the remainder of the query. These tasks can significantly increase the processing time. Many more instructions must be executed per item and per record than is the case for simpler queries made directly from the raw data. The speed increases made possible by compiling the queries into directly executable code compensate for some of these demands. In a worst case scenario, the executable code would be so large that the innermost code that is executed for every record accessed for the query could not be held fully within the processor instruction cache, requiring that code be brought in

from the off-chip L2 cache as each data record is accessed. This can greatly increase execution times. Again, we do not want this to happen.

These concerns can be addressed in several ways. The simplest is to allow only those transformations and recodes that do not overly tax the system. A second would allow complex new items to be created off line and added to a data set for use later. A third would share the processing over a pipe of processors so that each processor in the pipe can hold its portion of the code within its instruction cache. A fourth would be to wait until processors are available on the market with the cache sizes and power to handle the most demanding problems directly. None is an ideal solution. Experience may dictate that we will need to move from one approach to another. Initially, we expect to take the first approach: allow only transformations and recodes that do not overly tax the system. These can still be very powerful, both in their capabilities and in the ease with which they can be specified. Complexity as great as that required for the SES item-creation example should be possible.

Even if the RAM were available to create the 800-MB occupation by industry table referred to earlier, current network speeds would eliminate any chance of aggregating the results and returning an 800-MB table to the user within the target of 5 sec. Minutes, even hours, could be required with the return of the table to the user via the Internet, presenting a significant bottleneck, even for fast network links. An SMP system large enough to support common RAM for all the processors would eliminate the external communication problems associated with aggregating results for very large tables from loosely coupled processors. In a single, large SMP system, either all of the partial tables would already be accessible in RAM to the processor assigned to do the aggregation or the accumulation of results might be possible in one table accessible to all processors as they executed the query and the data were passed. For PDQ-Explore, this is perhaps the major advantage that SMP technology has over loosely coupled parallel systems. However, providing the necessary memory bandwidth to support multiple processors in the SMP machines contributes significantly to the relatively high cost of the systems.

As suggested earlier, tasks such as locating the percentile, decile, and/or quartile points in an income distribution can be handled easily by generating the full tabulation for an income item in the 5% PUMS (perhaps as many as one million categories—a 4-MB table), aggregating the results, determining the desired points in the tabulation, and returning to the user only the point values. The original Explore allowed a single percentile value to be obtained through a two- or three-step successive approximation procedure that narrowed the range at each tabulation. This process will be automated in PDQ-Explore or replaced by a one-pass determination of all the percentile values when sufficient RAM is available for the full tabulation of an item.

During early 1996, PDQ-Explore was tested on a four-processor IBM J30 SMP system to determine, among

other objectives, if dividing the overall task stream of PDQ-Explore into subtasks that could be distributed through a pipeline of processors would result in net performance gains. Our empirical tests indicated that the subtasks in PDQ-Explore were too unbalanced to allow for efficient pipelining. At best, pipelining typical tasks through four processors yields small gains over a single processor but not nearly the fourfold gains needed to be competitive with four processors running in parallel. Upon reflection, we realized that our queries would have to be divisible into almost pathologically balanced pieces of code for pipelining at this level to net any gain over parallel processing. Determining how to optimally split any given code sequence into balanced segments or dynamically managing the distribution of unbalanced segments to load the processors as evenly as possible over time would be extremely challenging, and likely fruitless, tasks.

The current version of PDQ-Explore, running on parallel processors accessing data from RAM cache, is approaching the limit of overall processing efficiency that can be achieved when memory access times, bus bandwidth, and processing power are well balanced with one another and with the query tasks. PDQ-Explore now uses a front-end design that translates queries to C++ code, compiles the code with optimization, and then broadcasts the executable code to the parallel processors for execution of the query. Single processors in the SMP (75-MHz 601 PowerPCs at present) perform at about the level of a single Pentium 90. This approach yields processing rates in the 1-million-records-per-second range for those processors. Faster processors—200- to 300-MHz Pentium Pros and 133- to 200-MHz 604e PowerPCs—may double or triple these rates, although memory bandwidth will at some point become a constraint, especially on the significantly lower cost PC systems. The J30 uses a crossbar memory bus configuration that provides several times the bandwidth of PC-based machines. However, our recent tests with a single-processor 200-MHz Pentium Pro-based machine have indicated that these increases in throughput can be readily attained with simpler memory configurations. We expected to use a dual- or quad-processor Pentium Pro board as the basic component in the preproduction platform, to assemble a single system based on that board, and to evaluate its performance late in 1996, before expanding the system to 32 processors.

To meet our performance target, the hardware system must be dedicated to the information system. Performance depends in part upon both the external RAM cache and the on-chip instruction cache associated with each processor being preserved during the execution of a given query. The presence of competing tasks can dramatically increase execution times to minutes and even an hour or more under worst case conditions.

Balancing Costs and Performance

RAM was used to cache data in the original Explore installation to circumvent the input bottleneck that has characterized the handling of large data sets. While this

works very well and sets the upper bound on performance capability for a hardware platform based on single-processor machines if the needed data are in the RAM cache, it is an expensive solution. The RAM can represent as much as one-half the cost of a high-performance platform. An alternative we have evaluated is to employ the techniques available for maximizing data delivery from disk and then serving the data to multiple processors in the SMP. The techniques implemented and evaluated included using disks in parallel, striping data across disks, optimizing disk formats for sequential data access (rather than random or direct access), and data compression.

Paralleling disks increases the data throughput by taking data from separate files located on each of the paralleled disks. Disk controllers can typically handle data faster than one disk can deliver the data. If data are needed from two files, a controller can handle two data streams simultaneously from two disks, possibly more than doubling the transfer rates attained if the two files were located on one disk. Disk striping distributes data across multiple disks, again allowing for higher transfer rates than can be attained from a single disk. Either technique can contribute to better performance. Optimizing block sizes and storing each data vector in sequential disk locations—eliminating fragmentation of files—also contribute to better disk performance. Compressing data files by using the routines built into IBM AIX 4.0 slowed disk performance significantly but did reduce file sizes by 50%.

Our initial disk evaluation indicates that disk performance can be substantially improved by using multiple drives on each processor, but overall performance of PDQ-Explore declines significantly as disk is substituted for RAM cache as the primary input source. By writing disk I/O routines that optimize for the requirements of our system, we believe we can significantly increase disk performance and substantially reduce the RAM needed to sustain acceptable performance, although patterns of usage can have significant effects. We will start with 128 MB per processor on the preproduction system and increase the RAM as early usage reveals the optimum balance among the components.

The high processing rates we currently achieve are possible only when the data are cached in RAM or when I/O and CPU demands are evenly balanced. The rates cannot be sustained with the input directly from disk, because we need to step our way, in parallel, through the items that are stored each to its own file. If the data are not cached, accessing more than one file at a time requires that seeks be made from one file to the next as we work our way through the data set. These seeks can reduce the overall throughput from the disks to 10% or less of normal transfer rates, depending in part upon the caching done on the disk and disk controllers and the caching and read-ahead algorithms used. In general, the most efficient way for PDQ-Explore to access the data is to read the data item by item into the RAM cache so that when an item is accessed, the full sequence of serial reads from the disk are

executed without interruption. Then, when all the data are in RAM, the query task can be executed. While some optimization of I/O can be done to speed the transfer of sequential data, disk I/O routines normally try to maintain a balance among the files as they are read in, repeatedly seeking from one file to another to another. This is not a serious problem if the task is compute bound—the processor (or processors) stay busy during the seeks, and useful work gets done. The consequences can be disastrous for a program such as PDQ-Explore, which can be seriously I/O bound. Over the course of processing a few relatively small files from disk, we have observed seek times consuming more than a minute while the processor needs only a couple of seconds to accomplish its task. Thus, the processors remain idle during most of the elapsed time required for the query to be executed.

Assuming that we want tasks to be executed in 1 sec, distributing the data so that each processor handles approximately one million records from any given data set can meet that need if the desired items are in RAM. If not, missing items must be read into the cache. A moderately complex query may reference eight items, perhaps stored in a total of 16 MB, for the million records. In our tests, typical fast PC disk and controller combinations can transfer data at sustained rates on the order of 3 MB per second. Thus, 5 sec or more may be required to move all the items from disk to the RAM cache, with less time needed if some items are already cached. The actual transfer rates attained by PDQ-Explore, when monitored directly using the UNIX `iostat` command and as inferred from the difference in execution times between tasks executed when items are in cache and when they are not, indicate somewhat slower rates, ranging from 100 KB to 2 MB per second. The low rates are obviously associated with considerable seeking on the part of the disks.

Some increase in performance can be achieved through careful selection of the disk system. Fast Wide SCSI II disk systems roughly double these performance figures. Single disks on the IBM J30 can sustain 7-MB-per-second transfer rates. Disk controllers can support two or more disk drives at full transfer rates. Thus, retrieving files located on two parallel drives has the potential to double throughput. Tests indicate that performance is, in fact, almost doubled. Simply paralleling drives does not guarantee faster access, however, because the items needed may be, by chance, on the same drive. Distributing files across parallel disks using disk striping can guarantee an increase in performance, although typically with a modest loss in performance due to the overhead of managing the distributed file—two disks will not deliver quite twice the performance of one. We may also be able to gain some increase in overall performance by passing requests for items on to the processors when a task is queued for execution rather than waiting until the executable code is distributed to the processors.

We conclude from our experience with the disk systems that we need to address directly the need to maxi-

mize the transfer rates when multiple sequential files need to be read to the RAM cache. We expect to be able to stripe across two high-speed disks and maintain sustained transfer rates of 15 MB per second. To attain that performance, we will obtain, adapt, or write disk routines to meet the specific needs of the system. Doing so will significantly reduce the time penalty incurred when items must be read from disk into the cache.

A second data transfer concern stems from the occasional need to move large quantities of data representing intermediate results from each of the parallel processors to the aggregating processor. The problem here is primarily the bandwidth constraints imposed by the network links among the processors. SMP technology offers an elegant solution to this problem by removing the need to move the data. All of the intermediate results are held in RAM common to all the processors. In fact, one can imagine an implementation in which all of the processors access a single table in the common RAM, thereby eliminating the need for a separate aggregation of the results. This would present some interesting, but likely solvable, problems related to synchronization of the processors and their access to RAM.

The SMP architecture also has the advantage of eliminating the need for multiple copies of the operating system, system daemons and related information, and query-related look-up tables and reference data in RAM. This can potentially either eliminate the need for several hundred megabytes of RAM or free that amount for more productive use.

We have not thoroughly tested pipeline strategies for increasing the speed of execution of queries on the SMP machine because our early results indicate that pipelining is a very inefficient strategy for handling the processing loads imposed by PDQ-Explore. Pipelining can provide gains in performance if the load can be evenly distributed down the pipe and if pipelining provides much higher "hit" rates on the instruction caches than those yielded by a single processor implementation. If not, paralleling the task is much more efficient than pipelining for any given number of processors. The executable code for typical PDQ-Explore queries can be held within the on-chip instruction caches of current processors such as the Pentium and PowerPC. The combination of a large RAM and the multiple processors can be very advantageous when such coarse-grained pipelining as directing output from PDQ-Explore as input to SAS or SPSS is to be done. The need to move data over an external network is eliminated.

One strategy we are investigating that can be uniquely supported by SMP architecture is the use of one processor to manage the RAM cache. The front-end machine would pass item requests to the cache manager as queries were received, so that the input of items not in the cache could begin as soon as the query is received rather than deferred until the compiled code requests the items at execution time. This has the potential to allow more efficient use to be made of the cache space, perhaps allowing for smaller caches to be used.

While data compression can significantly reduce the number of bytes that must be moved from disk to memory, uncompressing the data requires time and processing power. For example, a query requesting mean income by sex for college graduates fully employed in 1989 who were 23 years of age in 1990 might require that 8 bytes of data be read for each of the 12.5 million persons in the 1990 5% PUMS data set, a total of 100 MB. Assuming an eight-disk system that can retrieve the data at a true rate of 3 MB per second per disk, a little over 4 sec will be required to retrieve the data. If the data can be compressed by a factor of 4, the time reduces to 1 sec if the uncompression and subsequent processing can keep up with the maximum data flow. Even if the processor cannot sustain the maximum flow, it may still allow a net gain in performance over the use of uncompressed data. If a given single-processor system cannot realize a gain in throughput, which is generally the case with currently available systems, dedicating a processor to uncompressing the data may increase the data flow. Our tests, however, indicate that all the available processors should run the full task in parallel, including uncompressing the data, rather than having one or more uncompress the data and the other processors perform subsequent tasks.

Conclusion

PDQ-Explore, along with the data from all available years of PUMS and March Annual Demographic CPS data from the Bureau of the Census and Mortality Detail files from NCHS, will be installed on a preproduction hardware platform in early 1997. Other data sets will be added as they become available. Assuming that continued public support for development is obtained, the service will first be made available via the Internet to National Institute of Child Health and Human Development (NICHD)-funded population centers and researchers, other members of the Association of Population Centers (APC), and via local networks to the University of Michigan research community. Additional users will be recruited, as required, to determine the hardware performance capabilities and limits of the system. Performance bottlenecks will be identified and adjustments and additions made to the hardware configuration and to access procedures/priorities as required to optimize overall performance.

We propose to assemble first a 32-processor system based on dual- or quad-processor Pentium Pro boards with 4 GB of RAM distributed across the processors—128 MB per processor—and the equivalent of a small redundant array of inexpensive disks (RAID) for each processor. The PDQ-Explore software will be installed on the hardware platform and used and evaluated for several months. Then early in 1998 the hardware configuration will be extended to increase performance where needed most—processing power, RAM, disk, and/or linking network.

We anticipate that no charges will be assessed to non-commercial users during the 1997–1998 period, although this will depend upon receiving continued support through the end of 1998. One current grant will fund the project

into 1998 and a pending grant will continue the work through 1998. Some cost recovery may be possible from commercial users during the 1997–1998 period. By 1999, the project is expected to have matured into a fully commercial, self-sustaining service. Some of the early users from academic and research institutions are expected to become part of the paying user base that will support the service at that time.

PDQ-Explore, functioning as an innovative, dedicated research instrument, will provide social science researchers as well as less knowledgeable users with unique access to a variety of large census and survey data sets. The overall challenge we face as developers of the system is to balance the number of users, their behavior in using the system, the sizes of the largest data sets to be handled, the complexity of the queries to be handled, and the performance of the system to meet the requirements of a large enough community of users to support the system. Each subsystem—user interface, front-end processor(s), data management, back-end server, and linking networks—will be optimized to take full advantage of the performance capabilities of the corresponding hardware components. Disk subsystems and processors will be paralleled to meet the performance requirements of users. Our hope is that by providing easy, fast, low-cost, and meaningful access to these large data sets, the data sets will be more fully utilized and the public as a whole will become better informed about the broad range of social, demographic, economic, environmental, and public health issues to which the data are relevant.

Information on the current status of the project may be obtained at <http://www.pdq.com>

REFERENCES

- ANDERSON, A. F., & ANDERSON, P. H. (1994). *Interactive access to large census data sets* (PSC Research Report 94-315). Ann Arbor, MI: University of Michigan, Population Studies Center.
- ANDERSON, A. F., HANSS, T., & HERMALIN, A. I. (1991). *A contemporary research computing environment: Evolution transformed to revolution* (PSC Research Report 91-228). Ann Arbor, MI: University of Michigan, Population Studies Center.
- BUREAU OF THE CENSUS (1992). *Public use microdata samples, United States: Technical documentation*. Washington, DC: Author.
- CSM STAFF (1993). *User's guide to CSA: Conversational survey analysis*. Berkeley: University of California Computer-Assisted Survey Methods Program.
- DAVID, M. H., & ROBBIN, A. (1990). Computation using information systems for complex data. In *Proceedings of the First Conference on Advanced Computing in the Social Sciences* (pp. 8:25-66). Williamsburg, VA: Social Science Computing Association.
- DEPOMPA, B. (1996). Wal-Mart orders a super warehouse. *Information Week*, 570, 22.
- DEWITT, D., & GRAY, J. (1992). Parallel database systems: The future of high performance database systems. *Communications of the Association of Computing Machinery*, 35(6), 85-98.
- FARLEY, R. (ED.) (1995). *State of the union: America in the 1990s* (Vols. 1 and 2). New York: Russell Sage.
- FOX, D. J., & GUIRE, K. (1976). *Documentation for MIDAS*. Ann Arbor, MI: University of Michigan, Statistical Research Laboratory.
- KAKWANI, N. (1980). *Income inequality and poverty: Methods of estimation and policy applications*. Oxford: Oxford University Press.
- MCDANIEL, C. (1994). *GAHM*. College Park, MD: University of Maryland, Academic Software Group, ePress Project.
- MEIJ, H. (1996, May). *Accessing CIESIN data on the web via Ulysses*. Paper presented at Population Association of America annual meeting, New Orleans.
- MICHALEWICZ, Z. (ED.) (1991). *Statistical and scientific databases*. London: Ellis Horwood.
- MILLER, D. (1988, October). The great American history machine. *Academic Computing*, 2, 28-48.
- RADEN, N. (1996a). Maximizing your warehouse. *Information Week*, 571, 42-48.
- RADEN, N. (1996b). Boost warehouse performance. *Information Week*, 572, 77-89.
- SHANKS, J. M. (1989). Information technology and survey research: Where do we go from here? *Journal of Official Statistics*, 5(1), 3-21.
- STONEBRAKER, M., & DOZIER, J. (1991). *Large capacity object servers to support global change research* (SEQUOIA 2000 Tech. Rep. 91/1). Berkeley: University of California, SEQUOIA 2000 Project.
- TAYLOR, I. (1992, May). *Tablemaker* [Computer program]. Demonstration presented at the Third Conference on Computing in the Social Sciences, Ann Arbor, MI.
- TUCKER, J. (1996). On line transaction processing on an MPP: A case study. In *Proceedings of IBM Super! 1996 Conference*.
- WIEGNER, K. (1994). Illustrate charts its own path. *Information Week*, 498, 48-52.