tableBASE

Concepts and Facilities Manual

Release 5.1

Technical Support Hotline (613) 523-5588

To order additional manuals:

Data Kinetics Ltd.
Sales Division
2460 Lancaster Road
Ottawa, ON
Canada K1B 4S5

Telephone	(613) 523-5500
Facsimile	(613) 523-5533
E-Mail	tablebase@dkl.com

Data Kinetics Home Page http://www.dkl.com

Table of Contents

Preface

What this Manual is About

The pace of change continually increases. Companies that cannot keep pace fail. Many organizations around the world have turned to tableBASE, a software product from Data Kinetics Ltd., to ensure that they do not merely cope with change - they master it.

This manual provides an overview of the concepts and facilities found in tableBASE. In particular, it shows you how, through tableBASE, you can not only master change but also improve both the performance of your computer applications and the productivity of your IT staff.

Who this Manual is For

This manual is intended for organizations that want to use information technology as a competitive weapon. You will find that all members of your IT organization will benefit from tableBASE. This document should be read by:

- Managers of Information Services
- Systems Analysts
- Programmers, and
- End-Users, who are not familiar with tables or tableBASE.

Additional tableBASE References

This manual is one of several that describe tableBASE, the others are:

- tableBASE Batch Utilities Manual
- tableBASE Programmer's Guide
- tableBASE Administrator's Guide
- tablesONLINE/CICS User's Manual
- tablesONLINE/ISPF User's Manual
- tableBASE VTS Server Guide
- tableBASE Installation Manual
- tableBASE Workshop Manual

Chapter 1 Introduction

The principle of separating business rules from processing logic has long been used to create a wide variety of sophisticated computer applications, ranging from operating systems to stock trading. Separating rules from logic - by placing the rules in external tables - makes it easier to develop applications and, most importantly, gives you the power to rapidly modify the application to adjust to our constantly changing world. With tableBASE you can readily develop table-driven, rules-based applications, thus improving your staff's productivity. And, because tableBASE manages the tables or rules in memory, it also turbo-charges your application's performance.

Sample tableBASE Applications

Here are a few of the many innovative ways tableBASE has been used:

- A software development firm used tableBASE to convert all of its data for Y2K compatibility.
- A direct marketing firm uses tableBASE to improve DB2 performance by 60% - 70%. In one application CPU time was reduced from 15 to 1 minute, a reduction of 93%.
- A major insurance company relies on tableBASE to cut program maintenance times and costs. In one mission-critical application the number of program modules went from 81 to 1.
- A securities firm quadrupled the CICS transaction throughput of its on-line stock trading system from 50 to 200 transactions per second.
- A major manufacturer reduced CPU time in one IMS application by 20% and elapsed time by 45%.
- A non-programmer at an investment bank created over 150 CICS screen panels and associated table maintenance routines in two weeks, versus an estimate of 12 - 18 months using conventional tools.
- A major West Coast bank restructured a system whose tight operating window restricted financial performance. The first night the restructured system was run it saved millions of dollars in interest as it cut elapsed time from 7.5 hours to 28 minutes.

- A HMO developed a new Explanation of Benefits form, eliminating twenty hours of maintenance programming per month and freeing six people from a front end telephone screening process.

Why you should know about tableBASE

There are many reasons why you should know about tableBASE:

- If you are the **Data Administrator** or **Manager of Information Services**, tableBASE helps you to make the most effective use of your resources and respond to changes in corporate directives quickly and easily.
- If you are a **System Designer/Developer**, tableBASE shortens development time, simplifies and reduces maintenance and increases design flexibility.
- If you are an **End User**, tableBASE makes your systems more responsive to your needs and lets you exercise more direct control over their behavior.
- If you are an **Operations Manager**, tableBASE simplifies operations and dramatically increases execution speed and overall system throughput.
- If your main concern is the **bottom-line**, tableBASE is a cost-effective way to hone your competitive edge while reducing the demands on your human and computer resources.

Organization of this Manual

Here is a summary of how this manual is organized:

- The second chapter presents a high level **overview of tableBASE** and its components.
- The third chapter explains several important heuristics for distinguishing between **data suitable for main memory table processing** and data which should be processed using traditional I/O methods or database management systems.
- The fourth chapter explains where **tableBASE belongs in your environment** and why.
- Chapter five shows how **tableBASE**, using algorithms designed specifically for memory based processing, **improves** the execution speed and **response time** of your applications.
- Chapter six describes how **tableBASE streamlines the applications development effort**. tableBASE works with conventional programming

languages and established development techniques to accelerate system development and enhance programming and maintenance productivity.

- Chapters seven and eight demonstrate how tableBASE and tablesONLINE can help you **create table driven, rule based systems**. Included are examples of ways to build and maintain your tables. Chapter eight also shows how tablesONLINE can be used as a powerful application generator.
- Chapter nine contains **technical specifications of tableBASE**.
- Chapter ten describes the company behind tableBASE and the **support** and services available for this software.

Chapter 2
Overview of tableBASE

tableBASE

tableBASE is truly a unique software tool for IBM or compatible mainframe systems. It gives you the power to master change and provides significant **Productivity AND Performance** improvements.

On the Productivity Side

tableBASE provides application developers with a centralized system for managing all your data tables. tableBASE automates the creation, maintenance and control of main memory tables in your application programs. In fact, it provides all the convenience and facilities for managing these tables that a DBMS provides for managing files - with one significant difference. Applications using tableBASE services can **access data at memory speeds and avoid the I/O overhead** associated with accessing data from DASD.

Tables are found in every area of data processing. However, they are often managed by products that are disk based, that is, products which are architected for managing *files*. tableBASE is architected for efficiently managing *tables* which are completely resident in memory. It is being used today to meet a variety of data processing objectives:

- To promote standardization and system flexibility
- To minimize the tedium and cost of program maintenance
- To increase system life and efficiency
- To reduce development efforts
- To cost effectively re-engineer systems
- To simulate and evaluate the impacts of processing decisions, and
- To dramatically reduce physical I/O.

tableBASE provides an efficient and integrated infrastructure that lets you take control of all data tables, from small parameter tables to huge "scratch pad" extract files. tableBASE integrates easily into your existing application software environment. It can be called from any common language using standard IBM protocol. Along with its optional tablesONLINE component, it also forms a complete table driven application development environment. You can easily construct interactive menu-driven systems for data entry and retrieval.

On the Performance Side

tableBASE is also a sophisticated memory management tool. It dynamically manages tables in memory - and manages the memory it uses. Applications that use tableBASE obtain data at memory speeds - 1,000 times faster than disk access. tableBASE has facilities that ensure memory resources are used effectively.

tableBASE makes using in-memory tables simple and efficient, above the line in XA, or even in shared dataspaces under ESA. The results are reductions in I/O, CPU cycles, application run-times and on-line response of up to 90% over I/O based methods.

In recent years, main memory capacity has increased tremendously. There is every indication that this trend will continue. In the past, larger address spaces have dictated modifications to existing software. Less obviously, they have also dictated new approaches to system design. tableBASE technology is positioned not only to take advantage of today's increased memory size, but also to adapt applications automatically to future expansions. Its own evolution has centered on making it easy for program developers to dynamically manage data in memory.

Components of tableBASE

tableBASE is architected to provide maximum efficiency in a variety of environments.

The Base Product

The base product consists of the tableBASE nucleus and the TSO/BATCH interface. It can be accessed by any program using standard IBM protocol. The base product is currently available for VSE, MVS and OS/390 operating environments.

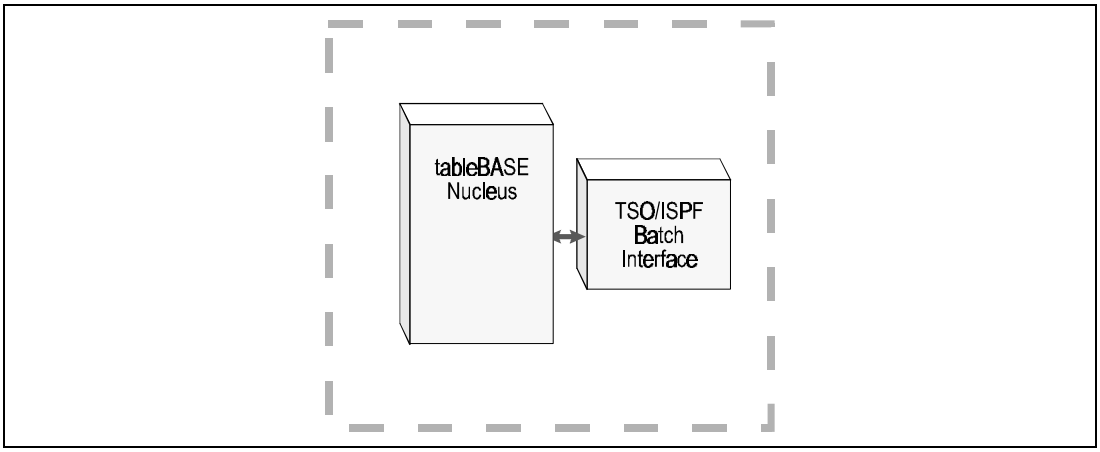

Figure 2-1: Base Product Components

Optional Environmental Interfaces

Optional components of tableBASE are available for the CICS and IMS/DC Teleprocessing Monitors. Both interfaces operate in the same manner and provide the same facilities as the BATCH/TSO interface, but for online applications. All environments may share access to tableBASE libraries on disk. Private copies of tables may be loaded and accessed in each unique environment through the respective interface. Access to shared main memory tables in MVS/ESA and OS/390 is available from any environment, batch or online, through an interface to the optional Virtual Table Share (VTS) subsystem.

Virtual Table Share (VTS)

The Virtual Table Share (VTS) component of tableBASE lets you exploit the huge memory resources available in MVS/ESA and OS/390. tableBASE/VTS allows applications running in different operating environments (Batch, CICS, TSO/ISPF, IMS/DC, DB2, etc.) to share data residing in a common managed memory pool.

Since only a single copy of data is loaded into the shared dataspace, I/O and system paging is even further reduced. Yet, tableBASE's inherent flexibility allows each application to have several different views of the same data.

tableBASE/VTS lifts traditional restrictions in memory usage, giving application designers the opportunity to harness leading edge technology. Your systems can now simultaneously access thousands of small tables and several large tables at memory speeds. tableBASE/VTS takes care of memory management including loading tables into memory. Programmers concentrate on fulfilling business objectives instead of battling with low level details.

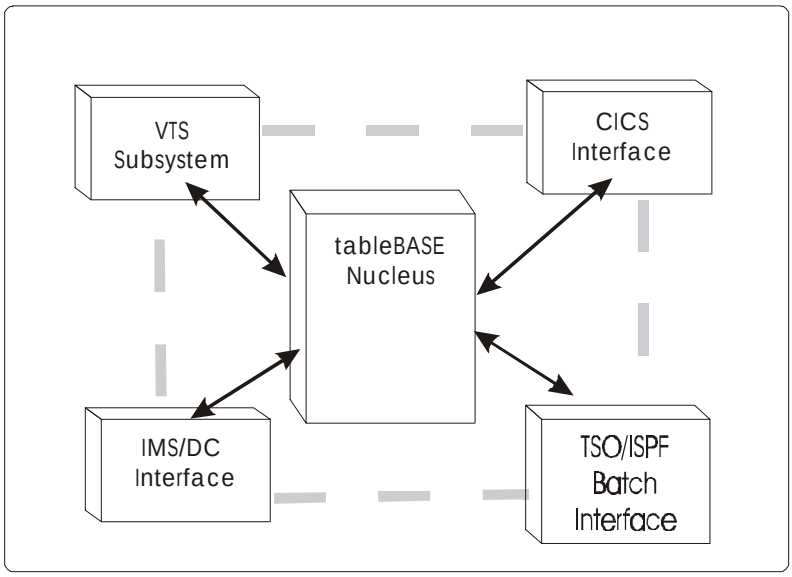

Figure 2-2: Environmental Interfaces

tablesONLINE

tablesONLINE was designed to give **both developers and end-users** access to tableBASE services. It is completely menu-driven. Application developers can define, update, test and process tables without writing a line of code. They create screens and entire applications for end-users by simply filling in the blanks with the desired parameters.

Currently there are versions available for CICS and TSO/ISPF.

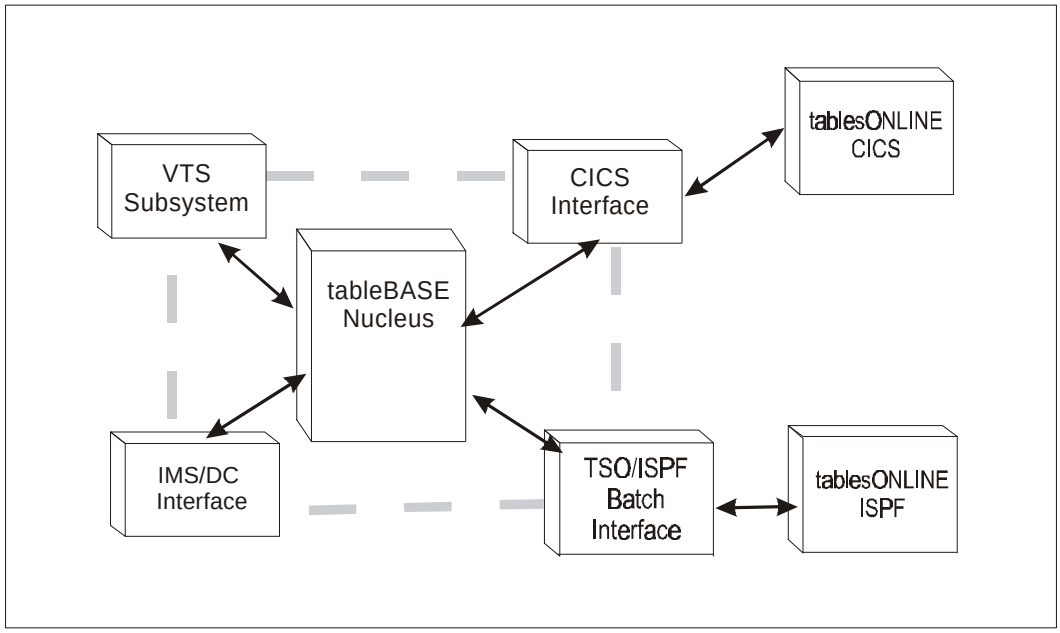

Figure 2-3: tablesONLINE

Chapter 3
What Are Tables?

Introduction to Tables

The de facto definition of the term "table" is a fixed length data structure consisting of a series of rows and columns, independent of a particular internal or external storage organization. Tables may be implemented in a number of ways:

- As files or database structures on auxiliary storage devices, under control of an appropriate file access method or database management system to facilitate maintenance of the data.
- As array-type data structures resident in virtual storage, hard-coded in the application program for optimal performance.
- As data structures resident in virtual storage, external to the application program, under control of an object management system such as tableBASE for combined advantages of easy maintenance and high performance.

Tables are simple data structures that match the user's perception of the data, simple lists that can be modified or extended as additional data becomes relevant. They are often thought of as having two parts - a key (location or an argument) and a result. For example:

Reference Tables

Parts Table:

Part No.	Description
444	wing nut
445	small bolt
446	C bearing

Actuarial Table:

Age	Rate
20-24 years	$5.00
25-29 years	5.25
30-46 years	8.50
47-50 years	10.50
51-52 years	11.00
53-65 years	20.00

The tables above are straightforward, simple lists. They often require several accesses during the run of an application and are often shared by several applications.

Decision Tables

Report Traffic Table:

Department	R01	R02	R03	R04....R7	R68	R69	R70	R71
Marketing	Y	Y	N	N.....N	Y	Y	Y	N
Finance	N	N	Y	Y.....Y	N	N	Y	N
Production	N	N	N	N.....Y	Y	Y	N	Y
Personnel	Y	Y	Y	Y.....N	N	Y	Y	Y
IT	Y	Y	Y	Y.....N	Y	N	N	N
Admin.	Y	Y	N	Y.....N	Y	N	N	Y

The table above determines which departments receive which reports. This table contains variables that are often hard-coded as "if then else" type statements. Changes to the reporting chain are now simple table updates.

Access Control Table:

Data Resource (table or file)	User ID	Access Type R-Read/W-Write
payroll	DKL001	R
payroll	DKL002	R/W
commissions	DKL091	W
vacations	DKL092	R
prod/sched	DKL097	R
prod/sched	DKL098	W

In this example, User DKL001 has read access only to the data resource "payroll". User DKL002 has read and write access to "payroll".

Table Data vs. DBMS (File) Data

At first glance, the choice between tableBASE and a file or database implementation for table structures is not necessarily clear. What are the essential differences in functionality? What kinds of data are best suited to each environment?

Fundamentally, databases are file management systems, buffering data at a record or block level. They allow entire tables to be loaded into main memory buffers only as an afterthought, as an exception, with the direct result that operator sets have been developed as a reflection and extension of file management operators. All operators function within a domain defined by the capabilities and limitations of:

1. A buffered approach, where only a portion of the data is assumed to be immediately accessible in main memory, and
2. The physical organization of the data as it resides on a peripheral device.

Database access methods were designed to manipulate data that is primarily resident on disk. This would not prove to be restrictive unless application requirements indicate that the table would be processed more effectively if resident entirely in main memory.

Operators in tableBASE have been designed specifically to manage data which resides primarily in virtual storage, not disk storage. Under tableBASE, all table data is assumed to be resident in main memory; it may be manipulated there with no impact on the external disk organization. In contrast to database systems, tableBASE focuses on main memory management and views the disk as a table objects repository.

For database algorithms, the assumption that only part of the data resides in memory, in buffers, imposes subtle, yet very definite, constraints on operations. All processing of main memory buffers must protect the integrity of data remaining on the disk. This is still true when buffers are defined large enough to contain the entire database. The access design is simply shifted to a main memory environment. The underlying design assumption remains - there is no guarantee that all of the data will be in the buffer. On the other hand, because **tableBASE was founded on the principle of total memory residency, it can provide operators which exploit that environment** to the limit, including:

- Efficient main memory data organizations
- Dynamic reorganization of tables due to
  - modified table structures
  - summarization requirements
  - update processing

  - table space expansion
- High performance search methods
- High performance index structures
- Dynamic index creation
- Dynamic alternate views

Database managers are I/O intensive. EXCPs are issued for accesses to:

- Indexes
- Data records
- Log records

With a focus on main memory, tableBASE avoids unnecessary I/O.

Database indexes are usually either predefined by the database administrator or automatically determined by the database management system at "binding" time, before the application is executed. With tableBASE, indexes are created dynamically at execution time, in main memory; they are not stored on disk.

Many time-honored reasons for logging are suddenly open to question when dealing with structures in main memory.

Databases are typically very large data objects with relatively low volume access patterns. Batch transaction runs may access less than 1% of the data records. Smaller data objects with high volume access patterns are customary with tableBASE tables.

Because of the large volumes of data, I/O intensive nature of the system and inflexible structure of indexes, databases must be reorganized offline, to allow online processing to continue without unreasonably lengthy interruptions. Automatic, dynamic reorganization of tableBASE tables, however, is routine.

Databases are not normally used for temporary data structures; they generally have very long retention periods. Temporary work files are often created in multi-step batch jobs, but at the expense of considerable I/O, I/O which can be avoided with temporary tableBASE tables.

For update purposes under a DBMS, resource locking mechanisms operate at the record or block level; only a small portion of the entire database is enqueued. This allows other tasks to update remaining portions of the database without having to wait for the first task to complete. When a tableBASE table is opened for write, the entire table is enqueued until the table is closed, but other tasks may open the table for read.

Logging Considerations

Logging is I/O intensive. Two duplicate log files are frequently maintained to facilitate recovery after system abends during writes to the log.

Log file management is complex, often requiring the overhead of dedicated subsystems.

Logging and journalling address various high priority goals:

1. Access and updates to sensitive resources must be audited to meet management and/or governmental requirements.
2. Programs must be capable of restarting after abnormal termination with a minimum of redundant processing and without reapplying updates which have already been applied. Batch processing run times, in particular, may be very long and complete reruns are prohibitively expensive.
3. The integrity of database information must be guaranteed. After a program or system abend, updates must be backed out to the last commit point or checkpoint before processing can safely be resumed. Automatic dynamic backout is a feature of many online database environments, while batch systems require that a separate job be submitted to backout updates to the last checkpoint. In the event that the datasets themselves are somehow "damaged", then a previous backup copy of the database can be restored to some established system checkpoint and updates can be reapplied from the log to bring the state of the database up to the last checkpoint before the problem occurred.

Some very fundamental principles of application design must be re-evaluated in light of the feasibility of main memory processing through tableBASE. Reasons 2 and 3 for logging as documented above, for example, may no longer be relevant in a main memory processing environment.

Where is the cost justification for logging overhead now that a 12 hour batch job is reduced to a 40 minute elapsed run time, or a 2 hour job reduced to 5 minutes? If row-level data is only updated in memory, where is the danger for disk data integrity due to program abends? Logging certainly plays an important role in many applications, but just as certainly, there are many other applications where logging represents processing overhead which is now completely unnecessary.

In addition, logging is record-oriented. It is not generally applicable for updates which affect the entire table, including columnar-type updates and reorganizations.

Table Processing Characteristics

How a data structure is to be processed should ideally determine where that data structure should reside. Factors to consider in establishing residency include processing sequence, frequency of access, table size and update activity.

Processing Sequence

Data structures which are processed sequentially do not benefit from residence in main memory and are better left on an external device. They may be handled effectively on a piecemeal basis in relatively small buffers. However, data structures which are accessed randomly or which are accessed over and over again perform significantly faster in memory than on an external device.

Frequency of Access

If the frequency of access is less than once per row, then the table could be processed effectively as a database; if there are multiple accesses per row, then the table should be resident in main memory. Loading data into memory should be avoided, of course, if it's not going to be accessed.

Size

Distinctions based on table size are less significant now, in an era of larger, cheaper memories and powerful virtual storage techniques. Residency in virtual storage, under control of a table management system, is dictated by capabilities and performance considerations, not size.

Some files, due to real memory resource restrictions, are simply too large to load entirely into virtual storage. To most analysts, however, they are still tables. The only difference is that they must, for practical purposes, reside in database environments **until sufficient real resources are available to move them where they truly belong... in virtual storage.**

Update Activity

If there is really a need for auditing or backup and restore features beyond the scope of a main memory oriented table management system, then a database management system is indicated. But this requirement should come under close scrutiny as a costly alternative to efficient processing in virtual storage.

tableBASE is designed to manage main memory table data. For the most part, file data remains in the realm of traditional I/O or DBMS methods. tableBASE is a **complementary technology** to your DBMS, providing the tools for managing and using table data **with minimal**:

- Programming
- Maintenance, and
- Machine overhead.

Data that should be in Tables

Functional Characteristics of Memory Resident Tables

Certain categories of table usage, or function, generally imply that the table processing patterns will fit the above guidelines for memory residency. These categories include semi-stable data, program control data, constants and parameters and transient data.

Semi-stable Data

Semi-stable data has a high ratio of read access compared with write access. The data may even be updated every few minutes, but the implication is that there is a high volume of retrievals between update cycles. This data does not normally need to be in a database. Logging of individual updates is not a requirement. Some examples are pay rates, organizational structure and operations calendars.

Because of such characteristics, these tables have always had special treatment. The actual implementation varies depending on many factors, and can be any of the following:

1. Indexed sequential files (VSAM).
2. In-line data tables - "77 levels".
3. Included ("copylib") tables.
4. Parameter files or run-time statements.
5. Differing search and sorting approaches with in-program data structures.
6. Program logic using external data for process control.
7. Program logic used for classifying values (i.e., summarization).

Many different techniques have been developed for handling and using this type of data because there are many variables involved in trying to balance system performance, programming complexity, system maintenance and reliability.

If one of these variables changes (for example: table size increases beyond the planned for maximum, or a new variable is added to a table or the table must be accessed in a different order or by a different key), then extensive programming, testing and re-testing is needed.

Many of the decisions which programmers had to make in the past are now made automatically and dynamically by tableBASE.

Program Control Data

This is a special class of semi-stable data which contains decision control information, or rules, for determining which routines should be executed under a particular condition or set of conditions. Control values of this nature have traditionally been hard coded in application programs for optimal performance, but this approach introduces a level of program complexity which hampers subsequent maintenance efforts. The data properly belongs under control of tableBASE, accessed by table driven application programs. With tableBASE, parameter sets which are often implemented in the form of program logic or as installation or run-time options can now be viewed as tables. Examples are report distribution lists, user and password tables and data validation tables. These "control tables" have the following characteristics:

1. They are used with very high frequency and must be very efficiently integrated with program logic.
2. They are usually small.
3. Changes are periodic but require careful planning and control.

These tables, because of their small size and high access, are often embedded within the programs using copy library statements. Programmers use ad hoc techniques to maintain these programs, and must re-link and re-test any changes.

With tableBASE, these tables can be organized and easily controlled. The data is still available to the system at memory speeds but tables are now maintained by an integrated system, controlled by parameters you set. You can set up automatic update and backup cycles, and automatic phase-in of new table versions. And this can all be done by updating tables, without resorting to maintenance of the actual software. There is no need to re-link and extensively re-test programs.

Constants and Parameters

Constants, literals and other parameter values that make up a significant part of a program's working storage are suitable for implementation as tableBASE tables. This is another special case of semi-stable data, which should be entirely resident in virtual storage, but easily maintained through tableBASE facilities. This data should be managed as tables rather than embedded in program logic. The typical approach is to place them in working storage with copy members at compile time. They are available to just a single task with no additional I/O overhead. This, however, can result in swollen demands for Dynamic Storage Areas, where interactive on-line usage requires multiple copies of these constants for each transaction generated. Maintaining these constants is also time consuming. Programs have to be re-linked and re-tested for every change.

With tableBASE, a single copy of these rows is instantly available to all tasks needing them with no I/O overhead. Constants are maintained external to the programs that use them.

Transient Data

Transient data lies at the other end of the spectrum from semi-stable data. It is volatile and forms the bulk of working storage in most online and batch applications. Once a process is complete, transient data is not retained. It will typically be regenerated each time the process is initiated. There is no perceived need to create images of the data on disk, with all the attendant overhead. This type of data exists only in virtual storage. Transient data tables have these characteristics:

1. The data is reduced, accumulated, summarized or reorganized from your primary data base.
2. The table can be dynamically reorganized to serve different objectives.
3. It is usually a temporary version of the "official" data. It can be discarded when the objectives are met and regenerated when next needed.

Probably the most frequent use of transient data in a batch environment is the summarization of large volumes of data. Examples include sales statistics, payroll summaries or funds transferred tables. The traditional approach to summarization, shown below, involves creating a sequential extract of the subset of records and data fields needed, followed by sorting the extract into the report order, and processing it sequentially.

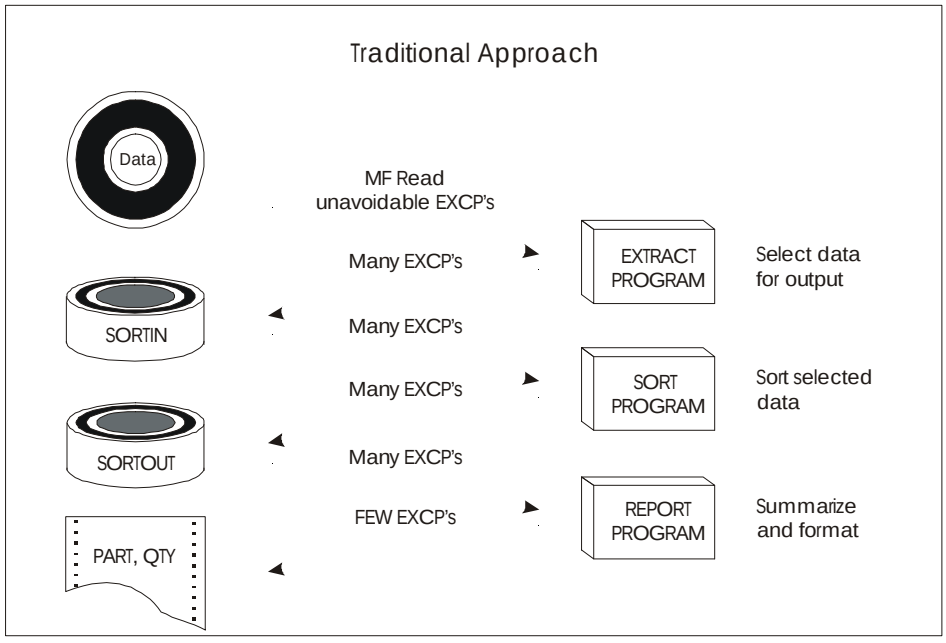

Figure 3-1: Traditional Approach to Sorting and Summarizing Data

tableBASE takes a different approach. First, you define an empty table in memory. Your huge file is reduced while being loaded into the empty table in memory. Without the need of further I/O, this consolidated data is reorganized in memory (or sorted) to suit your reporting needs. Program logic is simplified and execution performance dramatically improved due to elimination of great amounts of I/O and sorting smaller sets of data. tableBASE manages transient data totally in memory - no need for I/O or sort-in/sort-out files. This dramatic savings in EXCPs is illustrated below:

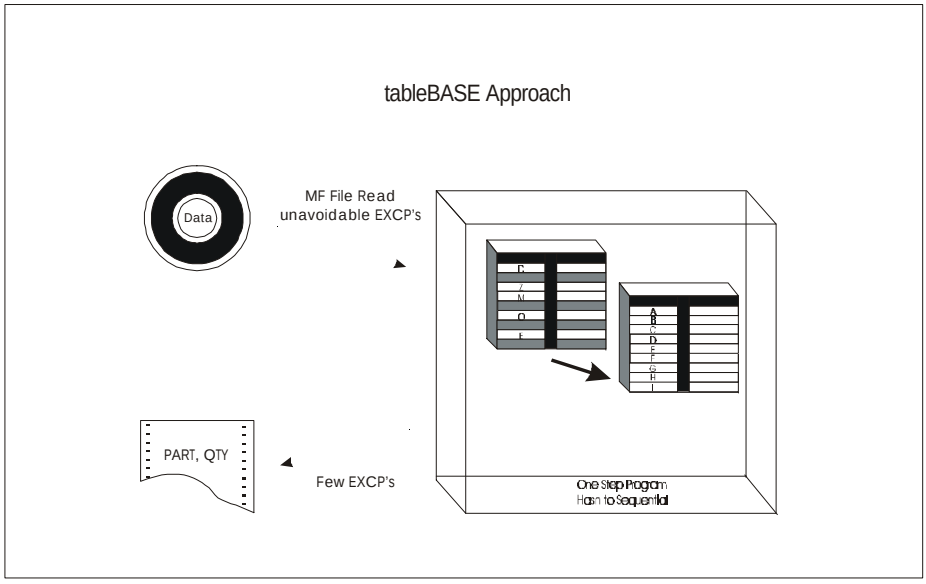

Figure 3-2: tableBASE summarizes data on first pass and then sorts data in memory

tableBASE Tables

tableBASE gives you a complete infrastructure for defining, building, maintaining, controlling and using table data. By placing into main memory tables the types of data which have been characterized above, you can improve application performance and productivity and get your table data under control.

Chapter 4
Where tableBASE Fits in Your Environment

Managing Variable Data in your Systems

Often, when considering the various elements that comprise a computer application, one element is almost never taken into consideration. This element consists of **transient data and semi-stable reference and control data, which together make up the most frequently accessed data in your systems.** We defined this data earlier as table data.

A DBMS does have the functional capability to process tables in some manner. However, processing tables is not its prime function; the tables are not normally resident in memory and, when they are, operators are limited to traditional file processing functions. The basic architecture and design of a DBMS prevent it from providing easy to use, high performance tabling algorithms and tabling techniques. As a result, performance and productivity problems arise.

Solving Performance Issues

To help boost performance, tables are often "hard coded" in programs as internal arrays or copy members, ensuring that the data is resident in memory along with the program. This addresses many performance issues but at the same time increases development requirements and maintenance problems. The resulting negative effect on programmer productivity is significant.

Solving Productivity Issues

To address the productivity issues, this data is sometimes externalized in a DBMS. While this reduces the development and maintenance problems, there is a corresponding negative effect on performance.

This type of data requires high access activity, which results in significant I/O against the DBMS or VSAM file. As a result, applications using this data experience increased response time for their on-line transactions and increased processing time for their batch programs.

A Recognized Problem

In the last few years IBM has recognized the problem of managing this type of data and offers a variety of partial solutions to try to bridge the gap. These useful new facilities include: extended addressing, ESA dataspaces and hiperspaces, Linear VSAM, Hiperbatch, CICS Data Tables, Cache and others. All have their place in particular applications, with particular restrictions, but none comes close to providing a total solution. Some are expensive, difficult to use and require implementation by systems, rather than application, programmers. Some are specific to batch or online environments only. Each one attacks a visible aspect or symptom of the overall problem, but none offers a comprehensive approach to managing all those classes of data which should be resident entirely in memory. Where the scope of these partial solutions is essentially limited to performance enhancement for traditional file processing, tableBASE goes beyond this basic vision to provide additional powerful operators which are only possible for data in memory.

The Evolution of Memory Resources

The evolution of tableBASE coincides with the evolution of memory over the past 20 years. Central memory capacity has increased by a factor of a thousand and virtual memory by a million. This dramatic increase in addressable memory offers the potential for revolutionary approaches to systems design. While other IT shops scramble to modify their programs in order to accommodate new memory technologies, tableBASE automates the management of memory resources, making it easy for you to capitalize on the evolution of memory today, and tomorrow.

Data Usage

It is generally accepted that 60% of all accesses are performed against just 1% of your data. Similarly, a small portion of your data is directly responsible for a large part of your application maintenance effort. That small percentage of data is what tableBASE has been explicitly designed to manage - more efficiently and more completely than any other product.

tableBASE is the Best Tool for Tables

tableBASE is not a replacement for a DBMS, it is a complement to your existing technologies. It was designed to deal only with memory resident tables and to help with performance and productivity issues relating to tables. Data which properly belongs in external files should be handled by a DBMS and data which belongs in memory resident tables should be handled by tableBASE.

Chapter 5 How tableBASE Improves Performance

tableBASE Improves Application Performance

In addition to its sophisticated table handling facilities, tableBASE offers memory and disk management capabilities which greatly reduce I/O operations and system overhead. Before tableBASE, table data was handled in a variety of ways as designers and programmers struggled to balance numerous factors such as processing speed, memory requirements, maintenance effort, data integrity, security and a host of technical issues. With tableBASE, tables are handled by a single solution which consistently improves performance.

- Batch systems which have I/O bottlenecks at table access points can have execution times dramatically reduced by using tableBASE.
- Interactive systems gain significant improvements in average response time.
- Online sorting and summarization is made easy and very efficient through tableBASE memory management services.

The overall decrease in machine resource demand can often save expensive equipment upgrades or additions.

What follows are the features of tableBASE that improve application performance.

Efficient Table Handling

Because tableBASE handles data management at a table level in central memory, a typical instruction path for retrieving and updating an row with tableBASE consists of 400 to 600 instructions. Contrast this with 5,000 to 20,000 instructions for a DBMS.

Access from tableBASE Libraries

When a table is needed by an application, tableBASE retrieves it from a table library and places the entire table in memory.

The table library is organized in a special tableBASE format to optimize storage capacity and access time. This storage format gives you:

- Shared library facilities.

- Fast, low-overhead table loading.
- Self-organizing libraries, eliminating the need for compression.
- Support for multiple revisions or generations of tables (up to nine generations can be kept).
- Sharing of unchanged data between multiple generations of Paged tables, eliminating data redundancy.

Table Access at Memory Speeds

Once a table is loaded, whenever the application needs to refer to an entry, it does so at memory speeds. This dramatically improves system performance over I/O based table calls in a buffered environment.

Choice of Search Strategies

tableBASE offers a variety of table organizations and search strategies to help you fine tune application performance. Depending on the characteristics of the table, its size, and its probable access order, you can optimize processing speed by selecting the right combination of organization and access strategy. With many tables, these choices are obvious. With others, you can easily experiment with the table to determine the best choices (see Chapter 7 for more details).

Shared Use

In a multi-processing environment, many tasks may need a single table concurrently. tableBASE takes care of this automatically, including resolving data security, update conflicts and queuing issues. If the table has already been loaded into a particular user region, it is re-used "as-is" with no I/O overhead. Simultaneous access to a table is an important feature in interactive systems. Multiple users would experience delayed response if additional I/O were needed for each table access.

Dynamic Reorganization

tableBASE tables can be reorganized instantly if an application needs to look at the data in a different sequence or use a different key or search strategy. tableBASE does this automatically - at memory speeds, without need for specially defined work areas **and without I/O.** A table can be reorganized, expanded or emptied dynamically or permanently defined with multiple organizations.

This feature is particularly effective when your application must consolidate detailed transaction data into a smaller number of summary rows. It is also useful when data is needed in several sequences for different reporting requirements. Your applications can have multiple views of one copy of the data.

Indexed Tables

With tableBASE, the concept of indexing differs from the traditional disk based model in a subtle, yet very profound, way. Recall that tableBASE data tables are entirely resident in memory. tableBASE indexes are also memory resident. In fact, they are built dynamically when the table is opened; they are not stored on the library with the data. And an index may be reorganized dynamically at any time, in batch or online, without incurring any I/O. When an indexed table is reorganized, only the index is affected; the data table remains in its original (random) organization. Multiple views of the data are available simultaneously through alternate indexes. tableBASE automatically sets up an index for you if requested (see Chapter 7, Pointer Tables).

These features extend indexing capabilities beyond what is practical or possible under a DBMS. On-line accesses can be processed using various ad-hoc hash indexes for high speed random access, while batch jobs refer to a sequential index for list processing. One job may need to reorganize a table dynamically for efficient summarization and reporting purposes, or to cross reference a table by keying on two or more different fields in an interleaved fashion. All these are routine functions under tableBASE.

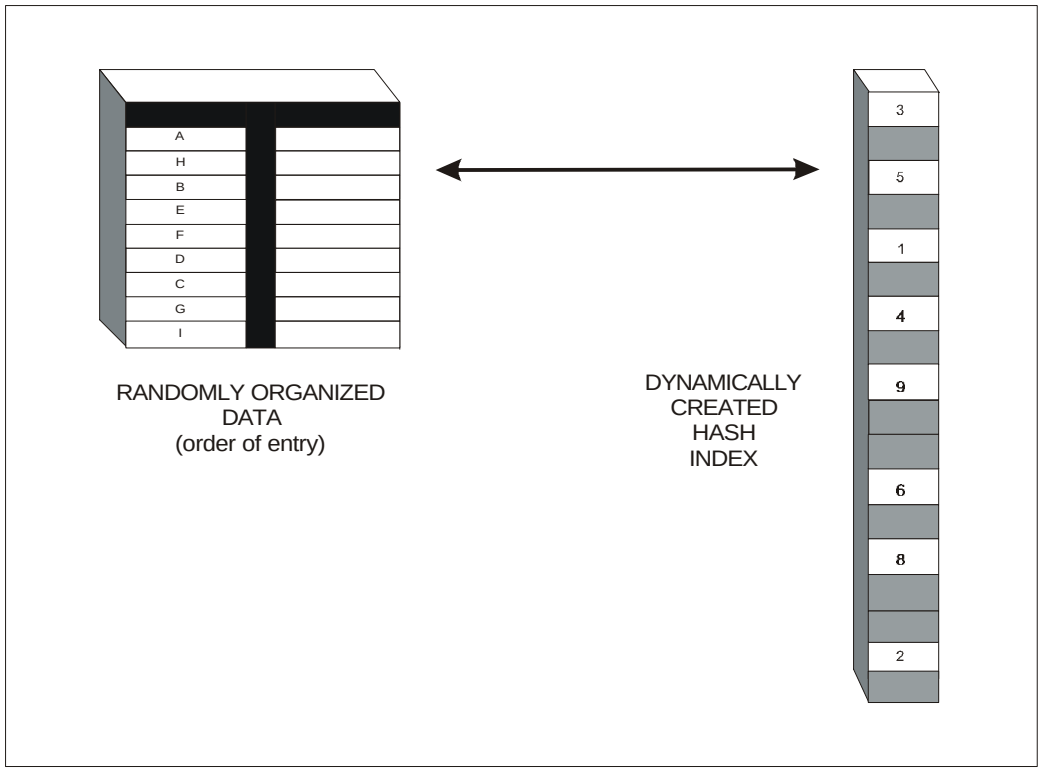

Figure 5-1: Dynamically Created Indexes Save Space and Process Faster

Effective Memory Management

tableBASE supplements the operating system's virtual storage management facility with its own, specialized table rollin/rollout mechanism. Memory management overhead is decreased since paging is pre-empted intelligently, based on patterns of actual table use, even with tables shared between separate tasks. This feature manages pool space for all tables in concurrent use.

tableBASE dynamically relocates tables in memory, as required, to provide contiguous space for new tables and prevent fragmentation. A report can be produced to provide statistics on the use of the table space area, including details for each table.

Chapter 6
How tableBASE Improves Productivity

tableBASE Improves Productivity

Before tableBASE, attempts to improve application performance often meant sacrificing productivity. Many strategies that developers employ to balance application needs with efficient use of machine resources have a corresponding negative impact on productivity. With tableBASE, in addition to increased performance, you get:

- Reduced design and programming time
- Extended options in systems design
- Reduced system complexity, shorter testing cycles
- Less programmer time spent in maintenance functions
- Increased user access and control
- Increased flexibility in adapting to change.

Before tableBASE

Let's look at some of the approaches used for managing tables before tableBASE.

Resident - Hard Coded Tables

This very common approach places tables within the application program itself as "copy library" modules or constants in working storage. This method has serious drawbacks:

1. Data tables are not directly accessible to non-programmers.
2. Any change to a table requires that programs be re-compiled, with all attendant risks and the need for thorough testing.
3. Tables are maintained in a variety of ways - often with little consistency or control. All programs using the same table have to be re-linked and re-tested to propagate the change.
4. Dynamic reorganization of table data is cumbersome and difficult. It requires coding of sophisticated routines and is often avoided in favor of less effective approaches.

5. Different programs may need different organizations of the table. Programs may diverge in their interpretation of data, and a variety of techniques must be developed for using it.

Auxiliary File Tables

Another common approach involves putting table data into simple sequential or indexed sequential files and loading them into the program when needed. This approach also has drawbacks.

1. Additional I/O logic must be added to the program.
2. The data files must be maintained. Since new tables or new rows may be added, deleted or otherwise modified, routines must be written to accommodate these changes.
3. A fixed table space and row structure place restrictions on table and program design, and further complicate the maintenance effort. Table space overflow and changes to field level structure require changes to program code.
4. Substantial I/O overhead is required to load the table each time it is required.

DBMS Based Tables

In an effort to gain some of the benefits of data base management systems, some system developers place reference table and control table data in a database to be processed in the same manner as file data. Although this solves some of the maintenance and control problems, the data must still be loaded into fixed table structures within the program. Otherwise the I/O overhead and access performance become unacceptable. Recall that this kind of data properly belongs in main memory. Despite claims of ease of use, development of tables in a DBMS environment requires careful planning by the Data Base Administrator, the Systems Developers and Programmers in order to avoid overhead problems - or misuse. And a DBMS may unnecessarily restrict control over how your data is stored, retrieved and manipulated.

With tableBASE

tableBASE does for memory resident table data what a DBMS does for external file data. It provides a complete and integrated facility to:

- Define
- Maintain
- Control, and
- Process table data.

All application programs can use the same high level programming capability to perform these table functions with simple, direct calls to tableBASE facilities. Programs are insulated from physical storage and access considerations in the same way that they are insulated from physical data structures in a DBMS setting.

Simplified Table Definition

A single command defines the table and can be invoked in the following ways:

- Interactively under tablesONLINE using our standard (or user developed) menu-driven system
- Through a Call by the application system
- Directly by the TBEXEC batch utility program or the TBDRIVER Toolset.

With tableBASE, you concentrate on the basic questions - what variables are needed for the table? - what keys and what data fields? tableBASE automates and controls all the other aspects of table management and memory management for you.

The following table attributes can be defined and changed easily with tableBASE, and without changing the logic of the application program:

- Organization
- Search method
- Indexing
- Storage method
- Passwords (read and write)
- Row size
- Key size
- Key location
- Number of generations
- Expansion factor

- Density (hashed tables)

Simplified Table Maintenance

Table maintenance is totally supported by tableBASE. No extra design or programming time is required to develop auxiliary maintenance sub-systems. You can add, change and delete table entries with a simple command. Also, changes to both table size and structure are automatically accommodated by tableBASE.

Table maintenance becomes a matter of filling in the blanks. In conjunction with development of table driven, rule based systems, this permits more direct control by end-users over their applications. Increased user responsibility and control over table data means greater control over system behavior, eliminating many of the change requests now processed by IT. Programmer time saved from program maintenance, table maintenance and the development of table maintenance software can be applied to new applications.

Your tables reside in a central library, rather than in several different locations. The end user can oversee the updating and maintenance of a single version of the table for convenience and increased control. If various organizations are needed, they are simply defined within tableBASE with no impact on maintenance or existing programs.

It is also possible to define multiple versions of a table for phased change, seasonal adjustment, regional differences or testing purposes. A table can be updated, tested and then marked for implementation at some future date. When that date arrives, the new table version will automatically be used by the application systems with no need to modify the systems themselves.

With tableBASE, you can plan and execute processing changes and variations in a more orderly fashion, using table driven controls, rather than involve your programming staff with last-minute updates to programs.

Simplified Software Development

tableBASE can eliminate weeks of design and programming which would normally go into handling:

- Table storage management
- Table loading and unloading
- Search algorithms
- Memory management

The weeks quickly turn into months if you include the considerations of multiple concurrent users. tableBASE performs all of these functions automatically; your application software concentrates on using the tables. You just set and adjust parameters which control the details and fine tune the process.

The following three tableBASE subsystems perform these table-handling functions for you.

Table Storage Management

tableBASE eliminates the development time spent on the questions of physical table storage: How should the table be stored? What access method? How should it be maintained? What about security and backup? With tableBASE all of these issues are addressed directly by standard tableBASE features.

tableBASE tables are stored in optimized form in tableBASE libraries to minimize both disk storage requirements and load time. tableBASE handles all of the physical I/O. Automatic generation control is provided for up to nine generations, with immediate access to any previous generation. Automatic table invocation based on time, date or other criterion allows development of test versions and pending versions. For multi-generation tables, tableBASE automatically optimizes generation redundancy, a particularly important feature for large, "paged" tables.

As tables are created, updated and deleted, tableBASE automatically reorganizes table library space; dataset compression is unnecessary. The tableBASE main memory environment and directory maintenance routines support all-or-nothing update failure protection. If the system or application abnormally terminates during update processing for any reason, the previous version of the table remains in effect. Only when the update is functionally and physically complete is the new generation accepted. This is important for tables which hold **complete sets** of information, where each row of the table is expected to be "in synch" with all other rows. Each row in the table shares (sometimes subtle) interdependencies with every other row. For example, in a table of currency exchange rates, all rows may be assumed to be "effective" at the same date and time. In this case, partial updates are unacceptable.

Shared Library Facilities

tableBASE allows libraries to be shared between users. As well, a single job step may access one or more libraries. With tableBASE, multiple concurrent tasks can open the same table (read only) from the same library. If a table has been opened for updating, other programs, job steps and tasks may read the table but will not be able to open it for updating. If more than one user is trying to update the same table, all but one user is locked out. The user may request that the task be "enqueued", that is, wait until the table is available, then continue with processing. tableBASE also allows different tables to be stored onto the same library concurrently by multiple tasks.

Table Memory Management

tableBASE tracks and manages table resource pool areas so you can take advantage of larger memory capacities. Without tableBASE, extensive design

and programming effort is required to optimize these resources. You may be faced with lengthy experimentation as you attempt to balance disk storage, real memory and virtual memory overhead considerations.

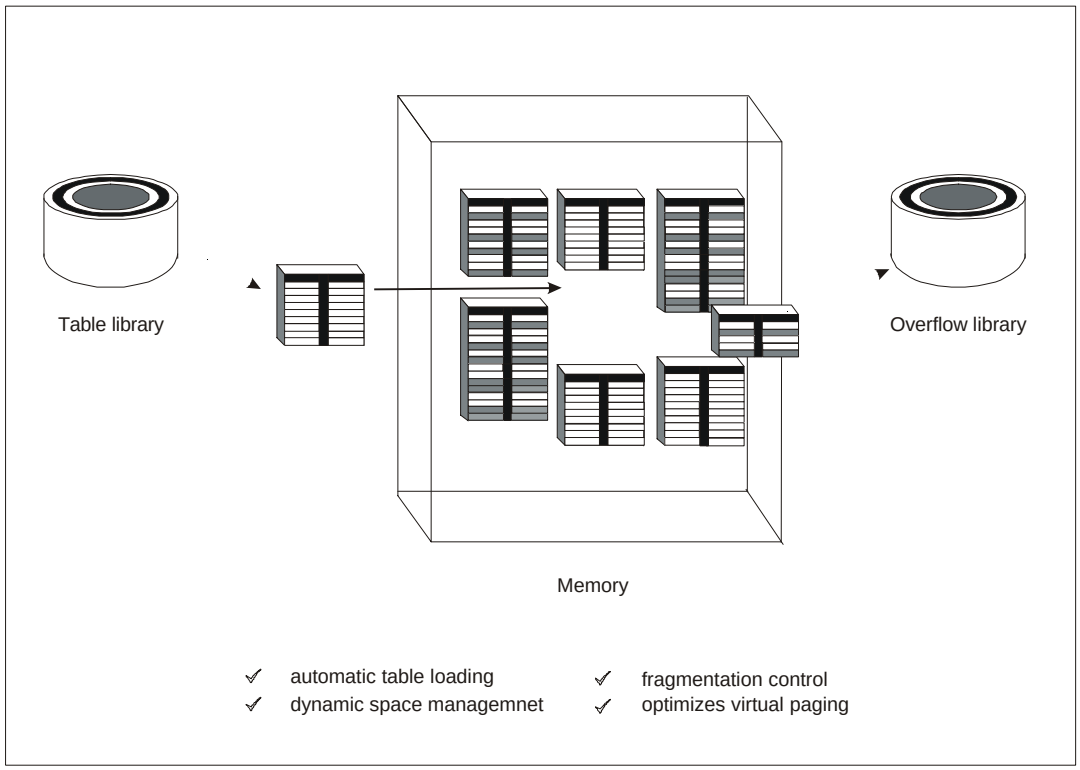

Figure 6-1: Memory Management Services

tableBASE Memory Management facilities maintain three table resource pool areas:

- Table library
- Dynamic memory
- Overflow library

tableSPACE

A function called tableSPACE can be invoked to limit and manage the total amount of space to be allocated for tables within a region. This space need only be allocated once - on the first call to tableBASE.

tableSPACE manages the allocated memory based on total table demand patterns. It uses the operating system's virtual memory management but adds an additional layer of sophisticated, highly specialized memory management and rollout/rollin strategies to further optimize performance.

tableSPACE makes table rollout decisions based on table size, structure and usage patterns to pre-empt operating system paging. Thus, table space is managed intelligently to optimum results based on the needs of applications. Furthermore, table space fragmentation is minimized through automatic, dynamic reorganization of tables to keep effective memory utilization high.

Online tableBASE applications always use tableSPACE memory management. For example, CICS users must use tableSPACE. It is generally more difficult to predict the amount of region space which should be allocated for tables in an online region than it is for batch jobs. tableSPACE places an upper limit on the amount of main storage used in these more volatile environments.

Batch applications may also use the tableSPACE services to control memory fragmentation and to limit the program's high water mark for tables.

When tableSPACE is **not** invoked, tableBASE obtains space individually for each table from pools generally available for such memory requests. Space is freed when the tables are closed.

Extended Architecture Support

Under XA, ESA and OS/390, space for tables or the TSR (tableSPACE Region) is obtained above the 16 megabyte line.

Table Usage Management

tableBASE simplifies the issues of table organization, search method, reorganization, expansion and maintenance. These issues are important concerns to the application developer. With tableBASE, programmers and designers are freed from the time consuming analysis and detail logic that the effective management of main memory tables require. Programmers are given various architectural options to choose from when creating tables with tableBASE, and they can easily fine-tune them for optimum performance.

Choice of Organization and Search Methods

With tableBASE, tables can be organized and searched in a variety of ways using several main memory organizations and search algorithms. Without tableBASE, options are much more restrictive. Different combinations of organization and search method have particular advantages. The choice of one search strategy over another can impact system performance. Manipulating data in memory is fast compared to disk; manipulating data in memory with an appropriate organization and search method, designed specifically for main memory tables, is even faster. For tableBASE tables, these decisions are made for the initial table definition, but may be revised at any time. (see Chapter 7 for more details).

Reorganization

With tableBASE, a table can easily be modified by adding either rows or columns. As well, new table organizations and search strategies can be implemented dynamically at execution time (no I/O required), or outside of the application using either batch or online utilities. It is also possible to have multiple views of a single table in use by the same program or by different programs.

When new rows are added to a table, the table size expands. tableBASE automatically manages table expansion. If during updating a table expands beyond the existing overflow area, tableBASE will automatically increase the allocated space, based on a predetermined expansion factor. This prevents a program abend from an overflow in a table. Once modified, the table can then be saved to the tableBASE library with a single command.

tableBASE gives you the ability to dynamically empty and rebuild tables in memory. This is particularly useful for summary or consolidated data. Some applications require that empty tables be defined once in memory and populated with summary data over and over again.

There is no need to create tailored routines to manipulate your data in memory. tableBASE automates the use of all your memory resources.

Table Driven, Minimal Maintenance Systems

Traditional application design and development attempts to address a complex task, often involving large numbers of possible conditions, using an inflexible program logic structure. New features are not easily incorporated into programs written in the traditional procedural style. Requirements must be clearly communicated from the user to IT maintenance personnel and control flow logic must be thoroughly understood before any changes can be made. Modification of programs usually implies re-compilation and re-linking, which takes valuable system resources away from important business tasks. Changing and re-testing programs can be a time consuming and tedious process for IT professionals.

It is generally acknowledged that the maintenance effort accounts for 70% or more of IT resources in a typical shop. According to some studies, it takes 50% of that cost just to determine what has to be changed! But, of course, program maintenance is simply a fact of life for IT. Or is it? Can we not plan for, and thus reduce or eliminate, some of that maintenance?

Table driven, rule based systems using memory resident tables are faster and easier to design, program, and maintain. Changes in program control variables become a simple matter of table updates. Program logic can be changed without physically changing programs. This eliminates recompilations, reduces system testing and helps to extend the life of your applications. Not only are your systems more flexible, they are more easily understood and maintained by other programmers. Programmers can add or delete functions quickly and efficiently.

tableBASE allows you to easily develop sophisticated table driven systems. It provides a comprehensive set of tools that automates all the low level detail of table handling without placing excessive demands on machine resources.

Using rule consolidation to reduce the following decision table for credit processing:

	DECISION RULES													
CONDITIONS	1	2	3	4	5	6	7	8						
OVER CREDIT LIMIT	N	N	N	N	Y	Y	Y	Y						
TIMES NOTIFIED <4 AND CUSTOMER HISTORY >2 YRS	N	N	Y	Y	N	N	Y	Y						
INVOICES OLDER THAN 60 DAYS	N	Y	N	Y	N	Y	N	Y						
ACTIONS														
PROCESS ORDER	X		X				X							
REFUSE CREDIT						X								
REFER TO CREDIT MANAGER		X		X	X			X						

Figure 6-2: Decision Tables Translate Input Conditions to Output Requirements

Business information systems are reflections of real world dynamic systems and, as such, they change in ways that are beyond our control. But although the low level details of future changes are unpredictable, many higher level features can be anticipated. For almost every application, certain classes of maintenance are to be expected. These include, for example:

- Educated guesses about potential future behaviour of real world systems modeled by the application
- Possible influences of outside agents, including government regulations, insurance bodies, etc.
- Corporate plans for growth and evolution

Parameters describing system attributes, which are potentially subject to change, are then placed into system control tables and reference tables. The program becomes more generic and flexible by using those values retrieved from tables, rather than values which are hard coded. In other words, the system is table driven. If and when the modeled attributes do change, system behaviour can be made to follow suit simply by updating the appropriate tables, with no change to program code.

Only tableBASE has the extensive main memory data management capabilities, beyond simple retrieval and update, to effectively support table driven or rule based programming. Database systems may provide the necessary maintenance flexibility by externalizing tables, but they cannot provide the performance and power of a main memory oriented system, where efficiency is of the highest priority.

Chapter 7
How tableBASE is Used

Using tableBASE

tableBASE is easy to use. You can access tableBASE programmatically in both batch and on-line environments or interactively at your terminal under control of CICS or ISPF. In this chapter we discuss programmatic access; Chapter 8 describes the interactive use of tableBASE.

The primary Application Programming Interface (API) for tableBASE is known as TBLBASE. This interface is available for programming languages using standard OS linkage conventions, such as COBOL, Fortran, PL/I, etc.

```
CALL 'TBLBASE' USING
                TBPARM-AREA
                COMMAND-AREA
                ROW-AREA
                KEY-AREA.

    IF FOUND='Y'
        PERFORM FOUND-ROUTINE
    ELSE
        PERFORM NOT-FOUND-ROUTINE.


COMMAND-AREA
 FK   TABLE01   FOUND   ERROR-CODE   COUNT
```

Figure 7-1: Making a Call to tableBASE

Steps in Table Access

TBLBASE

When an application requires a table row, the program makes a call to tableBASE through TBLBASE. If this is the first access to the table, space will be allocated to accommodate the table in the user region. The table is loaded into a designated area in memory called the tableSPACE region, which is managed by tableBASE services. For a retrieval request, tableBASE

searches the table according to the organization and search strategy defined for it and returns the requested row to the application program.

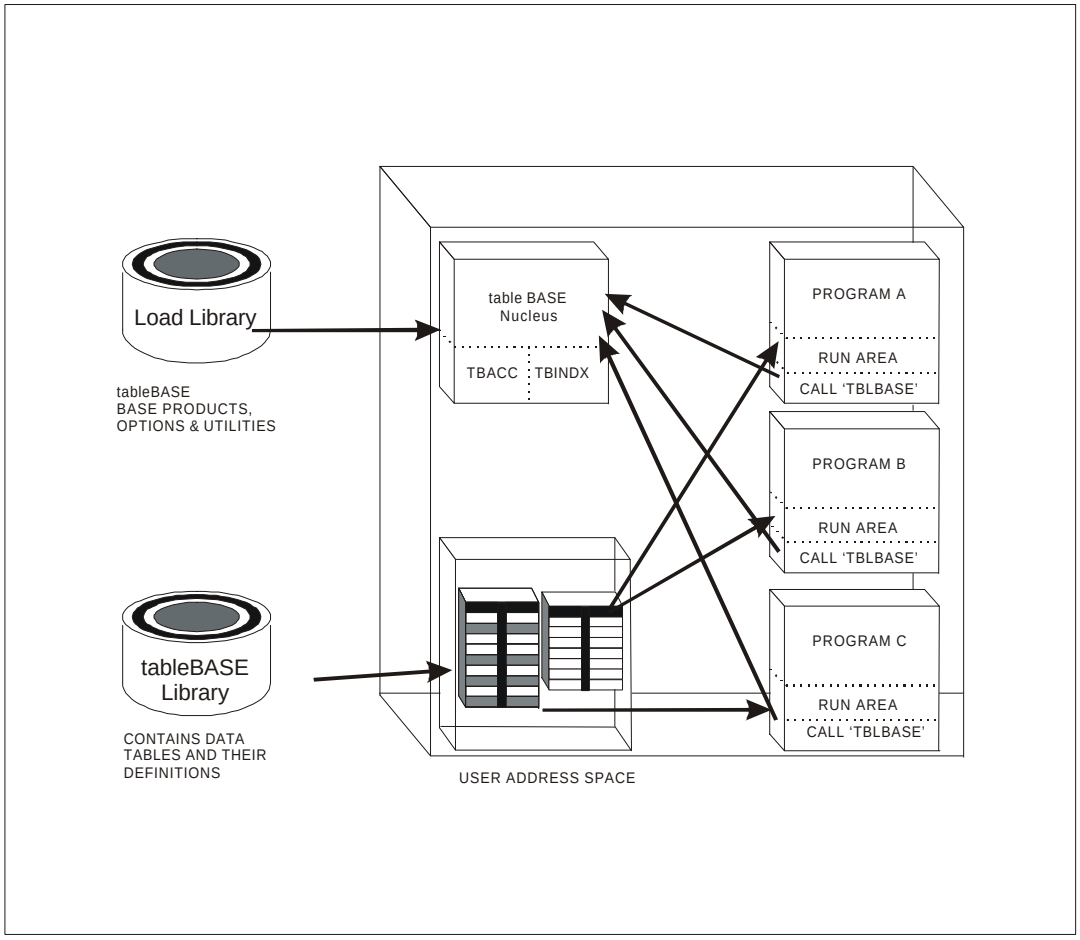

Figure 7-2: Accessing your Tables

The program needs only to provide a work space for one single row from the table. All accesses, updates, additions and deletions are done in memory. Organization and search methods can also be modified dynamically in memory. The table remains in memory until the job terminates or a command is given to CLOSE the table, which clears it from memory and releases the space it occupies. As indicated above, several programs may access one table - each with their own view of the data. Any single program may also have several views of one table; and you can define the same view for many different tables.

Four Basic tableBASE Activities

1. **Open Table** Loads the table into memory. A read only table may be modified internally but not rewritten to the library. A table opened for write may be stored back on the library.

2. **Manipulate Table** In Memory - choose from among the many table retrieval and update facilities tableBASE provides.

3. **Store Table** Writes the updated table back onto the library if it was opened for write.

4. **Close Table** Frees memory occupied by the table.

TBLBASE Commands

TBLBASE provides five command groups to perform the functions required to maintain and access tables:

1. RETRIEVAL commands provide the facility to retrieve entries from a table by key or by entry count.
2. UPDATE commands allow for modification of the contents of the table either by key or by entry count.
3. TABLE CONTROL commands provide the facility to Open, Close, Store and Define tables, as well as the ability to change the definition of individual table generations.
4. DIRECTORY MAINTENANCE commands provide the facility to alter information contained on the Directory concerning individual tables.
5. SYSTEM CONTROL commands specify the manner in which tableBASE is to handle error conditions, contention situations and the library search order.

RETRIEVAL COMMANDS	
Command	**Value**
SK	Search by Key
FK	Fetch by Key
FC	Fetch by Count
GF	Get First
GL	Get Last
GN	Get Next
GP	Get Previous
FG	Fetch Generic

UPDATING COMMANDS	
Command	**Value**
DK	Delete by Key
IK	Insert by Key
RK	Replace by Key
DC	Delete by Count
IC	Insert by Count
RC	Replace by Count
MT	Empty Table

TABLE CONTROL COMMANDS	
Command	**Value**
OR	Open Table for Read
OW	Open Table for Write
CL	Close Table
ST	Store Table
RL	Release Table (Previously opened for write)
DT	Define Table
CD	Change Table Definition
GD	Get a Table Definition
DV	Divert table to a library where it doesn't exist
DW	Divert table to a library where it does exist
CN	Change Name (in Region)
DU	Dump Table Contents
CA	Create Alternate Table Definition
IA	Invoke Alternate Table Definition

TABLE LIBRARY MAINTENANCE COMMANDS	
Command	**Value**
DG	Delete Generation
XT	Eliminate Table (on library)
CG	Change Generations to be kept
RN	Rename Table (on library)
NX	Get Next Table Name
DL	Define New tableBASE Library
LD	List Directory

SYSTEM CONTROL COMMANDS	
Command	**Value**
ML	Define tableBASE Libraries in use
LL	List Libraries in use
CS	Change tableBASE status
LS	List tableBASE status
LT	List open tables
BN	Banner Retrieval
SI	Set Indirect
DE	Disengage Library
AL	Allocate Library
UL	De-allocate Library
VS	Set Virtual System

Special Facilities for Accessing Tables

Indirect Table Access

This facility allows a table to be opened indirectly, by referencing a table name in another table, the primary table. You can access a "secondary" table via an indirect open command for a primary table. Each secondary table name is associated with some other search criteria, which is then used to identify the desired table. In this way, time or date sensitive tables can go into production according to a specified time, date or other criterion without operator intervention.

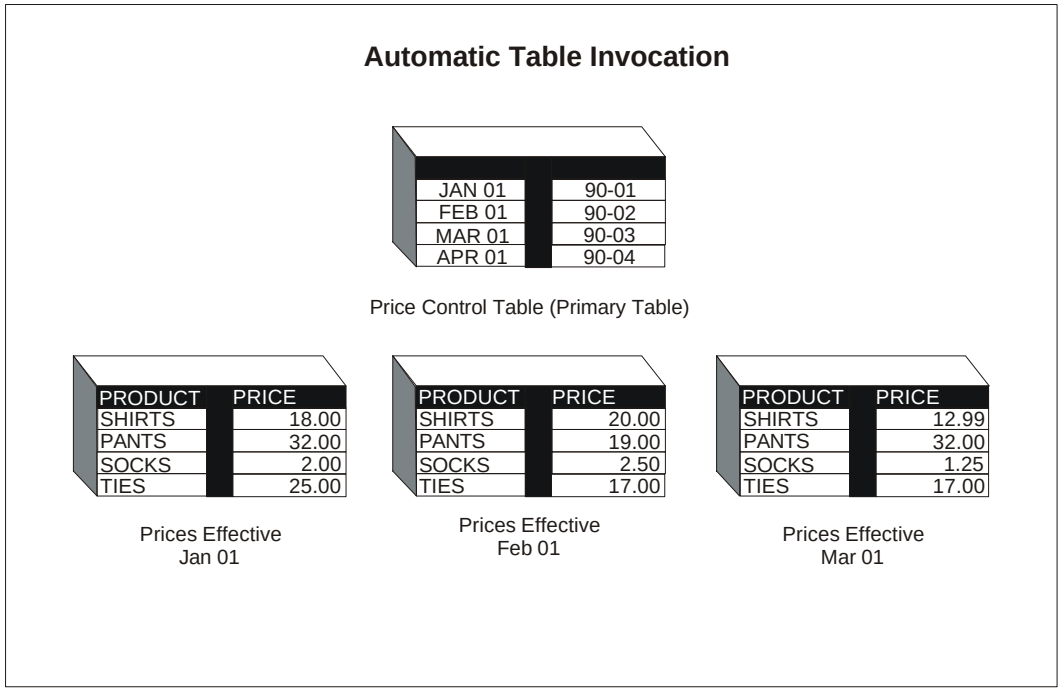

Figure 7-3: Indirect Access to Tables for Date Sensitive Data

Alternate Views
Multiple Views of Data

An alternate view permits access to the contents of a table using a different table key, organization and/or search method. This allows for dynamically re-organizing and manipulating several formats of a table without generating multiple copies of the same data.

This feature is intended for those situations where multiple views of a table are required at the same time for one or several applications sharing the same user region.

With tableBASE, you can define full featured many-to-many relationships. One view may be associated with many data tables of identical structure. Similarly, many views may be associated with a single data table in a consistent fashion.

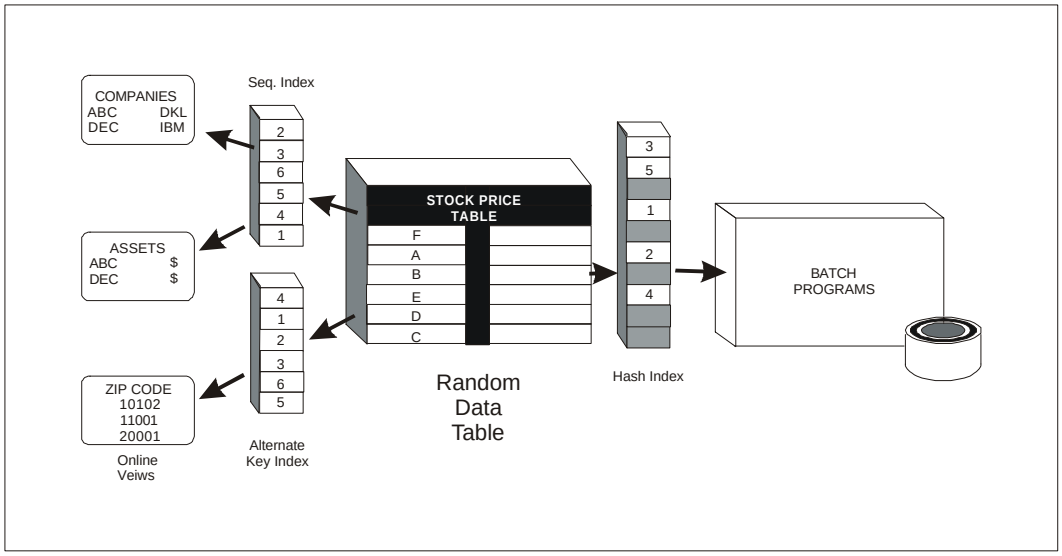

Figure 7-4: Alternate Views - Multiple Views of One Table

Table Organizations and Search Methods

tableBASE offers a choice of search methods and organizations for tables to provide the system designer and applications programmer with considerable scope and opportunity for optimization.

Table Organizations

Table Organizations describe how the data rows are physically stored in a table.

Sequential

These tables are sequenced by keys in an ascending or descending order. Sequential tables can be searched using the Queued Sequential, Binary or Address-Tree Binary search methods.

Hash

In a Hash table, the location of a data row is determined by a unique hashing algorithm. Rows are stored according to a randomized function of the key. This randomizing routine ensures that rows are spread uniformly throughout the table and not heavily clustered in any particular area.

Hash tables must be searched using a corresponding Hash search method.

Random
The order of the rows in a Random table is not considered to be important nor is it maintained in any particular sequence. All new rows inserted into a Random table are placed at the end of the table. Any deletions will cause the last row to be moved into the empty space.

A Serial search is the only practical search method for this organization.

User Controlled Sequence
Rows are stored in a "random-like" table where the sequence is under complete control of the user.

As for a Random table, the sequence of rows for a User Controlled table is not predictable from data field values, and a Serial search is the only practical search method for this organization.

Search Methods

Serial Search
A serial search proceeds by comparing the search key with the key of each row in the table. The search begins at top of the table and progresses through until the row is found or until all rows in the table have been exhausted.

A Serial search uses little overhead and is the fastest search method for small tables. It is also more efficient for User Controlled tables with a skewed frequency of hits. The user can place the most frequently hit rows at the top of the table.

The table must have a Random or User Controlled organization.

Queued Sequential Search
This search method is executed by comparing the search key to each table key progressing serially through the table until a row is found. Subsequent searches begin where the last row was compared, if the search key is deemed to be farther along in the table sequence. If the key of the next row examined is too high (or too low for Descending Sequential tables), the search resets to start from the beginning of the table.

Queued sequential searches are faster on partially or mostly sequenced data. A typical usage of Queued Sequential is for transaction matching in a master file type of update process.

The table organization must be Sequential or Descending Sequential.

Binary and Address-Tree Binary
The search key is compared with the key in the middle of the table. The search key is then determined to belong to either the top or bottom half of the table. It is then compared to the middle of the appropriate half of the table. This process of splitting the remaining table rows in half continues after each comparison until the desired row is found.

Address-Tree Binary is usually faster than straight Binary if the size of the table does not change often because the mid-point addresses are retained and

reused rather than recalculated. The mid-points are recalculated only if rows are inserted or deleted.

The table organization must be Sequential or Descending Sequential.

Hash Search
This searching technique randomizes the key to calculate an "approximate location" in the table and if some other row is found there (more than one key may randomize to the same location) a subsidiary method takes over to continue the search.

Under normal circumstances Hash searches are faster in terms of CPU time than a binary, serial and most sequential searches. It also requires fewer page accesses than any other search. Hash insertions are also extremely fast. In fact, Hash searches and inserts are often the most economical in terms of CPU time and page accesses.

Using a **Hash Pointer Table** will keep the actual data in a densely packed random table while only the sparsely packed index contains the address of the data rows. This saves on space while maintaining performance.

The table organization must be Hash.

The following table summarizes the legitimate combinations of table organizations and search methods:

	Search Methods					
	Queued Sequential	Tree Binary	Binary	Direct	Serial	Hash
Organizations						
Sequential	Y	Y	Y	Y		
Descending Sequential	Y	Y	Y	Y		
User Controlled Sequence				Y	Y	
Random				Y	Y	
Hash				Y		Y
Hash Paged				Y		Y

Types of Tables Supported

Resident Tables

Resident tables, while in use, reside in dynamically allocated storage within a user's region. Once loaded, no further I/O is required and tables are automatically extended as rows are added.

Transient Tables

Transient, or dynamic, tables are defined by the application program and are usually not retained after the program ends. One example might be summary data used in financial reports. These are memory resident tables that are automatically extended as rows are added.

Static Tables

Semi-stable or static tables are tables that are rarely updated. They have a high frequency of access compared to a low frequency of change. Some examples are insurance rate tables, ZIP code tables and process control tables.

Paged Tables

Paged tables are too large to load into available memory due to resource limitations or cost considerations and are accessed from the library one page at a time.

- Paged tables are accessed using approximately one EXCP per retrieval request and two EXCPs for updating.
- tableBASE allows for multiple generations of a paged table to be maintained without repeating the entire table.
- tableBASE can access large paged tables at least two times faster than access to similar VSAM files.

True Tables

This attribute of a table indicates that the data is physically stored in the same form as the logical representation. Inserts and deletes to the table will cause data rows to be physically moved in memory.

Pointer Tables

A pointer table includes an index which defines the order of rows in its associated data table. Insertions and deletions to the table will modify the index, with minimal movement of the actual table data.

A tableBASE Table

A tableBASE table is identified by an eight character name. The table contains both the data rows and a description, or definition, of that data. The definition includes such attributes as row length, key location and key size, organization and search method.

A tableBASE table may also have an associated View(s) as an extension of the table definition information. Views describe how the table will appear and how it will be edited in an on-line environment. Views are also used by the TBPRINT utility to describe the format of table print-outs.

Each table that is stored on the tableBASE library must have a unique name. Tables created for temporary use by an application can have almost any name. However, naming conventions for tables should be established by the tableBASE Administrator.

Building a Table

Now that you are familiar with the commands used in tableBASE, you are ready to build and manipulate tables.

Example of In-Memory Summarization

Below is a sample COBOL program that defines a new table and does an in-memory data summary. This program will extract data from a large inventory file and load it into a hash table in memory. This HASH table will be used to accumulate the total value for an unknown number of parts. At the end of the job, the totals are to be printed in sequence. (Note: Once a table row is found, the table need not be searched again, it can be referenced directly with any of the COUNT commands, e.g., RC - Replace by Count.) At the end of the job, we will be sorting the table by changing the definition of the HASH table to a sequential table (dynamically in memory) and printing the part number and total for each part.

```
DATA DIVISION
WORKING STORAGE SECTION
01  DATA-ROW.
        05      ........
        05      PARTNO........
        05      ........
        05      PART-VALUE........
        05      ........

01  TABLE-ROW.
        05      TABLE-PARTNO-KEY........
        05      TABLE-ACCUM-VALUE........

01  COMMAND-AREA.
        05      COMMAND                 PIC XX.
        05      TABLE-NAME              PIC X(8)        VALUE 'TABLE1'.
        05      FOUND-CODE              PIC X.
        05      FILLER                  PIC X(3)        VALUE LOW-VALUES.
        05      ERROR-CODE              PIC S9(4)       COMP.
        05      COUNT                   PIC S9(8)       COMP.

01  TBL-DEFINE-BLOCK.
        05      TBL-ORG                 PIC X           VALUE 'H'.
        05      TBL-METHOD              PIC X           VALUE 'H'.
        05      TBL-INDEX               PIC X           VALUE 'T'.
        05      TBL-SMC                 PIC X           VALUE SPACES.
        05      TBL-RPSWD               PIC X(8)        VALUE SPACES.
        05      TBL-WPSWD               PIC X(8)        VALUE SPACES.
        05      TBL-SIZE                PIC S9(8)       COMP VALUE +20.
        05      TBL-KEYSIZE             PIC S9(8)       COMP VALUE +11.
        05      TBL-KEYLOC              PIC S9(8)       COMP VALUE +1.
        05      TBL-ROWS                PIC S9(8)       COMP VALUE +500.
        05      TBL-GENS                PIC S9(4)       COMP VALUE +3.
        05      TBL-EXP-FACT            PIC S9(4)       COMP VALUE +200.
        05      TBL-LOW-DEN             PIC S9(4)       COMP VALUE +550.
        05      TBL-HIGH-DEN            PIC S9(4)       COMP VALUE +850.
        05      FILLER                  PIC X(8)        VALUE LOW-VALUES.
        05      FILLER                  PIC X(6)        VALUE LOW-VALUES.
        05      TBL-DATE-TIME           PIC X(12).
        05      TBL-GEN-NO              PIC S9(4)       COMP.
        05      FILLER                  PIC X(192)      VALUE LOW-VALUES.

PROCEDURE DIVISION

HOUSE-KEEPING.
```

```
        MOVE 'DT'                       TO COMMAND.
        CALL 'TBLBASE' USING            COMMAND-AREA TBL-DEFINE-BLOCK.

LOOP.
        ........
        ........ (obtain part value to be added to table)
        ........ (when finished go to PRINT-TABLE)
        MOVE PARTNO TO                  TABLE-PARTNO-KEY.

*       SEARCH THE TABLE FOR MATCHING PARTNO.

        MOVE 'FK'                                   TO COMMAND.
        CALL 'TBLBASE' USING                        COMMAND-AREA TABLE-ROW.
        IF FOUND-CODE = 'Y'

*               MATCHING PARTNO FOUND
*               UPDATE AND REPLACE TABLE ROW
                ADD PART-VALUE                      TO TABLE-ACCUM-VALUE
                MOVE 'RC'                           TO COMMAND

*       THE ABOVE "FK" OPERATION HAS LOCATED (RETURNED A COUNT
*       VALUE) WHERE OUR NEW ROW SHOULD BE INSERTED IN
*       THE TABLE

        ELSE

*       MATCHING PARTNO NOT FOUND
*       INSERT NEW TABLE ROW
        MOVE PART-VALUE                             TO TABLE-ACCUM-VALUE
        MOVE 'IC'                                   TO COMMAND.

        CALL 'TBLBASE' USING                        COMMAND-AREA TABLE-ROW.
        GO TO LOOP.

*       RESEQUENCE THE TABLE FOR PRINTING.

PRINT-TABLE.
        MOVE 'S'                                    TO      TBL-ORG.
        MOVE 'Q'                                    TO      TBL-METHOD.
        MOVE 'CD'                                   TO      COMMAND.
        CALL 'TBLBASE' USING                        COMMAND-AREA TBL-DEFINE-BLOCK.
        MOVE ZERO                                   TO      COUNT.
        MOVE 'GN'                                   TO      COMMAND.

PRINT-1.
        CALL 'TBLBASE'  USING                       COMMAND-AREA TABLE-ROW.
        IF      FOUND-CODE      = 'N'
        GO TO THE-END.
        DISPLAY         'PART NUMBER'   TABLE-PARTNO-KEY        'QUANTITY-'
                        TABLE-ACCUM-VALUE.
        GO      TO      PRINT-1.

THE-END.
        MOVE 'CL'                                           TO      COMMAND.
        CALL 'TBLBASE'  USING           COMMAND-AREA.

        STOP RUN.
```

Accessing and Maintaining Tables in Batch

There are several batch utilities provided with tableBASE to assist you in easily and quickly managing your tables.

TBEXEC

TBEXEC is a tableBASE Utility Program that is distributed with the base product and operates in Batch and TSO environments. It allows tableBASE users to perform a variety of activities against tables and table libraries without having to write application programs. For instance, through TBEXEC, you can create and initialize libraries, copy, print, update, delete and move tables from test to production environments.

TBPRINT

TBPRINT is a tablesONLINE reporting utility program that also operates in a Batch or TSO environment. It allows online users to produce reports based on field level information generated through tablesONLINE. This utility helps end users to easily create meaningful, customized reports.

TBDEFPRT

TBDEFPRT allows users to selectively PRINT table definitions and Views.

TBCOBFD

TBCOBFD generates copy books from your table definitions for use in your COBOL programs.

Chapter 8 Using tablesONLINE

Accessing and Maintaining tablesONLINE

tablesONLINE is a flexible, interactive front-end for tableBASE - and more. “Out of the box”, it provides speed and convenience for end users wanting to create, update, manipulate, test and process data tables. Because tablesONLINE is itself a table-driven system, you can readily tailor it to build a wide variety of applications. Thus, tablesONLINE helps organizations

- control program complexity,
- reduce development and maintenance workloads and, ultimately,
- improve the bottom line.

Currently, there are versions of tablesONLINE available for the TSO/ISPF and CICS environments.

tablesONLINE helps everyone involved with Information Technology:

Database Administrators	-	centralized control of table updating
	-	interface with security systems
Analysts	-	online table design and definition
Programmers	-	online table maintenance and testing
	-	interactive access to tableBASE calls
End users	-	online table maintenance without DP involvement

tablesONLINE handles data entry and table editing tasks and provides the access controls and data validation services essential to such tasks. Two of its components, the menu system and the table editor, serve as a framework for building applications.

The Menu System

The menu system manages the control flow of the various programs which make up tablesONLINE itself. To define tables, you simply make selections from the menu and fill in the blanks. The menu system can also

- Call any non-tableBASE program,

- Initiate any transaction and
- Manage complex sequences of these actions.

In fact, it provides a convenient framework for managing any system which can be described in terms of sequences of independent actions.

The Table Editor

The table editor provides a framework for editing and displaying data. It includes an extensive set of built-in edit and display features and also offers an exit program capability to enable you to exercise your imagination to the fullest in solving problems relative to the editing and displaying of data.

Using tablesONLINE

When logging on to tablesONLINE, you will be presented with one of the following screens:

- Administrator's Menu
- Applications Developer's Menu
- End-User's Menu

The menu that appears will depend on your level of authority. This is established by the systems administrator when the product is installed. An example of the Administrator's menu is shown below.

```
tablesONLINE 5.1.0 Administrator ------ ------------ Administrator ------------
 COMMAND ===>


 To select, enter number/symbol on command line:

 D    DEVELOP APPLICATION  - Create and Develop Table Applications
 M    EDIT MRO TRAN IDS    - Add/Change Sets of tablesONLINE Transaction IDs
 T    TRANSFER TO USER     - Transfer to User Application Menus
 1    EDIT APPL. CONTROL   - Add/Change/Delete Applications and/or Users
 2    DELETE SESSIONS      - Delete Active tablesONLINE Sessions
 3    COPY APPLICATION     - Copy an Application's Controlling Tables
 4    EDIT USER PROFILES   - Add/Change/Delete Items on User Profile Table
 5    EDIT HELP TABLES     - Edit tablesONLINE Help Tables
 6    EDIT TUTORIAL TABLE  - Add/Change/Delete tablesONLINE Tutorial Table
 7    EDIT X-AUTHORIZATION - Add/Change/Delete Cross Authorizations for Users
 8    EDIT USER APPL TABLE - Edit the User/Application Relationship Table
 9    EDIT CONSTANTS       - Add/Change Sets of tablesONLINE System Constants

 Enter HELP at any stage for help within tablesONLINE.
 Enter PF for program function key assignments.
 Enter X to suspend tablesONLINE and return to CICS.
```

Figure 8.1: Administrator's Menu

The System Administrator

If you are the tablesONLINE System Administrator, the Administrator's menu will be displayed. The high level of authority associated with the screen above gives you complete control over who uses the tablesONLINE environment and how they use it. You also have the option of developing online applications. Option 'D' on your menu (shown above) brings up the Application Developer's Menu (shown below) where you can easily build the framework for a new application. Screens are defined for end-user applications simply by editing tableBASE tables. Once a user application is completely defined to the system, you can make it available to your end-users by simply updating the appropriate security tables.

```
tablesONLINE 5.1.0 Administrator ------ --- Application Developer's Menu ------
 COMMAND ===>


 To select, enter number/symbol on command line:

 A    EDIT TABLE           - Add/Change/Delete Rows in a Table
 B    BROWSE TABLE         - Display Contents of a Table
 C    TBDRIVC              - Execute TBLBASE Commands
 D    DEFINE TABLE         - Define Table, View and Data Descriptions
 U    UTILITIES            - Copy/Rename/Delete a Table
 1    EDIT MENU TABLE      - Add/Change/Delete Application Menu Items
 2    EDIT PFKS TABLE      - Add/Change/Delete Application PF Keys
 3    EDIT CMDS TABLE      - Add/Change/Delete Application Alias Commands
 4    EDIT HELP TABLE      - Add/Change/Delete Application Help Items
 5    EDIT MSGS TABLE      - Add/Change/Delete Application Messages
 6    EDIT DESC TABLE      - Add/Change/Delete Application Screen Descriptions
 7    EDIT LIBR TABLE      - Add/Change/Delete Application Library Names

 Enter HELP at any stage for help within tablesONLINE.
 Enter PF for program function key assignments.
 Enter X to suspend tablesONLINE and return to CICS.
```

Figure 8.2: Application Developer's Menu

Application Developers

If you are an Application Developer, you will initially be presented with the menu shown above. This menu allows you to build screens for the end-user without the need for detailed programming or knowledge of the system's I/O protocol. To do this, you simply edit tableBASE tables.

Options 1-7 provide all the features necessary to set up the environment for your end-users. For instance, you define the help tables, PF keys, library access authority, even the commands available to your end-users, simply by filling in the blanks on the screens that appear for each option on the main menu.

Building a Table

To define a table you would choose 'D' off the Application Developer's Menu. The screen below would appear:

```
tablesONLINE 5.1.0 Administrator ------ ----- Define Table and View -----------
 COMMAND ===>


 To select, enter number/symbol on command line:

 D    DEFINE ALL           - Define All Table Elements (View and Data)
 1    DEFINE VIEW          - Define the Fields in a Table's View
 2    DEFINE VIEW SUPPLMT  - Define Supplementary Information for View
 3    DEFINE DATA TABLE    - Define Data Table Definition (DT Block)
 4    EDIT DISPLAY ORDER   - Edit the Order of Fields in View for Display
 5    BROWSE VIEW          - Browse Field Descriptions in View
 6    CREATE ALTERNATE     - Create/Edit Alternate Index for Data Table
 7    RESTRUCTURE TABLE    - Restructure Data Table (After Updating View)
 8    GENERATE COPYBOOK    - Submit a Batch Job to Create a View Copybook
 9    DEFINE M2M           - Assign Object Names to View and Data Combinations



 Enter HELP at any stage for help within tablesONLINE.
 Enter PF for program function key assignments.
 Enter X to suspend tablesONLINE and return to CICS.
```

Figure 8.3: Table Definition Menu

To begin building your table at this point, option 'D' is equivalent to options 1, 2 and 3, chained together. These three options prompt for the necessary parameters to completely define a table. You actually save key strokes by executing these options in order - since the system will calculate some of the required values for you from previously specified information. For example, row size can be calculated from the sum of field sizes.

Option 4 allows you to change the display order of your data. **Option 5** allows you to browse the different table definitions you have just created. **Option 6** allows you to create alternate definitions of the same table to give you multiple views of one table. **Option 7** enables you to restructure a data table. **Option 8** will generate a COBOL copybook. **Option 9** provides access to the M2M table where you can define the many-to-many table relationships.

Utilities

tablesONLINE also provides all the necessary utilities to help you manage your tables. Below is the menu that appears if the 'U' option is chosen from the Application Developer's Menu.

```
tablesONLINE 5.1.0 Administrator ------ Utility Menu -------------------------
 COMMAND ===>


 To select, enter number/symbol on command line:

 1    COPY TABLE           - Copy a Data Table (To Another Library - Optional)
 2    COPY VIEW            - Copy a View (To Another Library - Optional)
 3    DELETE TABLE         - Delete a Generation of a Table
 4    DELETE VIEW          - Delete a View
 5    RENAME TABLE         - Change the Data Table Name
 6    RENAME VIEW          - Change the View Name
 7    CHANGE PASSWORDS     - Change Either the Read or Write Password
 8    WRITE PROTECT VIEW   - Place a Write Password on a View
 9    EDIT PROFILE         - Edit User Profile




 Enter HELP at any stage for help within tablesONLINE.
 Enter PF for program function key assignments.
 Enter X to suspend tablesONLINE and return to CICS.
```

Figure 8.4: Utilities Menu

As with other menus, choosing options from the Utilities Menu will display screens which prompt for additional information.

The End User's Menu

If you are an end user, you will sign on directly to a particular application screen, perhaps similar to the type of menu shown below. The functions available on end-user menus are determined by the Application Developer (Option 1 on the Application Developer Menu). Screens for end users can vary from user to user and for groups of users. This helps simplify end-user tasks and assists with security.

```
 Personnel Administration System ------- Sally Jones' Work List ----------------
 COMMAND ===>

 To select enter number/symbol on command line:

 1    BROWSE PAY CODES     - Browse the pay codes in the Environment table
 2    EDIT ADDRESS TABLE   - Edit the Employee Address Table
 3    PAYROLL CODE         - Maintain Payroll Code/Name Table
 4    WEEKLY               - Weekly Job Number Update
 5    MONTHLTY             - Monthly Job Number Update
 6    INS RATES            - Update Insurance Rates
 8    PAYROLL RUN OPTIONS  - Set Payroll Run Options









 Enter HELP at any stage for help within tablesONLINE.
 Enter PF for Program Function Key assignments.
 Enter X to suspend tablesONLINE and return to CICS.
```

Figure 8.5: End User's Menu

Since you can add or delete functions from an End-User Menu, you can hand over additional functions to the user as they become more experienced.

tablesONLINE as a tableBASE Front End

The tablesONLINE Menu System

The tablesONLINE menu system (shown earlier) provides a user interface for tablesONLINE's own functions and for any other processes that are conveniently controlled from a menu-driven interface. It provides a general framework for managing any application which can be adequately controlled by choices among sequences of independent actions.

Making a selection off a menu can initiate any of the following:

- Display another subordinate menu and await another selection
- Start the tablesONLINE table editor with editor options preset (perhaps in read-only mode) and optionally with table and library set
- Transfer control to any CICS transaction
- Load and run any CICS program
- Execute a sequence of the above four actions or terminate tablesONLINE.

There is considerable flexibility available to the developer for building a system that the users need. Each user may have a tailored starting menu, or groups of users may all share the same starting menu.

It is possible to offer some users no menu choices, going directly to the table editor from tablesONLINE initialization.

For other users, a single menu may be all that is required. For example, a personnel clerk might choose among:

- Edit Employee Table
- Edit Classification/Pay Rate Tables
- Run Payroll Process
- Print reports

A Front End for Table-Driven Systems

tablesONLINE becomes a powerful **system maintenance tool** as well as a data manipulator for table-driven applications. Table-driven software, like tablesONLINE, can be reconfigured without recompilation simply by editing the tables which control its behaviour. This significantly **reduces the maintenance effort**.

Furthermore, **users can reconfigure their own software** rather than having to call on the IT department. This **improves service** to users and **reduces the IT workload**.

tablesONLINE is itself a table-driven system and can be extensively reconfigured simply by editing the tables which control it. The reconfiguration options available include:

- Developing application-specific menus and/or data entry screens
- Calling user exit programs for data validation functions beyond those built into tablesONLINE
- Calling other routines to perform security checks, calculations or whatever operations are appropriate to your applications
- Controlling access to programs (tablesONLINE programs or others)
- Controlling access to data (in tableBASE tables or elsewhere).

Data Entry

tablesONLINE may be used as a data entry system or, more generally, as a table editor for data used by other applications. tablesONLINE provides **full data validation** capabilities based on Views which describe the data expected. These Views are created by the application developer.

There are several straightforward built-in field types. For each of these, tablesONLINE provides:

- **Automatic validation** whenever the field is edited
- **Automatic translation** between the stored form of the data and the display representation
- Additional services such as:
  - **labelling data** with the field name from the View and,
  - applying **user-specified display attributes** to data whenever it is displayed.

Data Validation

A comprehensive list of data validation capabilities -- checking field values against View data definitions -- is built into tablesONLINE:

- Display masks
- Edit patterns
- Existence checks
- Exclusion checks
- Range checks
- Automated data importation from other tables

- Row creation/update date

The majority of validation requirements can be addressed by these built-in features, but for those important special cases, **exit programs can provide additional customer tailored validation.**

Extensions to Editing via User Exits

In addition to these built-in features, tablesONLINE provides a flexible interface to user-written exit programs which can perform additional validation and/or data transformation tasks. Calls to such programs are automatically made whenever a user performs specific actions in the tablesONLINE editor. Exits can perform checks or coordinate updates on multiple rows in a table. For instance, creating a row for a new employee may imply changes to the row for that person's supervisor as well as new rows being inserted into other tables.

Exit programs can also be used to extend tablesONLINE to deal with any data which fits logically into a tabular format, for example, data in VSAM files or database records. With such extensions, tablesONLINE becomes a powerful data entry and data maintenance tool, even for applications which do not directly use tableBASE.

Such exit programs allow the developer to **make independent choices of data representation** for display and storage. This can lead to large savings in DP resources, both machine power and staff time, without sacrificing user convenience.

Other Special Features of tablesONLINE

tablesONLINE incorporates many features to make it convenient for you to work with tables:

- **Cursor Sensitive Help** Which can be customized for local usage.
- **Scrollable Menus and Tables** Scroll up, down, left or right.
- **Freeze Keys** Allows users to fix key fields on the screen while scrolling up,down, left or right on other fields of data.
- **Extensive Data Conversion and Validation Routines**
- **COBOL code generator** Generates COBOL field definitions from table descriptions.
- **Customizable PF keys**
- **Windowing** Allows multiple sessions at one time.

Chapter 9
System Specifications

Operating Environments

Hardware Required:

- IBM S/370 Mainframe or Compatible

tableBASE currently operates in:

- MVS/SP
- MVS/XA
- MVS/ESA
- OS/390
- VSE

There are environmental interfaces for:

- TSO/ISPF
- CICS
- IMS/DC

tableBASE is accessible from all common languages using standard IBM protocol, including:

- COBOL
- COBOL II
- C
- PL/1
- FORTRAN
- ASSEMBLER
- NATURAL

Memory Requirements

- tableBASE: 100K
- tablesONLINE/ISPF: 250K
- tablesONLINE/CICS: 250K
- INTERFACES: 200K each

Plus enough memory (real or virtual) for the tables themselves

Library Requirements

The space required for a tableBASE Library is approximately the sum of the following:

- Ten (10) blocks of 3120 (2041 under DOS) bytes for library overhead.
- Enough space to hold the contents of all the tables in the library.
- 10% of the space required for each table's contents, for table overhead.

Security

tableBASE tables are individually protected with passwords. Read and write passwords may also be different to protect tables from accidental damage.

A Master Password facility is provided that will permit access to a table even if the original password is lost.

Since you are able to keep up to nine generations of each table, users can recover from a bad update by replacing invalid generations with older generations of the table.

tablesONLINE also provides several user exits at the Table, Row and Field levels (see section on tablesONLINE Table Editor). You can easily control what tables, fields and rows each user has access to.

In addition, tableBASE interfaces to all popular security packages such as: RACF, ACF-2, TOP SECRET, etc.

Installation of tableBASE

tableBASE is a simple product to work with and installs easily. It normally requires little or no re-linking of modules, depending on your installation requirements. tableBASE does not modify the operating system in any way.

Chapter 10 Support

Customer Support

Hotline

tableBASE customers obtain support by calling the 24-hour Hotline Support number (613) 523-5588, by facsimile to (613) 523-5533 or by e-mail to support@dkl.com. Our customers are given immediate response within business hours Monday-Friday, 8:00 a.m. to 5:00 p.m. EST, and 24-hour guaranteed response outside of normal business hours. (Normal response time outside of business hours is one hour.)

Internet

You will find a wealth of information at our web site (www.dkl.com) any day of the year. Here, a FAQ (Frequently Asked Question) list is maintained for your use. You can download the PTF(s) you need to remedy a problem. You may wish to subscribe to the tableBASE Forum, a moderated mailing list. Or, you may find a useful tip in our "Tip of the Day".

Consulting Services

Data Kinetics also provides on site consulting for clients who have special development needs. Our highly skilled specialists are available to help you apply memory based technologies toward minimal maintenance high performance systems. Consultants can be on-site within 24 hours in emergency situations.

tableBASE Training

tableBASE training is available in two formats - intensive, instructor-led workshops and, coming soon, over the Internet, using the Data Kinetics virtual conference center (conference.dkl.com).

3 Day tableBASE Workshop

This workshop teaches students the various tableBASE facilities for accessing and updating tables in batch and online environments. Students gain hands-on experience with tableBASE by coding programmed solutions to workshop problems. In addition, they are encouraged to bring examples of real world problems to the class for analysis.

Some table driven programming techniques are introduced, and students learn where to use tables for improving operational efficiency of their programs and reducing maintenance requirements.

1 Day Minimal Maintenance Program Design Seminar

The objective of this seminar is to teach students how to design and code applications using table driven, "relational" strategies for programming. This is an intensive introduction, which presents several techniques for improving system flexibility and minimizing future maintenance requirements. These techniques guide programmers and analysts in identifying the factors that define a problem. In addition, they help programmers visualize the interaction of these factors to focus on the consistency and completeness of the solution.

About Data Kinetics

Data Kinetics Ltd. has been providing computer-based solutions to the IT department of Fortune 1000 organizations world-wide since 1977. Organizations seeking a competitive edge through information technology have come to rely on Data Kinetics and its expertise in adaptive technology. The company's flagship product, tableBASE, is the leading tool set for creating adaptive technology applications on the mainframe. Data Kinetics also provides consulting and training services. The company is certified to the standards of ISO 9001.

Data Kinetics Ltd.
2460 Lancaster Road
Ottawa Ontario
CANADA K1B 4S5

Telephone:	613-523-5500 800-267-0730 CAN/USA 613-523-5588 Hotline
Facsimile:	613-523-5533
Web Site:	http://www.dkl.com

A sampling of tableBASE clients from our growing customer list includes:

- ADP Business Information Services
- Alberta Public Works
- Allied Irish Banks
- American Express
- Baltimore Gas & Electric
- Bank of New York
- Bear Stearns
- Blue Cross/Blue Shield
- Boeing Corp.
- Charles Schwab & Co. Inc.
- Citibank
- Commercial Union Insurance
- Dupont
- Enron Corp.
- Fidelity Investments
- Fireman's Fund
- Frontier Corp.
- GEICO Corporation
- General Accident Insurance
- Goldman Sachs & Company
- Health Management Systems
- Highmark
- Hudson's Bay Co.
- ING Canada
- Key Services
- Litton Computer Services
- Los Angeles Times
- Mattel
- Mazda Canada
- MBNA
- Northwestern Mutual Life
- Office Depot
- Oslo Kommune
- Pennzoil Co.
- Progressive Insurance
- Prudential Assurance
- ReliaStar
- Rogers Cable Systems
- Sears Roebuck and Co.
- Smith Barney Shearson
- State of Georgia, Department of Administrative Services
- Travelers Insurance
- United Dominion Trust
- Visa International