tableBASE

Concepts and Facilities Guide

Release 6.1.1

Document Number: TBM009-R611v1.1

Publication Date: tableBASE Release 6.1.1 Version 1.0 - September 2012

tableBASE Release 6.1.1 Version 1.1 - April 2015

DataKinetics Technical Support: 1-613-523-5588
Email: tableBASE@dkl.com

DataKinetics Ltd.
240 - 50 Hines Road
Ottawa, ON
Canada K2K 2M5
http://www.dkl.com

Telephone: (613) 523-5500
1-800-267-0730 (toll free in the US and Canada)

Facsimile: (613) 523-5533

Table of Contents

Preface

This guide provides an overview of the concepts and facilities of tableBASE.

Audience for this Guide

This guide is for anyone new to tableBASE. It describes the fundamentals of the tableBASE architecture.

What is Covered in this Guide

This guide identifies processing features of the tableBASE core product, along with the optional DataKinetics VTS and tablesONLINE products. Basic tableBASE commands are introduced and concepts such as table search methods and organizations, indexing, and table protection are explained. Basic table activities—for example, opening and closing a table—are described, with code examples. tableBASE training courses and other available support options are outlined in the last chapter.

Naming Protocol

Version 6 features the new tableBASE naming protocol. All tableBASE executables begin with DK1 for easy identification, a prefix that has been reserved with IBM for exclusive use by DataKinetics Ltd.

Aliases are retained so that no changes are required to your existing applications.

What You Should Know to Use this Guide

It is helpful to have general application development and mainframe knowledge to understand some of the table concepts and facilities described in this guide.

Glossary

The following terms may be found throughout this guide:

Data Table	A Data Table is the actual raw data. Each Data Table has a table definition (DT-BLOCK) that is used to generate the Index for the Data Table.
Index	An Index is defined for each Data Table. A Data Table Index is generated dynamically when a table is opened or defined based on the information in the table definition (DT-BLOCK).
Alternate Index	An Alternate Index is an Index that may be defined for a Data Table. The Alternate Index has an Alternate Index definition (ALT-DEFINITION) that defines the key, organization, and search order. Alternate Indexes are optional, and there is no limit to the number of Alternate Indexes a Data Table may have.
Delivered defaults	The defaults that are delivered with the product. Also known as *factory defaults*.
Installation defaults	The defaults set at installation time by an administrator, which may or may not be the same as the delivered defaults. Defined using the TBOPTGEN file. (These defaults may be overridden by an individual application using the TBOPT file.)
TSR	Table Space Region. A data space of up to 2G is used by tableBASE to house tables. The data space is owned by an application in the associated address space. The application uses tableBASE to access data within the tables.
Local TSR	As above.
Shared TSR	A VTS-TSR; see below.
Temporary Table	A temporary table exists only within a TSR, and is created by the DT command (or IA). It is never stored in a library. A temporary table can be distinguished from a library table using the GD command output—if found, a temporary table will show no dataset name.
Linked Table	A linked table (also known as a remote table) is created when a user issues a command to open a table that is already open in a VTS-TSR specified in the LIB-LIST. The table entry in the local TSR is linked to the existing open table in the VTS-TSR. No updates are allowed to a linked table.

Table Expansion	Dynamic allocation of space for tables in the TSR when the initial space allocated becomes insufficient.
Multitasking Batch	An MVS region that implements multitasking by attaching multiple TCBs. This can include a batch job that attaches several subtasks or a transaction processing region like DB2 stored procedures that implements multitasking through multiple TCBs.
View	A tablesONLINE View provides the field, edit and display attributes for a Data Table with its Index. In releases previous to Version 6 (and Version 5) the View was referred to as a Field Definition Table (FDT).
Alternate Index View	A tablesONLINE Alternate Index View is identical to a View but applies to a Data Table when access is through an Alternate Index.
VTS	Virtual Table Share. The term "VTS" may be used to refer to the DataKinetics VTS product, which permits TSRs to be shared among applications. These shared TSRs are called VTS-TSRs.
DataKinetics VTS	The product that provides the shared VTS-TSR capability.
VTS-TSR	A Virtual Table Share (VTS) Table Space Region (TSR) is a shared TSR, and resides in a shared data space. Applications can access tables within a VTS-TSR, and use the information as if it were within their local TSRs.
VTS Agent	The DK1VAGNT program, which initializes VTS-TSRs in tableBASE, and then sits idle until the VTS-TSR is to be terminated. If VTS Manager is installed, VTS-TSRs are managed by VTS Group Managers, and the VTS Agent is not required, but is still available for transition purposes.
VTS Manager	A DataKinetics product that extends the functionality of the VTS product.
TPM	An internal DataKinetics development term which refers to the VTS Manager.
VTS Group Manager	The component of the VTS Manager product which manages VTS-TSRs.
TPVM	An internal DataKinetics development term which refers to the VTS Group Manager component of the VTS Manager.
LDS	Linear Data Set

Conventions used in this guide

This guide uses conventions to differentiate code and typed commands, and to display the names of parameters:

Convention	Description
`code examples and commands`	Code examples and commands appear in this type of font: `this is an example of the font.`
MAXNMTAB	Names of parameters appear in upper case simply for ease of reading; actual case used is upper or lower or a mixture.
Version	Following IBM standards, the term *version* refers to a generation of a software product that has significant new code or new functionality. *Version* is a more general term than *release*. For example, *Version 6* includes *Release 6.1* and *Release 6.2,* and is equivalent to *Release 6.x.*
Release	Following IBM standards, the term *release* refers to a program or set of programs which represent a specific revision to the base version of a software product. For example, *Release 6.0* is a term that is used to identify the first release of *Version 6*. Subsequent releases made available under the Version 6 umbrella, such as *Release 6.1*, will provide additional revisions to the base product.
Modification Level	Following IBM standards, the term *modification level* refers to the application of specific program enhancements and error corrections to the release of a software product. For example, *Release 6.0.3* is at *modification level* 3, and *Release 6.1.0* is at *modification level* 0.
MVS	MVS is a generic term which is used when referring to z/OS and other related IBM operating systems.

What This Guide Contains

Chapter 1 offers an introduction to tableBASE and its capabilities. Each product feature is highlighted and described.

Chapter 2 looks at the concepts in tableBASE, including the architecture, tables, Views and Indexes.

Chapter 3 explores tableBASE facilities, driver programs that execute tableBASE commands, and utility programs.

Chapter 4 examines the basics of using tableBASE, from opening a table to optimizing application performance.

Chapter 5 details the system specifications needed to use the optional VTS product to enhance tableBASE.

Chapter 6 provides an overview of the menu system and table editor of the optional tablesONLINE product.

Chapter 7 details the system specifications needed to run tableBASE.

Additional tableBASE References

This guide is one of a series that describes tableBASE and tablesONLINE:

- *tableBASE Release Notes*
- *tableBASE Installation Guide*
- *tableBASE Concepts and Facilities Guide*
- *tableBASE Batch Utilities Guide*
- *tableBASE Administration Guide*
- *tableBASE Programming Guide*
- *tableBASE Quick Reference Guide*
- *tablesONLINE/CICS User's Guide*
- *tablesONLINE/ISPF User's Guide*

VTS Manager References

The following manuals describe the VTS Manager product:

- *VTS Manager Concepts and Facilities Guide*
- *VTS Manager Administration Guide*

Customer Support

Customers with maintenance agreements can obtain 24-hour hotline support through a direct line to our Technical Support staff (+1-613-523-5588).

The DataKinetics customer support Service Level Agreement lists the guaranteed response times for each problem severity level. High priority critical and serious problems receive immediate attention from our Technical Support staff.

Before calling the DataKinetics Technical Support staff, be sure to consult the "Support" section of our web site (www.dkl.com) for all the latest information on our technical support protocol. Look in "Support" for our "FAQs", which provide answers to frequently asked questions posed by our users.

The "Customer Login" link supplied on our web site navigates to a special support section which is available to customers who have maintenance agreements with DataKinetics Ltd. A username and password are needed to access this section of the web site—consult your tableBASE administrator, or DataKinetics customer support, for this information.

Training and Professional Services

DataKinetics Ltd. is proud to be able to offer our customers the very best in training and technical consulting services. See the "Services" section of our web site for all the latest information regarding our offerings in this area.

About DataKinetics

DataKinetics®, the leader in enterprise IT optimization, enables the top data centers in the world to overcome today's mainframe throughput, performance, capacity and interconnectivity challenges. In business for over 30 years, DataKinetics products are powering nearly a billion mission-critical transactions per second every day for the world's largest banking, insurance, credit card, brokerage house and retail organizations.

1 Introduction

Overview

Organizations running large-scale transaction-processing mainframe computing infrastructures are constantly having to address the ongoing issues of change, resources and performance.

- **Change**: How does an organization keep its computer applications current in a world operating on Internet time? How does it do so quickly and economically?
- **Resources**: How does an organization optimize the return on computer resources? Is it possible to get more out of the current mainframe environment?
- **Performance**: How rapidly does an organization serve customers? Are customers forced to wait while data is accessed?

tableBASE was specifically designed to help organizations solve the problems associated with these issues. It is the proven in-memory table manager for today's enterprise mainframe environment.

Mastering Change

tableBASE lets you separate business rules from processing logic. Business rules need to change fairly often to deal with the volatility of the global economy, while processing logic is much more stable. By having business rules stored in an external table, changes can be made to the rules immediately, when and where needed. There is no need to get programmers to update applications-- a process that can take days or weeks, depending upon the amount of testing and rework required.

Optimizing Resources

tableBASE itself uses a minimum of computing resources; more importantly, because of its turbocharged performance you can get more done with your current computing platform and, thus, postpone costly upgrades.

Improving Performance

tableBASE will outperform a stand-alone DBMS by an order of magnitude; typically 10 – 25 times faster, dramatically improving the performance of your transaction-processing applications.

What is tableBASE?

A full featured table manager, tableBASE is specifically designed to enable the rapid development of applications that require processing of many table lookups. It manages tables in memory and allows multiple users to concurrently read, modify and permanently store data.

tableBASE is a software tool that helps companies improve the performance of their mainframe applications by reducing the I/O usage and CPU cycle consumption inherent with database access. tableBASE optimizes your applications using indexes and access algorithms designed for in-memory use. The immediate benefit is faster running applications; a secondary, but no less valuable benefit is a reduced urgency and scale of future hardware upgrades. As mainframe upgrades become necessary due to increased demands on your data, powering your applications with tableBASE will reduce the required the scale and expense of any planned upgrade.

For just about any mainframe program, tableBASE can speed processing. For example, the traditional approach to mainframe data processing, illustrated in Figure 1-1, involves many database accesses for every transaction.

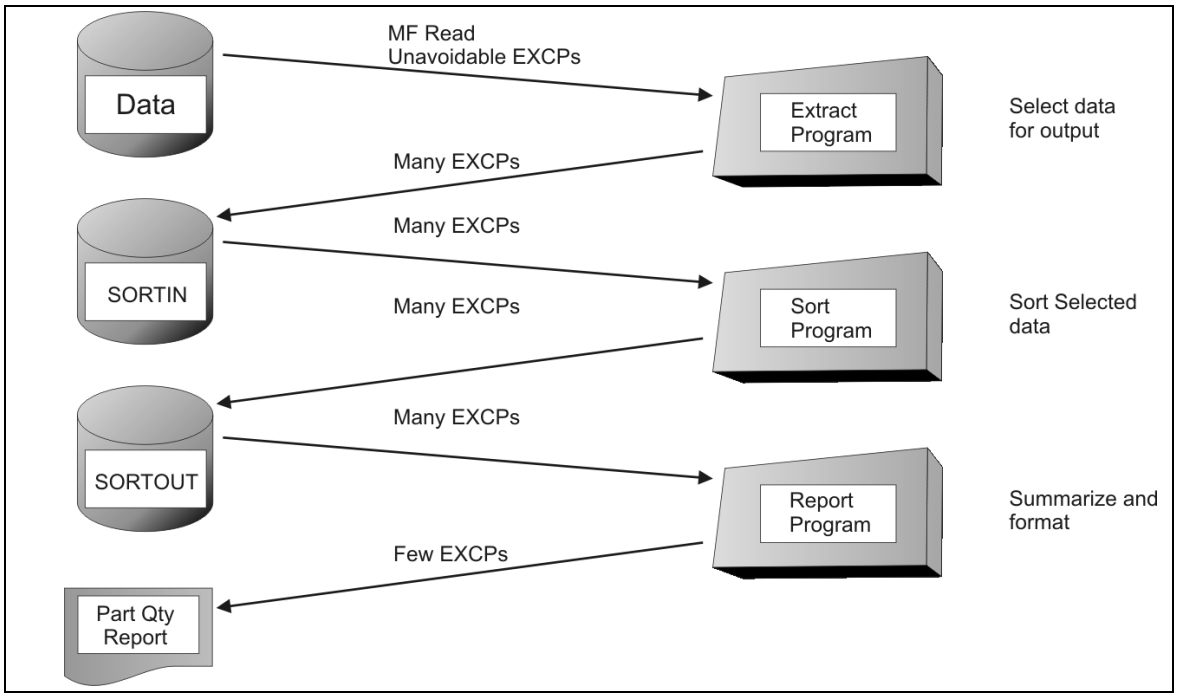

Figure 1-1: Traditional approach used for transaction processing

tableBASE takes a different approach. First, an empty table is defined in memory. The appropriate data is then extracted and loaded into the empty table in memory. Without the need of further I/O, this consolidated data is reorganized, or sorted, in memory as required for transaction processing. Program logic is simplified and execution performance dramatically improved by eliminating large amounts of I/O and sorting smaller sets of data. tableBASE manages data totally in memory, without any I/O or sort-in/sort-out files. This dramatic savings in EXCPs is illustrated in Figure 1-2.

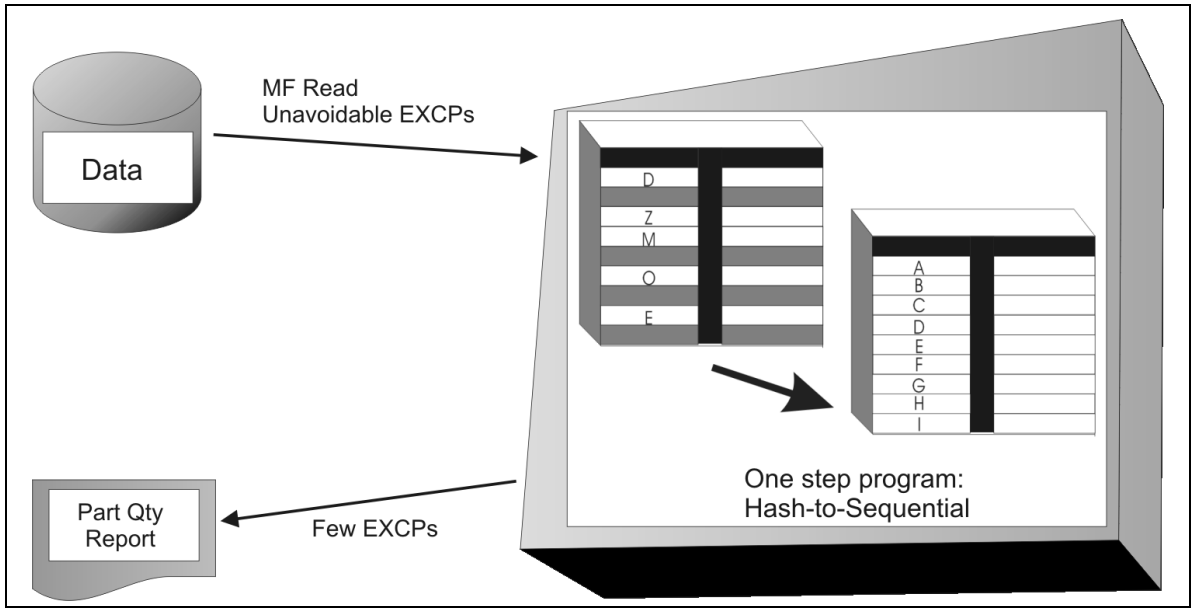

Figure 1-2: tableBASE approach used for transaction processing

tableBASE is a complete infrastructure for defining, building, maintaining, controlling and using table data. Application performance and productivity can be improved by placing tables with these types of data into memory.

Sixty percent of all accesses are performed against just one percent of data. Similarly, a small portion of data is directly responsible for a large part of the maintenance effort for an application. 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 a complement to your existing technology. It is designed to deal only with memory-resident tables and to help with performance and productivity issues relating to tables. Data that properly belongs in external files should be handled by a DBMS and data which belongs in memory-resident tables should be handled by tableBASE.

tableBASE Features

Date-Sensitive Processing

Business needs may require date-based processing. tableBASE allows you to implement business rules based on date. Within a table row, indicate the date when the row should be effective. As tableBASE cycles through the table, it will retrieve only those rows which contain a currently active date. For more information see "Date-Sensitive Processing" in the *tableBASE Programming Guide*.

Accessing a Table Indirectly

Rather than use rows in a table to execute different processing rules, some organizations use complete tables. For example, a fuel distributor might implement different pricing policies using different tables for its various classes of customers. The processing logic is the same for all customers, but the details affecting the final price are stored in different tables for each class of customer. tableBASE provides indirect table access from a row in another table.

Flexible Indexing

All tableBASE tables are indexed. When a table is created, a primary Index is defined. There may be times when a table needs to be indexed differently than the primary Index, tableBASE permits Alternate Indexes to be defined for a table such that the table organization, search method, and key are different from those used for the primary Index.

Version Control

tableBASE can maintain up to nine generations (or versions) of tables. If the latest generation of a table is incorrect, you can revert quickly to an earlier generation can be reverted to quickly. tableBASE generation control is flexible, allowing any of the last nine generations of a table to be accessed.

Multitasking

tableBASE is fully re-entrant and extends tableBASE performance benefits to multitasking applications. In a batch environment, tableBASE multitasking is provided when a primary task attaches one or more subtasks.

In a CICS TS environment, tableBASE multitasking is provided for applications executing in the CICS TS quasi re-entrant (QR) Transaction Control Block (TCB) and for applications executing in the CICS TS Open Transaction Environment (CICS TS OTE). With DB2 stored procedures, tableBASE multitasking is supported for both DB2- and Work Load Manager (WLM)-managed stored procedures.

Table Management

tableBASE complements your DB2 DBMS. Some data is best suited to standard DBMS storage and retrieval. However, data structures that are accessed over and over again perform significantly faster in memory. When there are multiple accesses per row, the table also performs best if it resides in main memory.

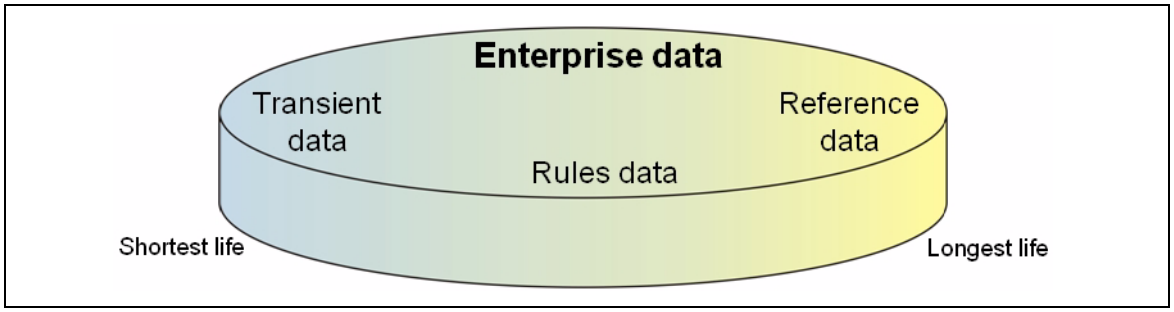

Figure 1-3: Types of data

Certain types of data often require the housekeeping overhead provided by a DBMS like DB2; however, dynamic, or transient data (left), frequently accessed, or reference data (right) and rules data are well served by the high performance main-memory processing that tableBASE provides.

While tableBASE can improve performance in virtually any system, it is best suited to specific types of data. The most impressive benefits of tableBASE are seen with the following:

Reference Data

Reference (or semi-stable) data has a high ratio of read access compared to write access. The data may even be updated every few minutes, but there is a high volume of retrievals between update cycles. This data does not normally need to be in a database since logging of individual updates is not a requirement. Semi-stable data is ideal for in-memory processing. Some examples are pay rates, inventory codes, and rate tables.

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.

In a traditional environment without tableBASE, if one of these variables changes, then extensive programming, testing and re-testing is needed . For example, if a table increases beyond the planned maximum size, or a new variable is added, or if the table must be accessed in a different order, or by a different key.

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

Constants and Parameters

Constants, literals, and other parameter values that make up a significant part of the memory of an application are another form of reference data, which should be managed as tables rather than embedded in program logic. Typically the approach has been to place these values in memory with copy members at compile time. Without using additional I/O overhead, these values are then only available for a single task. This can increase demand for dynamic storage areas, if interactive online usage requires multiple copies of these constants for each transaction generated. Maintenance of these values is time consuming, as programs must be relinked and re-tested for every change.

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

Rules Data

This is a special class of reference data that contains rules for determining which routines should be executed under a particular condition or set of conditions. Rules 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 is both difficult and expensive to maintain.

With the rules for the control of the program residing in tables and accessed by tableBASE, application maintenance is greatly simplified. Examples are report distribution lists, user and password tables, and data validation tables.

Rules data is:

- used with very high frequency and must be very efficiently integrated with program logic
- usually small
- changed periodically, requiring careful planning and control

Rules tables, because of their small size and frequent access, are often embedded within the programs using copy library statements. Programmers often use ad hoc techniques to maintain these programs. Changes require relinking and extensive retesting.

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 and controlled by user-defined parameters, such as automatic update and backup cycles, and automatic phase-in of new table versions. This can all be done by updating tables, without changing the application software. There is no need to relink and extensively retest programs.

Transient Data

Transient data is the opposite of reference data. It is volatile and forms the bulk of memory 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 involved. Transient Data Tables have these characteristics:

- data is reduced, accumulated, summarized, or reorganized from a primary data base
- the table can be dynamically reorganized to serve different objectives
- it is usually a temporary version of the original data, and can be discarded when the objectives are met, and regenerated when 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 tables of funds transferred. The traditional approach to summarization, illustrated in Figure 1-1, 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 1-2 shows how tableBASE improves the performance.

tableBASE Sophisticated Memory Management

tableBASE handles tables as data structures resident in main memory. It provides high performance and easy maintenance by using a memory management system that is optimized for table structures.

tableBASE integrates memory management capabilities that are lacking in many programming languages. It provides:

- automatic table load and unload facilities
- efficient main-memory data organization
- a variety of high-performance search methods
- dynamic runtime expansion of allocated Table Space
- dynamic runtime reorganization of tables
- high performance Index structures
- dynamic runtime Index creation and modification
- dynamic runtime Alternate Views

Most of the above are not found in traditional DBMSs or databases because those systems are primarily DASD oriented. Because tableBASE is main memory oriented, it offers facilities and features beyond that of a typical database or DBMS.

tableBASE at Work

tableBASE can help improve the data-processing applications in your organization by:

- promoting standardization and system flexibility
- minimizing program maintenance
- increasing system life and efficiency
- reducing development effort
- providing cost-effective methods for system re-engineering
- simulating and evaluating the impacts of processing decisions
- dramatically reducing physical I/O

Here are a few of the many innovative ways tableBASE is being used:

- A major credit card processing company uses tableBASE in their clearing and settlement application, where they clear and settle 100 million transactions a day. Tables, composed of business logic and validation rules, are used to guide a transaction through processing logic. This is performed at memory speed.
- A major insurance company uses tableBASE in virtually every policy management system, giving them the flexibility to modify business rules easily and respond quickly to change.
- tableBASE is embedded in the environment of a large US bank to allow them to produce an integrated monthly bank statement for their customers and to insert very targeted marketing messages on the statements themselves.
- tableBASE enables an Irish bank to respond quickly to rate changes by updating a single table entry to modify an interest or exchange rate.
- A leading US financial institution supplements DB2 with tableBASE in applications that rely heavily on looking up reference tables, using temporary tables and summarizing data. They have increased their overall application performance by a factor of 10.
- Another insurance customer uses tableBASE to edit and manage all the data being input to the corporate data warehouse. tableBASE is used to define the rules for input and output to the data warehouse.

Benefits of tableBASE

The memory-based architecture of tableBASE dramatically reduces the time to access, sort, and summarize data. Applications run significantly faster.

By increasing the speed of applications, more CPU cycles are made available and the overall system throughput is enhanced. This additional CPU time may postpone the need for mainframe upgrades or may be used to accommodate more transactions.

Additionally, the tableBASE architecture lends itself to developing rules-based table-driven applications. By placing application processing rules and decision logic in external tables, data changes are made in tables, not in code. With legacy code, this approach maximizes your investment in applications by extending their productive life. With new application development, this approach compresses the software development cycle and minimizes the testing effort. In each case, the applications designed with a rules-based, table-driven approach are more stable and easier to maintain.

tableBASE is 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 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 shared memory, it can be used by other tasks with no additional I/O overhead.

Additional Products which Enhance tableBASE Base Product

The base product is available for z/OS and consists of the tableBASE nucleus and batch interface. Additional products and interfaces are available, including:

- CICS TS and IMS TM interfaces
- tablesONLINE—a completely menu-driven online editor for tableBASE
- VTS (Virtual Table Share)—provides an environment to share tables across regions
- VTS Manager—provides a powerful management structure for VTS-TSRs

CICS TS and IMS TM Interfaces

Optional interfaces are available for CICS TS and IMS TM. Both interfaces provide the same facilities and operations as the batch 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 environment through the respective interface.

tablesONLINE

tablesONLINE is an optional product which gives developers and online users access to tableBASE services. Application developers can define, update, test, and process tables from a menu-driven interface. Screens and applications for online users can be created by filling in the fields with the desired parameters. tablesONLINE is available for CICS TS and, in a simplified version, for TSO/ISPF.

VTS (Virtual Table Share)

Optimizing memory resources can be especially challenging when there are different operating environments in a single computing infrastructure.

The optional VTS product allows access to shared, in-memory tables across multiple CICS or IMS regions, from both batch and online environments. VTS allows applications running in different operating environments, (for example batch, TSO/ISPF, CICS TS, or IMS TM), to share data for read and write access.

Loading one copy of data into shared memory reduces I/O, system paging, and overall storage requirements. VTS takes care of memory management and loads tables into memory so that applications and online users may access data from thousands of small tables, or a single table as large as 2 GB.

A VTS-TSR is a shared tableSPACE Region (TSR), and resides in a shared data space (a Local TSR is a private data space). After initiating contact with a VTS-TSR, a calling application can use it as if it were its own local TSR.

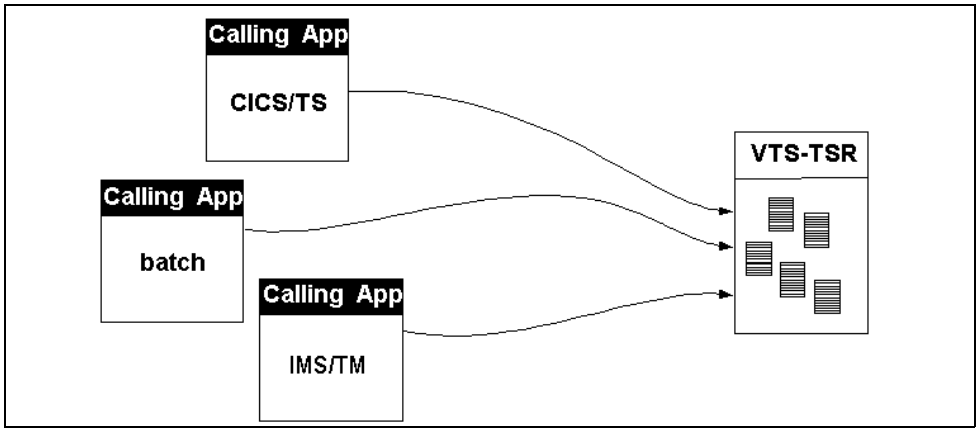

Figure 1-4: Calling application accessing VTS table data

A VTS-TSR may be called, using standard tableBASE commands, by any programming language that uses the standard IBM calling protocols including C and C++. For more information on VTS, please see the *tableBASE Administration Guide*.

VTS Application Examples

These are some examples of how tableBASE and VTS are used:

- Brokerage firms share several common-stock rate tables among applications that run in different regions. Stock information is dynamic, needs to be accessed quickly, and must be maintained in a common data pool. VTS makes this easy to implement by providing instantaneous access to a representation of live data that is shared among all users—a powerful time- and resource-saving opportunity.
- Service industries with data entry/inquiry systems executing in different IMS TM and/or CICS TS regions share common message tables, and any other tables common to the system. Online decision-support systems operate with the most current data at memory speeds.
- Banks share money market rates and lender information across several regions and environments. Online and batch applications process orders and transactions faster.
- Insurance companies share business rules, processing parameters, insurance rates, and customer information for online broker inquiry systems and actuarial or claims processing applications.
- Retail organizations have shared, in-memory access to store codes, freight tables, inventory lists, and supplier and customer information for online customer inquiry systems, distribution, sales forecasting, and reporting systems.

VTS Manager

VTS Manager is a shared table management product that extends the capabilities of tableBASE. It allows identical shared tables to be used across many LPARs within a Sysplex environment. VTS Manager also provides powerful change-control features for the seamless updating of high-use tables, and increases the performance of the applications accessing your table data.

For more information about VTS Manager, see the *VTS Manager Concepts and Facilities Guide* and the *VTS Manager Administration Guide*.

2 tableBASE Concepts

tableBASE Architecture

tableBASE is a programming tool that is installed on z/OS mainframes. The tableBASE software works with CICS TS, IMS TM, and Batch. The engine is fully re-entrant, allowing tableBASE to be used with multithreaded applications. tableBASE stores tables on DASD in libraries, and opens tables into a tableSPACE Region (TSR) for in-memory processing. An optional VTS (Virtual Table Share) product allows shared access of tables by multiple regions. Data spaces are used for both the Local TSR (private) and VTS-TSRs (shared).

Note: The term TSR is used to indicate a data space used to hold tableBASE tables in memory. A TSR can be either private (Local) or public (VTS-TSR).

tableBASE Nucleus

The tableBASE nucleus provides the core functionality of tableBASE and is the same for all environments. Each operating environment requires an instance of the tableBASE nucleus and a Local TSR.

Unified tableBASE stub

The tableBASE API in Version 6 can be used in CICS, batch, IMS, and VTS operating environments. tableBASE programs that use this stub can be ported from one environment to another without requiring any stub relinking. Of course, programs must continue to conform to the requirements of each execution environment.

tableSPACE Region

A tableSPACE Region (TSR) provides virtual storage for tableBASE tables—it is in a private data space, and has a maximum size of 2 GB. It is also known as the Local TSR.

The size of the tableSPACE Region determines the total amount of space that can be allocated for tables within a region.

There are two types of TSR: local and shared.

Shared TSR

A shared TSR is in a shared data space, and can be accessed by other regions. To use shared TSRs, you must install the VTS optional product. A shared TSR is also known as a Virtual Table Share TSR (VTS-TSR). There are two types of shared TSR: updateable and read-only.

Updateable VTS-TSR

The VTS-TSR provides a shared memory space for tables that need to be accessible to applications running in different operating environments—for example batch, CICS TS, TSO/ISPF, or IMS TM. VTS-TSRs are shared data spaces (see Figure 2-1). VTS-TSRs can increase performance by reducing paging, and save memory space by having only one copy of a table in memory.

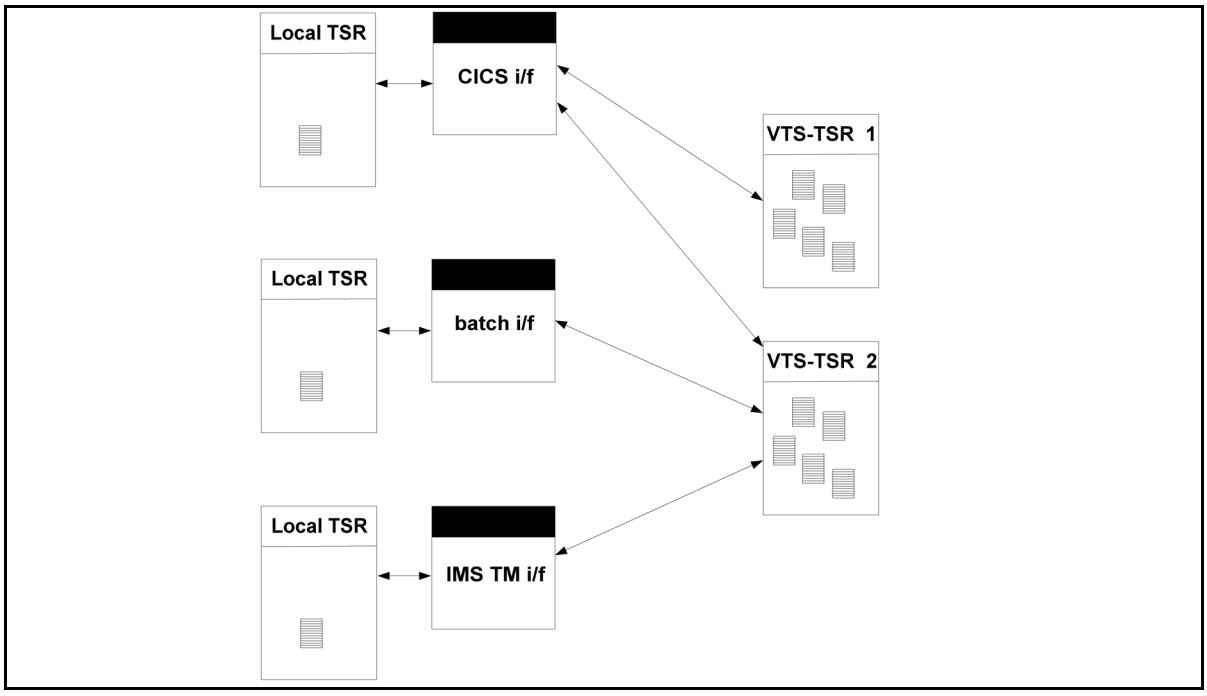

Figure 2-1: Local TSR and Updateable VTS-TSR

Read-only VTS-TSR

The VTS-TSR can be Read-Only when using the VTS Manager optional product. This means that the link from the Local TSR to the VTS-TSR only allowed the Local TSR to read data from the VTS-TSR (see Figure 2-2). The Local TSR cannot make updates to data in the Read-Only VTS-TSR.

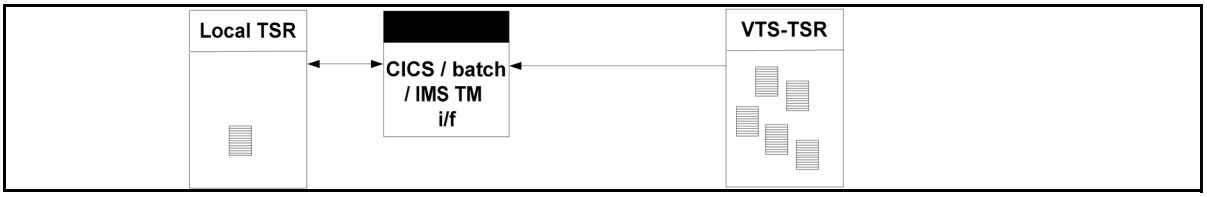

Figure 2-2: Read-only VTS-TSR

Memory model

The memory model uses segmented memory for efficient memory management. Segmented memory means the space allocated for data rows are not contiguous and therefore the rows do not need to be moved to accommodate updates. Instead, an Index is used for each table to point to the rows. This allows for efficient use of the local TSR and VTS-TSRs. When an entire segment becomes empty, the space is freed for reuse.

The data is maintained as efficiently as possible as a result of using segmented memory. When there is insufficient memory available to load data together, tableBASE will use whatever space is available — even if it is not contiguous.

All tables are stored internally as Pointer tables. Previous to Version 6, tableBASE allowed two type of tables: Pointer tables and True tables. True tables were characterized by not having an Index and by being stored in contiguous memory. In Version 6, the concept of True tables still exists, however they are treated within tableBASE as Pointer tables, as all memory is in segmented memory that requires Indexes. The True table Indexes are transparent to the application program.

Dataspaces

tableBASE uses Dataspaces for the local TSR and VTS-TSR. This allows the size of a TSR to be up to 2 GB. The use of Dataspaces also means that TSRs are protected against accidental overwrites by faulty application programs, and that no space is taken up for the local TSR from the region's above-the-line virtual memory. (Maximum table size is limited by the maximum 2 G size minus the system overhead.)

Dataspaces used for the VTS-TSRs do not affect the MAXCAD parameter of the MVS operating system. An IEFUSI MVS system exit may limit dataspace usage.

Paged tables

tableBASE loads the table into the Data Space and allows the operating system to handle paging, rather than having tableBASE paging individual blocks to and from a much slower DASD-based tableBASE library.

tableBASE Libraries

tableBASE stores persistent copies of tables in a tableBASE library on DASD. The delivered default includes one defined library (MAINLIB). There may be more than one library; some clients have a different library for each application, or for a series of applications, while others house all their tables in MAINLIB. Additional libraries can be defined by a call to tableBASE, or through the tableBASE utility called TBEXEC. When more than one library is used, application performance can be tuned by specifying the sequence in which libraries are searched to locate tables.

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

The tableBASE library is organized to optimize storage capacity and access time, so that:

- library facilities are shared
- tables load fast, with little overhead
- the need for compression is eliminated, because the libraries are self-organized
- up to nine table generations can be kept

Shared Library Facilities

tableBASE allows libraries to be shared between users, and 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 write, other programs, job steps, and tasks may read the table but will not be able to open it for write. The user may request that the task wait until the table is available (enqueue) then continue with processing. tableBASE also allows different tables to be stored into the same library concurrently by multiple tasks.

tableBASE tables

For enhanced flexibility and performance, tableBASE tables are structured differently from the typical row and column tables found in spreadsheets. While a tableBASE table can easily contain the data from a spreadsheet, it can also contain data of far greater organizational complexity.

tableBASE tables are a collection of fixed-length data rows. Each row contains a key and unstructured data.

Applications that access tableBASE tables—for example, tablesONLINE—apply column definitions to the table data. The column definitions themselves (metadata) are contained in tables in tablesONLINE.

Besides data rows, a tableBASE table also includes a table definition that contains attributes such as row length, key location, key size, organization, and search method. The table definition is used to generate the primary Index.

Although tableBASE tables are usually loaded from, and ultimately stored on, a hard disk, tableBASE tables are entirely resident in memory during processing, providing a greater performance advantage over disk-based tables for many types of data.

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.

Table Organizations and Search Methods

tableBASE provides a variety of table organizations and corresponding search methods to facilitate table access. Because tableBASE works in memory, more efficient algorithms for storing and retrieving data can be used than those available in a conventional, disk-based alternative. This flexibility allows for optimal application performance.

Table Organizations

Since tableBASE operates in-memory, it offers table organization options optimized for in-memory use. Tables can be organized in any of four ways:

- Sequential
- Hash
- Random
- User-controlled sequence

Sequential

Sequential tables are ordered by ascending or descending sequential keys. Sequential tables can be searched using one of:

- Queued Sequential Search
- Binary Search
- Bounded Binary Search.

Hash

Hash tables are sequenced using a hashing algorithm that determines the location of a row of data in the table. 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 Hash Search.

Random

There is no particular sequence to the table. New rows are inserted at the end of the table. In releases prior to Version 6, deletions caused the last row to be moved into the empty space. In Version 6, there is no such movement into the empty space, simply assume that the entries are in random order.

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 controlled by the application program. The location of insertion of a new row is controlled by the COUNT field. If inserting a row using a key, the insertion is at the end of the table.

Like 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

The tableBASE software offers you a variety of search methods that are optimized for performance, table organization, and in-memory use:

- Serial Search
- Queued Sequential Search
- Binary Search
- Bounded Binary Search
- Hash Search

The search is applied to the Index and the Index then points to the appropriate row of the Data Table.

Serial Search

A serial search compares the search key with the key of each row in the Index. The search begins at top of the Index and progresses through the Index until the row is found that contains the search key or until all rows in the Index have been examined.

A serial search uses little overhead and is the fastest search method for small Indexes. It is also more efficient for user-controlled Indexes with a skewed frequency of hits. The user can place the most frequently accessed rows at the top of the Index.

The table organization must be Random or User-controlled sequence.

Queued Sequential Search

This search method is executed by progressing serially through the Index and comparing the search key to each Index key until a row is found. Subsequent searches begin where the last row was compared, if the search key is farther along in the Index sequence. If the key of the next row examined is too high (or too low for descending sequential Indexes), the search resets to start from the beginning of the Index.

Queued sequential searches are faster on partially-sequenced or mostly-sequenced data—for example, transaction matching in a master file type of update process.

The table organization must be ascending or descending Sequential.

Binary Search

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

The table organization must be ascending or descending Sequential.

Bounded Binary Search

The bounded binary search process compares the search key to the endpoints of an Index to determine if the search key is within the Index range. If the search key is not within the range, then the system returns a not found message. If the search key is within the range, then a binary search process is used to find the key position.

An bounded binary search is a fast technique for performing inserts into ordered data.

The table organization must be ascending or descending Sequential.

Hash Search

This searching technique randomizes the key to calculate the location of a row in the table. A hash search is the best search method for large tables that are usually accessed randomly. Performance is enhanced because there is no search looping, however this creates greater software overhead becuase of the calculation required for the search.

Under normal circumstances hash searches and hash-based insertions use less CPU time than binary, serial, and most sequential searches, and require fewer page accesses than any other search.

Space utilization is minimized by a Hash Pointer Table—the actual data is kept in a densely packed random table, while the more sparsely-packed Index contains only the address of the data rows.

The table organization must be Hash.

Search Summary

Table 2-1 summarizes the valid combinations of table organizations and search methods allowed by tableBASE.

Table 2-1: Search Summary

Table Organization	**Search Methods**				
	Queued Sequential	Bounded Binary	Binary	Serial	Hash
Sequential	Y	Y	Y		
Descending Sequential	Y	Y	Y		
User-Controlled Sequence				Y	
Random				Y	
Hash					Y

Indexes

tableBASE Indexes are dynamically generated in memory when a table is opened from the tableBASE library (see Figure 2-3). Indexes are not stored in the library with the data, but are created using the table definition. As a result, 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. This feature of tableBASE allows for experimentation with different table organizations to optimize data access using the table definition

These features extend indexing capabilities beyond what is practical or possible with a DBMS. Online 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.

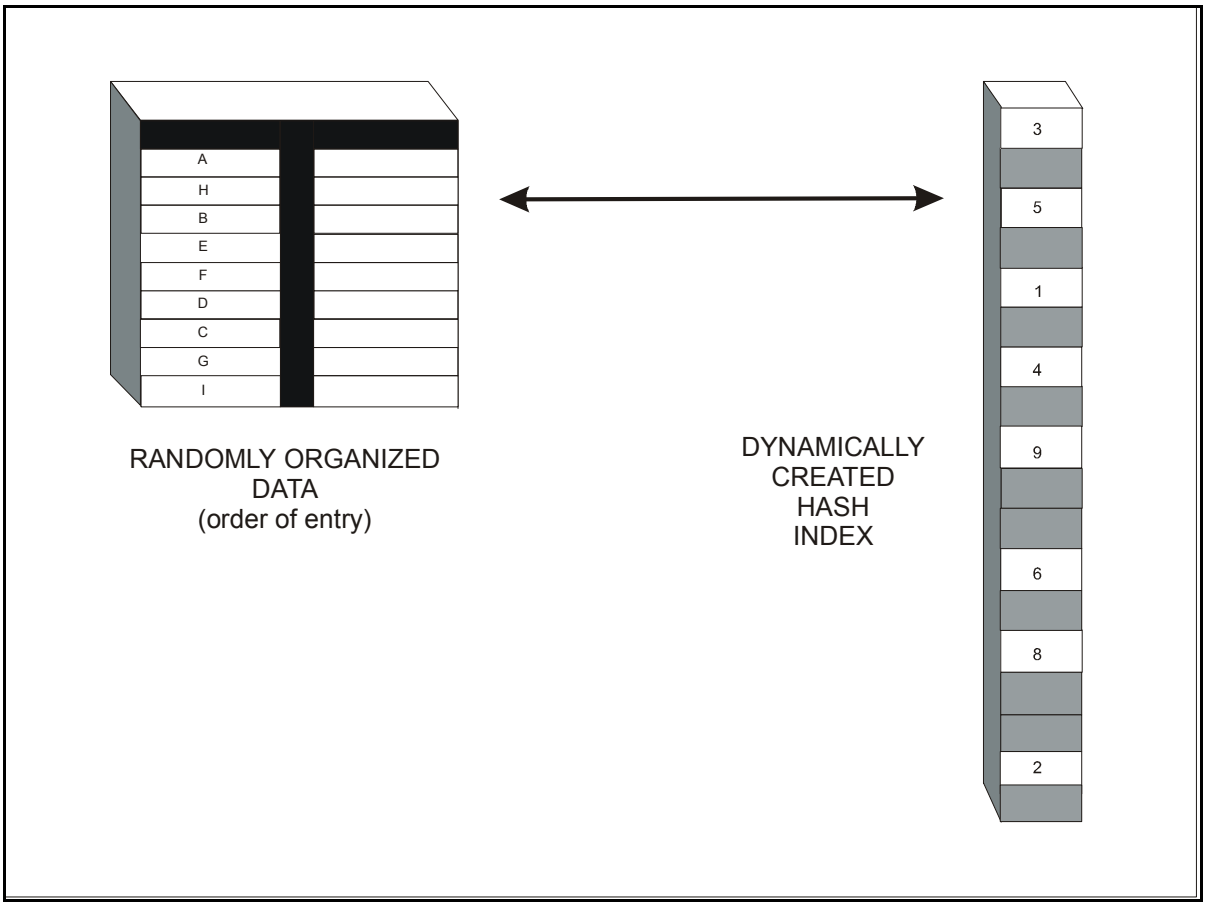

Figure 2-3: Dynamically Created Indexes Save Space and Process Faster

Alternate Indexes

An Alternate Index permits access to a Data Table using a different key, organization and/or search method than those originally defined in the table definition (see Figure 2-4). This allows for many different ways of indexing the data without generating multiple copies of the data. Alternate Indexes are generated by alternate table definitions (ALT-DEFINITION). Since there is a single copy of the Data Table, all Alternate Indexes reflect any changes made to the Data Table.

tableBASE supports many-to-many (M2M) relationships between Alternate Indexes and Data Tables. One Alternate Index may be associated with different Data Tables of identical structure. Similarly, many Alternate Indexes may be defined for a single Data Table.

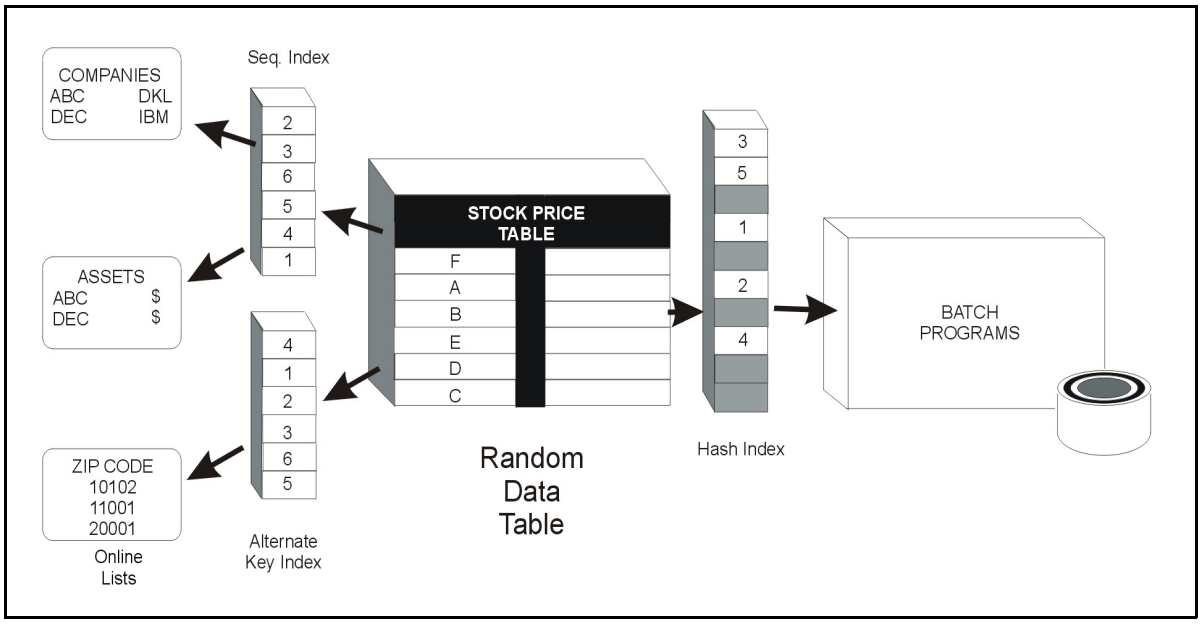

Figure 2-4: Alternate Indexes

tableBASE Application Programming Interface (API)

tableBASE can be accessed programmatically in both batch and online environments, or interactively at a terminal under CICS TS or ISPF.

The API for tableBASE is called TBLBASE, which offers a consistent API for batch, CICS TS, and IMS TM environments. TBLBASE is available for programming languages using standard IBM calling conventions. The most common usage of TBLBASE is to retrieve and update items in a table.

TBLBASE

When an application requires a table row, the program makes a call to TBLBASE. If this is the first access to the table, a data space will be allocated to accommodate the table. The table is loaded into a designated area in the data space called the tableSPACE Region (TSR), which is managed by tableBASE services. For a retrieval request, tableBASE searches the table according to its organization and search strategy, and returns the requested row to the application program.

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 (see Figure 2-5). 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. Any number of applications may access one table for read access, however write access is limited.

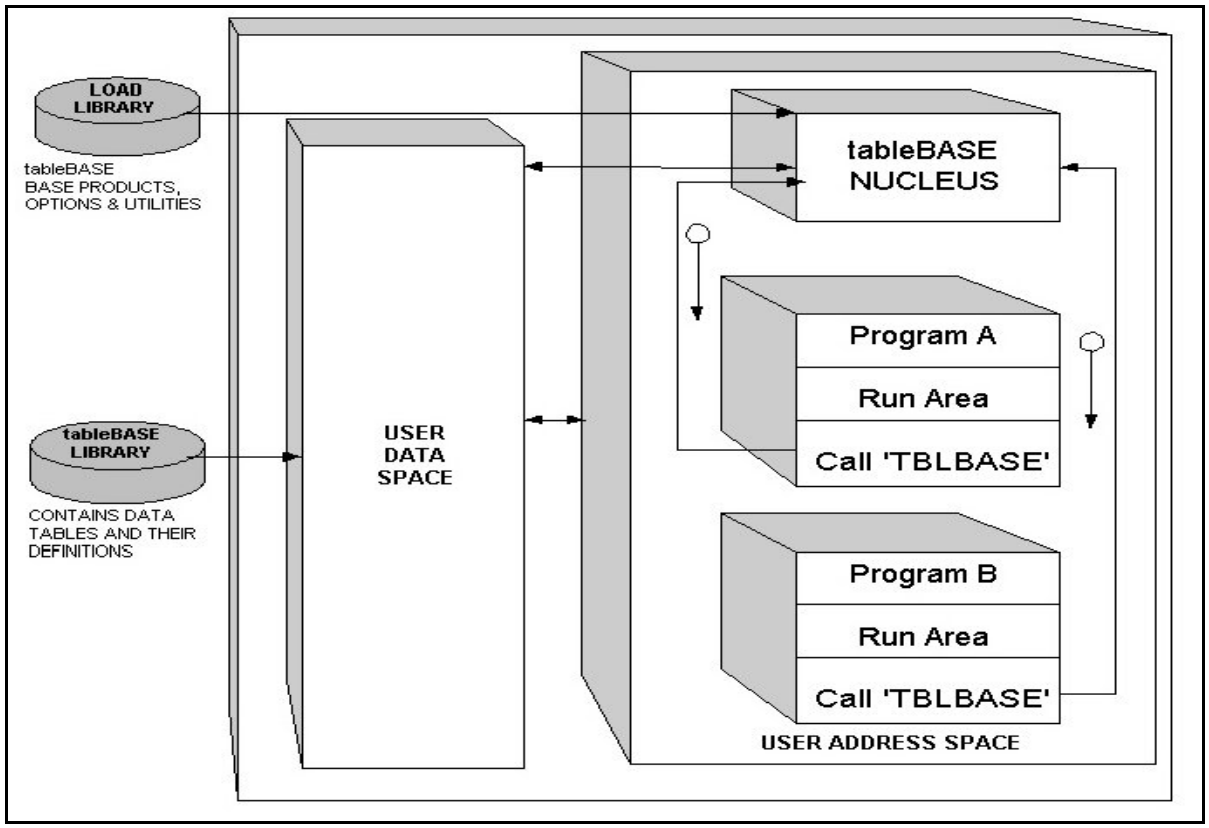

Figure 2-5: Accessing Tables

Protecting tableBASE tables

tableBASE tables can be protected in the library using read and write passwords, and can be protected when loaded into memory using a LOCK-LATCH. These passwords offer only limited protection against unauthorized use of tableBASE tables. More formal security can be implemented with user exits and third party measures such as Resource Access Control Facility (RACF), eTrust CA-ACF2 Security for z/OS (ACF2) and eTrust CA-Top Secret Security for z/OS.

tablesONLINE exits provide protection at the table level, field level, and row level. See the *tableBASE Programming Guide* for more information.

Note: See your tableBASE administrator if a table password has been lost or forgotten.

Read and Write Passwords

Passwords are used to open tables from the library. Once in memory, these tables can be accessed by any application that has access to the memory.

A read password protects a table from being opened for either read or write.

A write password protects a table from being opened for write. For more information on passwords see the *tableBASE Programming Guide*.

LOCK-LATCH

LOCK-LATCHES are used by tableBASE to protect a table in memory. Applications give a LOCK-LATCH password when the table is opened in a multi-user environment. Subsequent operations must use the same password in order to perform updates.

Maintenance

tableBASE Maintenance

tableBASE requires minimal maintenance. It is self-adjusting to accommodate growth and provides easy mechanisms for modifications. For more information on the batch utilities provided to assist with maintenance see "Batch Utility Programs" on page 39.

Once in production, many users find that tableBASE can operate for years without any need for attention.

Table Maintenance

Table maintenance is integrated into tableBASE. No extra design or programming time is required to develop auxiliary maintenance subsystems. Table entries can be added, changed, and deleted with a simple command. Changes to both table size and structure are automatically accommodated by tableBASE.

Table maintenance becomes a simple matter of filling in the blanks. In conjunction with development of table-driven, rule-based systems, this allows more direct end-user control over applications. Increased end-user responsibility and control over table data means greater control over system behaviour, eliminating the need for the change request process required by traditional application types. A programmer's time is saved from program maintenance, table maintenance, and the development of table maintenance software.

The end user can oversee the updating and maintenance of a single version of the table for convenience and increased control. If various table 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, processing changes and variations can be planned and executed in a more orderly fashion using table-driven controls, rather than involve programming staff in last-minute program updates.

VTS-TSR Maintenance

Many tableBASE users execute VTS-TSRs on a 24x7 basis. No maintenance is required.

Installation

A standard tableBASE installation takes one person approximately two days. For more information see the *tableBASE Installation Guide*.

Administration

tableBASE requires a minimal amount of administration. Most of the time required by the tableBASE administrator is spent during the installation and set up of tableBASE in the development environment. For more information, please see the *tableBASE Administration Guide*.

Development

tableBASE software provides developers with the ability to place data in memory for quick access, reducing I/O and CPU usage. It is a flexible tool with minimal overhead. User exits have been provided in tableBASE to allow developers to add custom functionality such as authorization security, auditing, and commitments. For more information, see Chapter 4 – Using tableBASE.

3 tableBASE Facilities

This chapter describes the driver and utility programs that are part of the tableBASE product.

Command Executors

CICS TS

In CICS TS, TBDRIVC is the pseudo-conversational transaction that accepts tableBASE commands from the terminal and invokes the TBLBASE API to execute them. It allows interactive access to all tableBASE operations. For more details, please see the *tableBASE Programming Guide*.

Batch and TSO/ISPF

In batch and TSO/ISPF, the TBDRIVER program interprets input driver commands, converts them to corresponding tableBASE commands, and invokes tableBASE to execute them. It allows interactive access to almost all tableBASE operations, and supports some special macro commands of its own. For more details, please see the *tableBASE Programming Guide*.

Utility Programs

Batch Utility Programs

There are several batch utility programs supplied with tableBASE. These utilities are designed to simplify the task of maintaining the tableBASE environment. The key programs are TBEXEC, TBPRINT, TBDEFPRT, TBCOBFD, and TBCOMP.

TBEXEC

This is the primary tableBASE batch utility. It provides basic functionality such as initializing new table libraries, defining tables, deleting tables, making mass updates to tables, and copying tables among libraries. TBEXEC also provides a print facility especially designed for the printing of tables used in testing.

TBPRINT

TBPRINT is a tablesONLINE reporting utility program that 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 create meaningful, customized reports.

TBDEFPRT

TBDEFPRT prints Views and field definitions.

TBCOBFD

TBCOBFD generates copybooks from table definitions for use in COBOL programs.

TBCOMP

This utility compares tables and identifies any differences for review.

Conversion Utility for Libraries

All tableBASE libraries must be converted to Version 6 format for use with tableBASE Version 6. A conversion utility called DK15CONV has been provided to assist with library conversion from Release 5.0 to Release 6.0.

Note: tableBASE Version 6 libraries are not compatible with previous releases.

For information on converting a library from Version 5 to 6, please see the *tableBASE Installation Guide*.

4
Using tableBASE

This chapter describes how to use tableBASE.

tableBASE Development

Accessing data from memory can be more than one thousand times faster than accessing data from DASD. tableBASE enables developers to leverage mainframe memory for performance improvement by placing data into memory-resident tables.

Typically, there are four roles involved in the development of tableBASE applications: administrator, data analyst, programmer, and end user.

Administrator

The tableBASE administrator is responsible for the initial set up of tableBASE, performing product installation, configuration, and ongoing administrative functions. For more information, see the *tableBASE Administration Guide*.

The tableBASE administrator:

- installs and configures tableBASE components
- coordinates implementation of tableBASE upgrades
- maintains policies and procedures for tableBASE administration and use
- sets up users and user authorizations
- defines tableBASE libraries
- maintains company-wide table libraries
- maintains security rules for tableBASE libraries
- defines and controls naming standards for tableBASE tables
- generates and maintains the tableBASE Master Password
- assists system support with tableBASE start-up jobs for all online regions

Data Analyst

The tableBASE data analyst designs a set of tableBASE tables that meet application requirements.

Programmer

Programmers develop batch and/or online applications.

Systems personnel write specialized software for managing job scheduling, terminal routing, and/or enhanced security.

For more information see the *tableBASE Programming Guide*.

End User

End users are usually professionals like data-entry clerks, stock brokers, bank personnel, or insurance adjustors who use tablesONLINE, or another custom interface, to create, update, manipulate, test, and process Data Tables.

For more information see:

- *tablesONLINE/CICS User's Guide*
- *tablesONLINE/ISPF User's Guide*.

tableBASE Commands

The tableBASE software has five command groups that perform the functions required to maintain and access tables, they are:

- Retrieval Commands—provide the facility to retrieve entries from a table by key or by entry count
- Update Commands—allow for modification of the contents of the table either by key or by entry count
- Table Control Commands—provide the facility to open, close, store, define tables, and the ability to change the definition of individual table generations
- Directory Maintenance Commands—provide the facility to alter information in the directory about individual tables
- System Control Commands—specify how tableBASE handles error conditions, contention situations, and the library search order

Table 4-1: Retrieval Commands

Command	Description
SK	Search by Key
FK	Fetch by Key
FC	Fetch by Count
FG	Fetch Generic
FN	Fetch Next by key
GF	Get First
GL	Get Last
GN	Get Next
GP	Get Previous

Table 4-2: Update Commands

Command	Description
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 4-3: Table Control Commands

Command	Description
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 4-4: Directory Maintenance Commands

Command	Description
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

Table 4-5: System Control Commands

Command	Description
ML	Modify Library search order
LL	List Libraries in use
CS	Change tableBASE processing Switches
LS	List tableBASE processing Switches
LT	List open Tables
BN	BaNner retrieval
SI	Set Indirect
DE	DisEngage library
AL	Allocate Library
UL	Un-allocate Library
VS	set Virtual System

Basic tableBASE Activities

There are five basic table activities that can be accomplished using tableBASE:

- Define a Table
- Open a Table
- Access a Table
- Store a Table
- Close a Table

Define a Table

A single command defines the table in memory. The table can then be stored in a designated library. The DEFINE A TABLE command can be invoked:

- interactively under tablesONLINE using our standard (or user-developed) menu-driven system
- directly through a call by the application system
- by the TBEXEC batch utility program or the TBDRIVER Toolset

tableBASE automates and controls all the other aspects of table management and memory management.

The following table attributes can be easily defined and modified, without requiring a change to application program logic:

- organization
- search method
- indexing
- Storage method
- passwords (read and write)
- row size
- key size
- key location
- number of generations
- expansion factor
- density (hashed tables)

Note: Defining a table automatically opens the table for write.

Open a Table

When a command is invoked that opens a table, the table is loaded into memory. A table opened for read-only access may be modified in memory but not stored to the library. A table opened for write access may be stored back to the library.

Unlike many databases, tables in tableBASE may also be opened implicitly by commands such as Get First (GF) or Get Last (GL).

Access a Table

Access to tables in memory is accomplished using the table retrieval and update commands.

Indirect Table Access

This facility allows a table to be opened indirectly, by referencing a table name in another table, the primary table (see Figure 4-1). You can access a secondary table using 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-sensitive or date-sensitive tables can go into production according to a specified time, date, or other criteria without operator intervention.

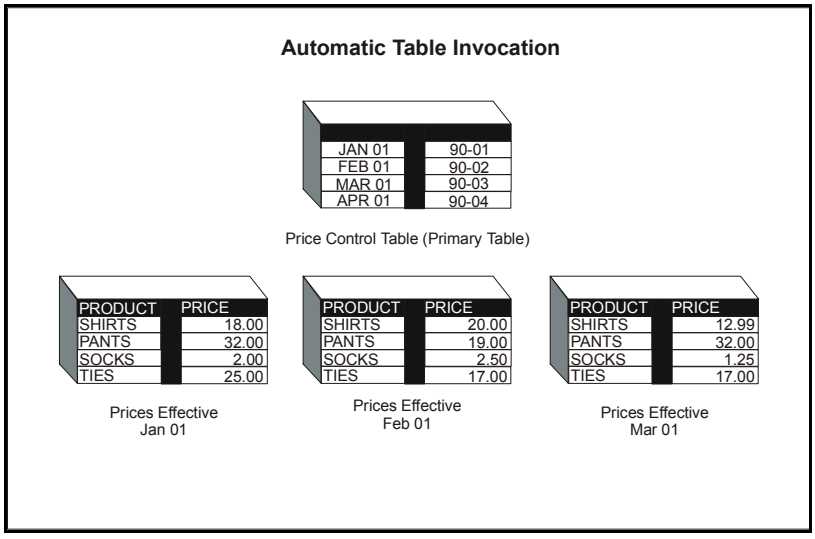

Figure 4-1: Indirect Access to Tables for Date Sensitive Data

Store a Table

If a table is opened for write, the process of storing a table writes it into the library (DASD).

tableBASE tables are stored in optimized form in tableBASE libraries to minimize 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 criteria allows development of test versions and pending versions.

As tables are created, updated, and deleted, tableBASE automatically reorganizes table library space; dataset compression is unnecessary. The tableBASE memory environment

and directory maintenance routines support all-or-nothing update failure protection. If the system or application terminates abnormally for any reason during update processing, 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 holding complete sets of information, where each row of the table is expected to be in sync with all other rows. Each row in the table shares interdependencies with every other row. Sometimes these interdependencies are subtle. For example, in a table of currency exchange rates, all rows are assumed to be current at the same date and time—partial updates are unacceptable.

Generations

Each time a table is stored, a new generation of the table is created. With up to nine generations of each table available, users can recover from a bad update by replacing invalid table generations with older generations of the table.

Close a Table

This activity frees memory that has been occupied by a table, and removes the table definition from the TSR.

An Example: Open, Store, and Close a Table

Before issuing a call to access a row, a table and its desired action needs to be specified to tableBASE. For example, to open a fictitious table called PAYROLL, code something like:

```
MOVE 'OR' TO PAYROLL-COMMAND.
CALL 'TBLBASE' USING TB-PARM

                    PAYROLL-COMMAND-AREA.
```

It is assumed that the table has been identified in PAYROLL-COMMAND-AREA. If not, precede the above code with:

```
MOVE 'PAYROLL' TO PAYROLL-TABLE.
```

Above, OR means that the table will be opened for read-only access. The OW command opens a table for read and write access. Once the table is read from the library into memory, all operations take place in memory. If changes to the table need to be saved, the table must be stored into the library using the Store command, like in the following example:

```
MOVE 'ST' TO PAYROLL-COMMAND.
CALL 'TBLBASE' USING TB-PARM
                    PAYROLL-COMMAND-AREA.
```


When a table is open, tableBASE allocates space in memory to accommodate the table. The table is loaded into the TSR. For a retrieval request, tableBASE searches the table according to its organization and search strategy, and returns the requested row to the application program.

All table accesses are performed in memory. Organization and search methods can be modified dynamically in memory. The table remains in memory until the table is closed or the TSR holding the table terminates.

An example of code that closes a table is:

```
MOVE 'CL' TO PAYROLL-COMMAND.
CALL 'TBLBASE' USING TB-PARM

                     PAYROLL-COMMAND-AREA.
```

Note: Closing a table does not store it.

5 Using VTS (Virtual Table Share)

The DataKinetics VTS product augments the tableBASE core product by providing the capability to share table data. Using tableBASE, regions load tables into a local table space region (TSR). Many regions loading tables into their TSRs simultaneously can hamper system performance, especially if the regions are using the same tables. After augmenting the core product with VTS, many regions can access the same data from a single shared TSR; a virtual table share (VTS-TSR). This can:

- improve system performance both at IPL and during run-time
- reduce system resource usage including system memory, CPU cycles and I/O

tableBASE VTS concepts are introduced in "VTS (Virtual Table Share)" on page 22, with more detail under "tableSPACE Region" on page 25.

This section describes how you can use tableBASE VTS to:

- Improve system performance
- Reduce system resource usage
- Improve data integrity
- Implement message queueing

Performance enhancement using VTS

Performance issues experienced when over-taxing the local TSR

In some instances, as demand for data access increases, and tableBASE usage increases, customers can experience increased performance issues both at initialization, and during run-time. Consider the scenario below, where a large number of tables are loaded into a single TSR from multiple libraries.

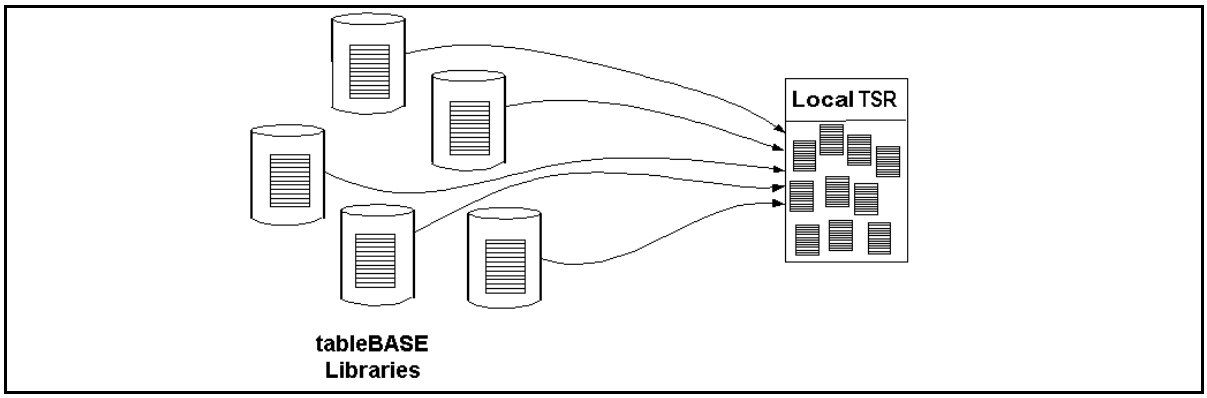

Figure 5-1: Performance issues at initialization

At initialization, data is loaded from tableBASE libraries into the local TSR row-by-row, with associated indexes being constructed for each table. This is a very I/O intensive process, and can be a quite time-consuming, particularly if there are a large number of tables being loaded from a large number of libraries, as is the case here.

The situation can be made worse if another set of tables has to be loaded after the first set. The sequential nature of loading significantly delays the application for the entire duration.

During run time, access congestion can be an ongoing performance issue, especially if the local TSR is burdened with many tables (access is not efficient if a TSR contains many tables).

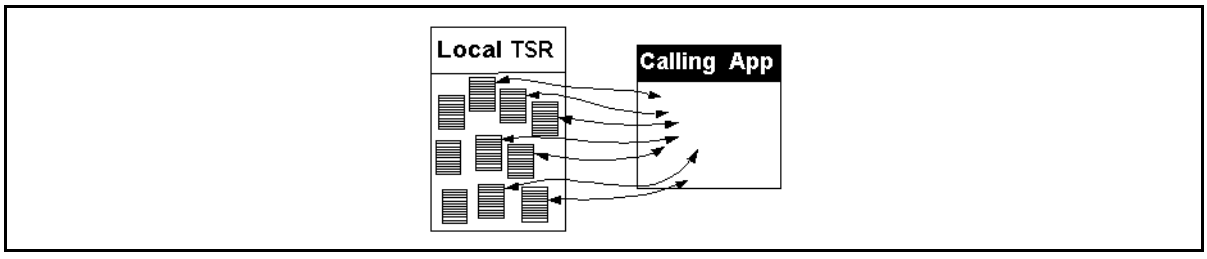

Figure 5-2: Performance issues at run-time

DataKinetics VTS product, an in-memory data-sharing performance-enhancement product which augments the tableBASE core product, is the solution for both initialization and run-time performance degradation.

VTS provides the solution

The VTS product can dramatically improve performance during both initialization time and run time. This is accomplished using a shared TSR, known as a VTS-TSR. Consider now, the same scenario as before, but this time using a shared VTS-TSR in addition to the local TSR used before (see Figure 5-3).

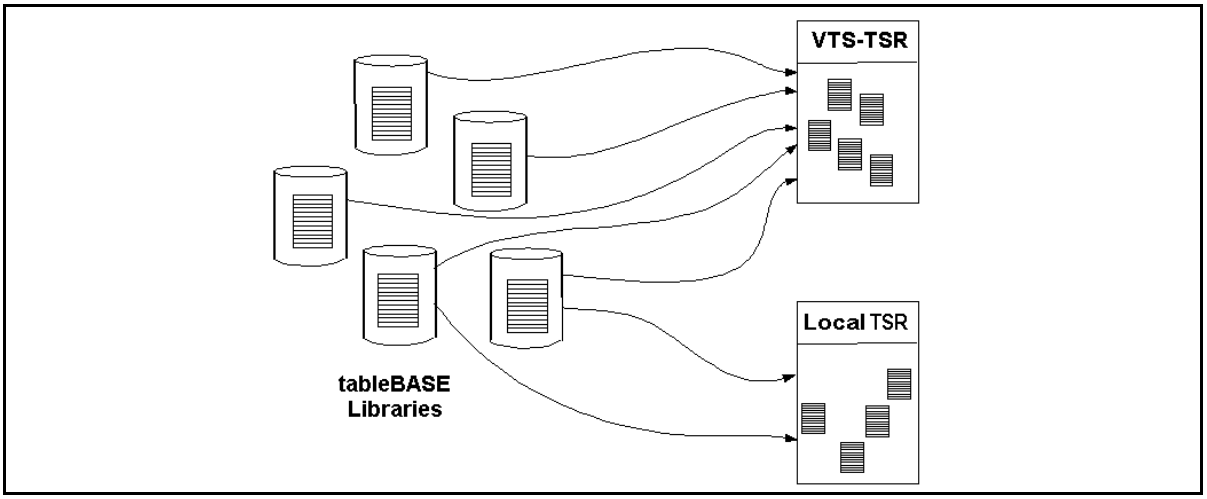

Figure 5-3: VTS-TSR off-loads the Local TSR

This divides the burden between two TSRs, rather than housing all tables within a single TSR. Access to tables is more efficient if there are fewer tables in a given TSR.

At initialization time, the considerable amount of time, CPU and I/O activity required to load data from the tableBASE libraries into the local TSR can be reduced considerably. This translates into sharply lowered initialization times for the local TSR. The shared VTS-TSR is loaded with table data once and is left running—tables can be reloaded without taking down the entire TSR.

During run-time, space is freed-up in the local TSR (see Figure 5-4); it no longer has to serve as a data repository for those tables moved to the VTS-TSR. The VTS-TSR, which contains only table data, acts as a very efficient in-memory repository. The VTS-TSR outlives the applications that use it—and it can be started, initialized, populated with tables, and made available to the applications that use the data.

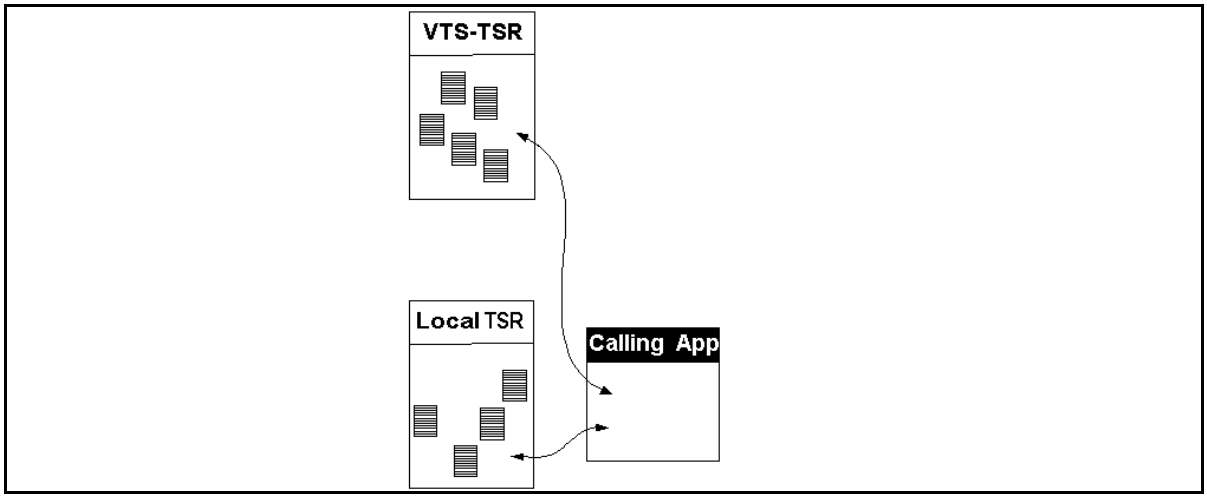

Figure 5-4: Improved performance with fewer tables per TSR

More options with VTS

While the solution shown above is a powerful response to the performance problems of the described scenario, there are still more possibilities offered by VTS.

No tables in local TSR

There are some advantages in relocating all tableBASE table data into a VTS-TSR. This solution allows very fast application initialization—no tables have to be loaded into the local TSR, all data stays in the VTS-TSR, which is independent of application initialization and shut-down.

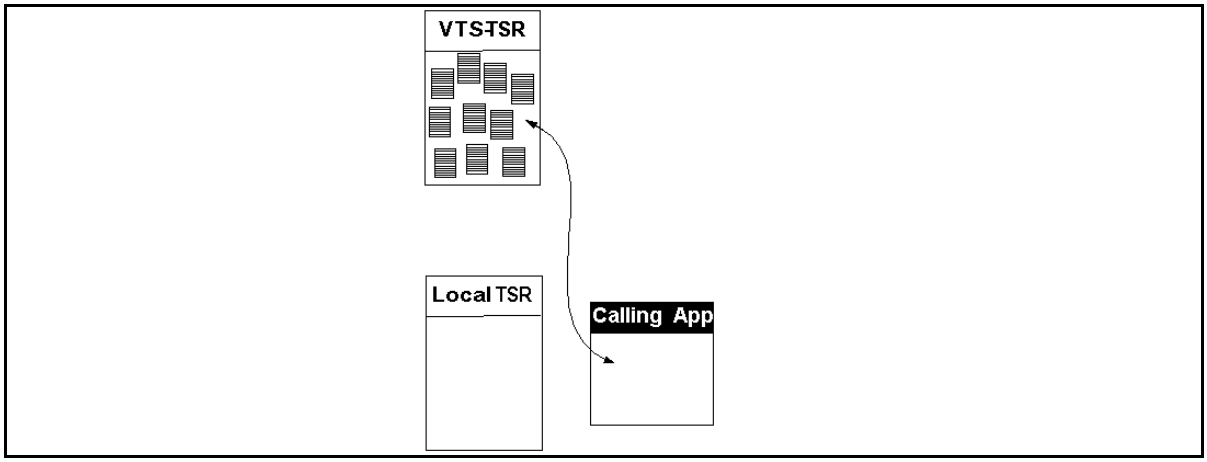

Figure 5-5: No tableBASE data in local TSR

Another advantage is that the application is completely isolated from the data, making the data impervious to application corruption or failure.

Multiple application environment

There are some advantages in accessing tableBASE data from multiple applications. First, VTS has inherent compatibility features that allow access from multiple environments simultaneously, allowing for diversified applications.

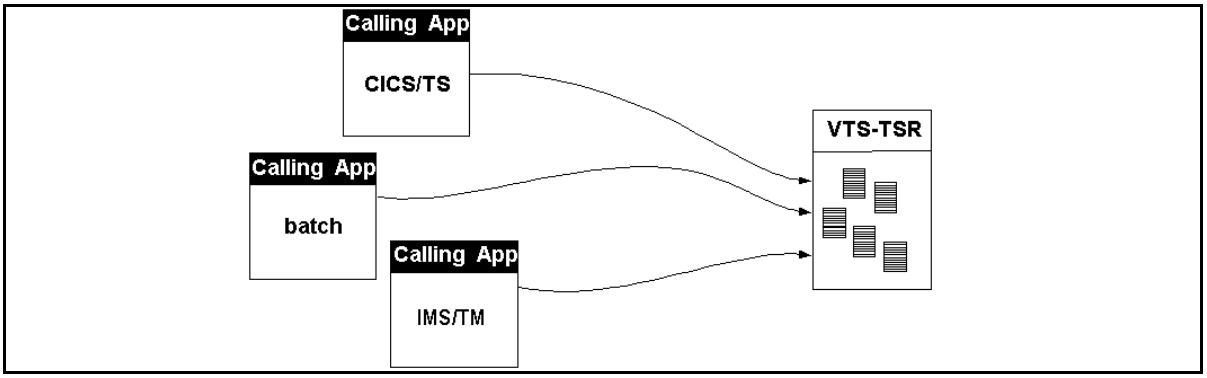

Figure 5-6: Multiple applications accessing shared tableBASE data

Applications from different regions (e.g., CICS/TS, batch, IMS/TM, etc.) can access the same shared data within a VTS-TSR. Second, cloned applications in multiple address spaces can access the same shared data, avoiding application transaction limitations, and increasing overall system throughput.

Multiple VTS-TSRs

It is possible to use multiple VTS-TSRs to further limit access congestion (recall that access to tables is more efficient if there are fewer tables in a given TSR).

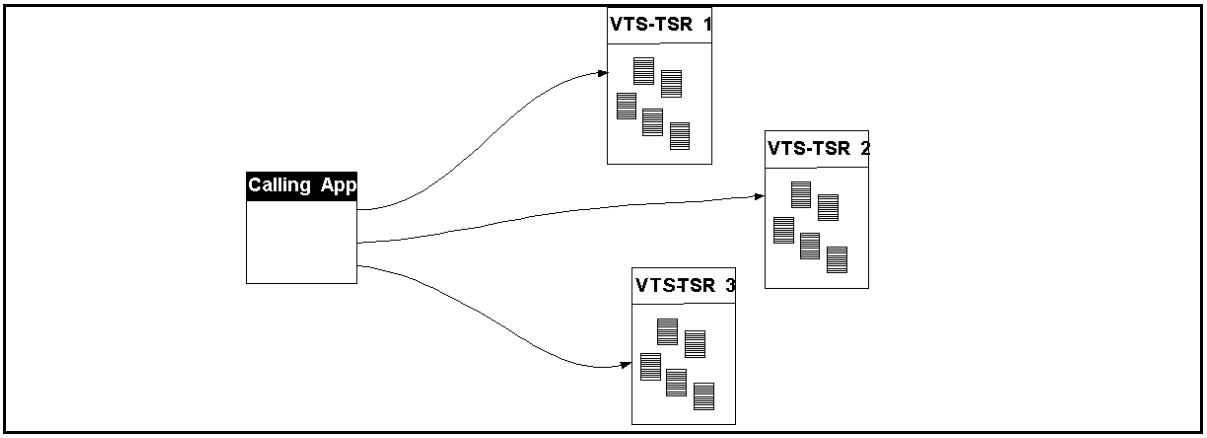

Figure 5-7: Multiple VTS-TSRs

By combining solutions (see Figure 5-8), one group of applications can access a common VTS-TSR, while another group of applications can access a second common VTS-TSR. In this way, access traffic in one application group does not affect the performance of the second application group.

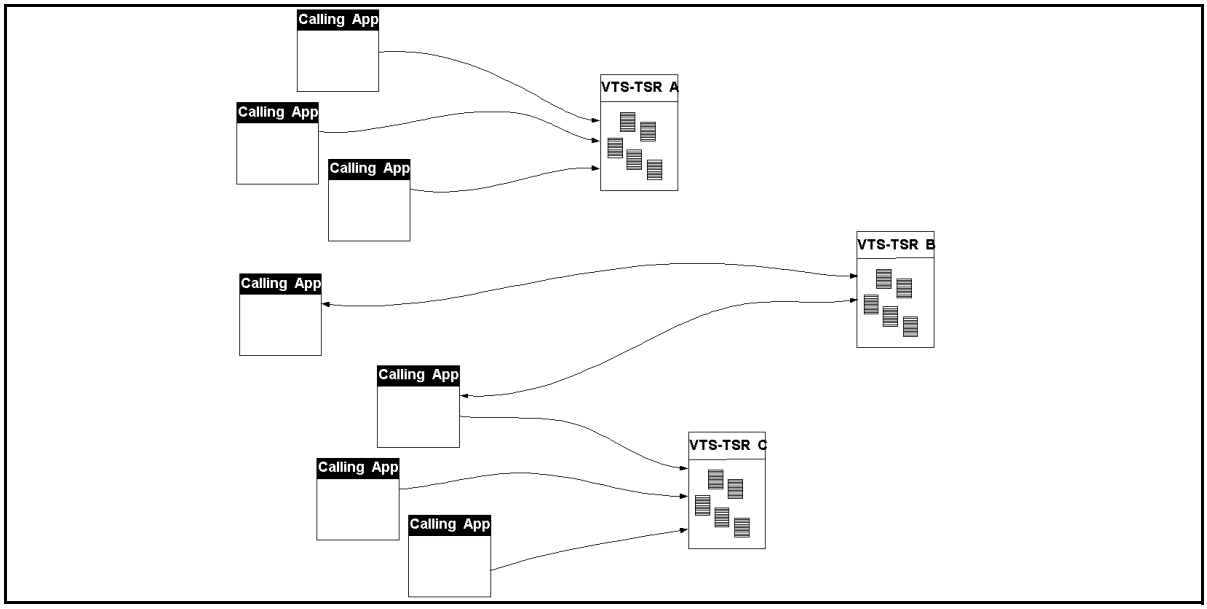

Figure 5-8: More VTS solutions

Read-only tables provide faster access than read-write tables due to the shared locked mechanism. To take advantage of this, all read-only tables can be located in one shared VTS-TSR, while all read/write tables can be located in another. In this way, accesses to the read-only tables will not be affected by updates, which will significantly improve access performance for the read-only tables, for all accessing applications.

Reduction of system resource usage using VTS

Like all programs, tableBASE-empowered applications must use a certain amount of system resources. In a typical single-region scenario, a tableBASE TSR uses real memory (it actually resides within memory). Also, any application using tableBASE will also use paging storage resources (this is managed by the operating system), CPU cycles and I/Os.

As a business scales, so too does the use of system resources. As more applications are brought online to handle growing business needs, the demand on these resources multiplies. Applications using tables simultaneously, will use resources simultaneously. At some point, you will be faced with expensive upgrades in hardware.

After augmenting the core product with the VTS product, all regions can access the same data from a shared TSR (see Figure 5-6); a VTS-TSR, which uses a fixed amount of system resources—far less than the amount used by multiple regions with dedicated TSRs.

In this way, tableBASE and VTS will help you reduce the replication of data, thereby reducing the demand on both system memory and paging storage, and therefore reduce the need to upgrade hardware.

Improved data integrity using VTS

In a typical single-region scenario (see Figure 5-9), data is sourced from a tableBASE library, into a memory-based TSR. Application(s) within a CICS or IMS region will then access the data in the TSR. Using tableBASE in applications in this way leaves little room for DASD data integrity issues.

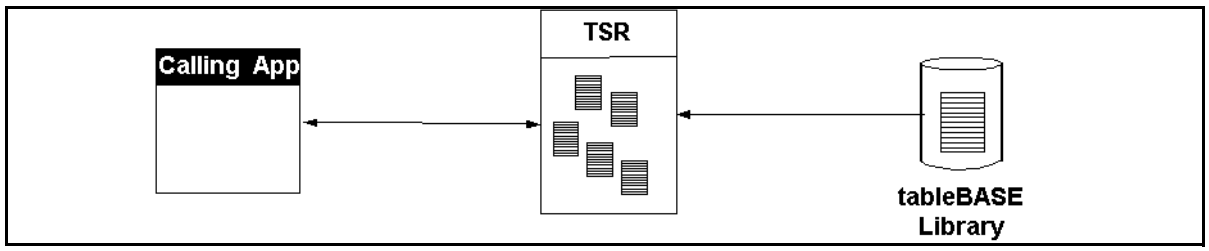

Figure 5-9: TSR access using a single region

However, problems can arise if care is not taken, when multiple CICS regions access data sourced from your library. For example, consider the scenario where three applications load data into their local TSRs from a library (see below). If applications within the top two regions perform only read operations, while applications in the bottom region perform both read and write operations, it is apparent that there can be loss of data integrity. When it shuts down, the library is updated, making it inconsistent with the top two TSRs.

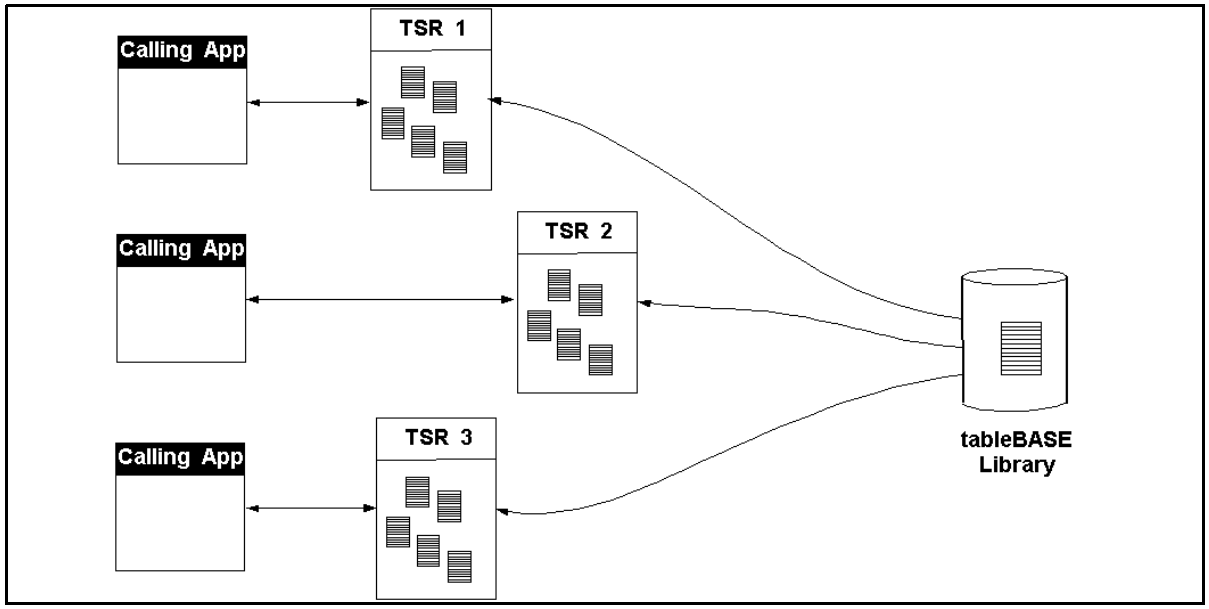

Figure 5-10: TSR access using multiple regions

Access control measures can solve the problem, but will very likely impact performance. The more powerful solution is VTS.

VTS enhances data integrity

VTS augments the tableBASE core product by providing the capability to share table data. Using tableBASE, every region loads tables into a local TSR. After augmenting the core product with VTS, all regions can access the same data from a shared TSR; a VTS-TSR.

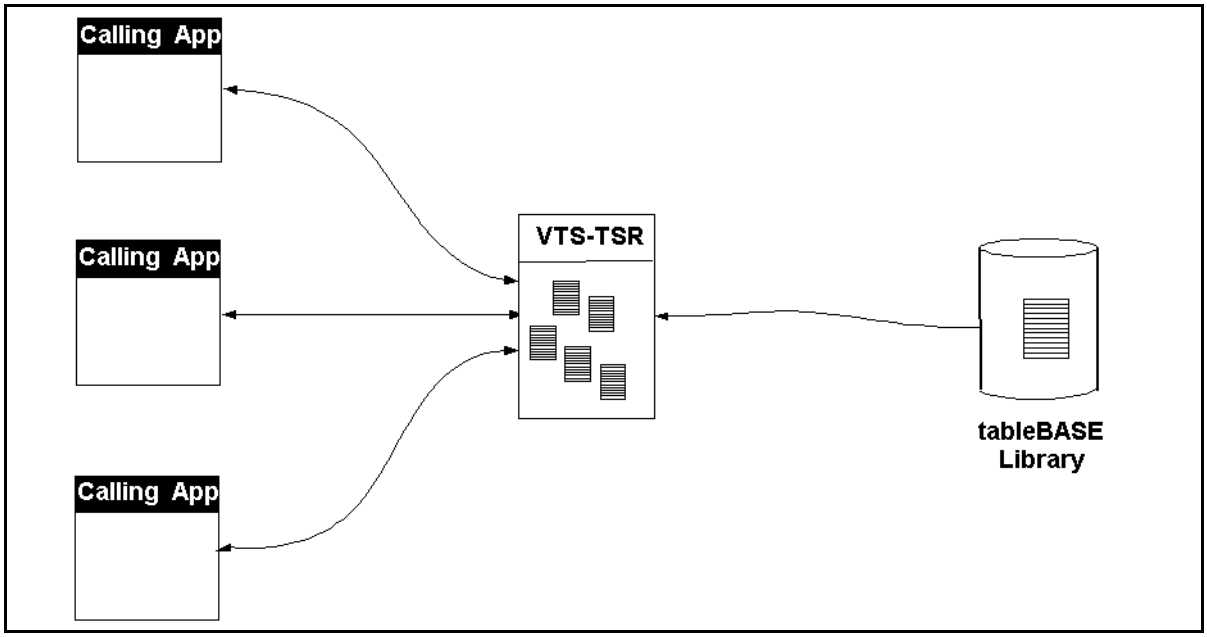

Figure 5-11: Shared TSR access using multiple regions

Considering this scenario, if a single shared TSR is used, rather than the local TSRs, all regions will access a single copy of the data. Changes to the data made from one region are visible to all other regions immediately.

The problem of maintaining data integrity for multiple regions is now moot—this solution virtually guarantees data integrity across regions.

6
Using tablesONLINE

Accessing and Maintaining tablesONLINE

The DataKinetics tablesONLINE product is a flexible, interactive front end to tableBASE. It provides speed and convenience for end users that need to create, update, manipulate, test, and process Data Tables. Because tablesONLINE is a table-driven system itself, it can be customized to build a wide variety of applications. tablesONLINE helps companies to:

- control program complexity
- reduce the workloads of development and maintenance personnel

tablesONLINE is available for CICS TS, and a simplified version is available for TSO/ISPF

tablesONLINE provides facilities to create column definitions for each row in a Data Table. These definitions, called Views, are managed by tableBASE and used by tablesONLINE to control the display and editing of Data Tables.

tablesONLINE handles data entry and table editing tasks and provides the access controls and data validation services essential to such tasks. Two tablesONLINE components, the menu system and the table editor, form the framework for application development.

The Menu System

The tablesONLINE menu system provides a user interface to tablesONLINE functions and for any other processes that are conveniently controlled from a menu-driven interface (Figure 6-1 is a sample of a menu). It provides a general framework for managing any application that can be adequately controlled by choices among sequences of independent actions.

A menu selection can initiate any of the following responses:

- display a subordinate menu to perform 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 TS transaction
- load and execute any CICS TS 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 could have a tailored starting menu, or groups of users could all share the same starting menu.

It is possible to offer no menu choices to some users, and have them go directly to the table editor from tablesONLINE initialization. For other users, a single menu may be appropriate. For example, a personnel clerk might choose among the following selections:

- edit employee table
- edit classification/pay rate tables
- run payroll process
- print reports

The Table Editor

The table editor is 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, providing unlimited potential to solve problems relative to editing and displaying data.

Using tablesONLINE

The menu that appears immediately after logging into tablesONLINE is determined by authority level, as set by the tableBASE administrator. One of three screens will appear:

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

tablesONLINE System Administrator

The **Administrator** screen (see Figure 6-1) contains selections for tablesONLINE access and how they are used. Selecting Option D—DEVELOP APPLICATION—opens the **Application Developer's Menu** (see Figure 6-2) where applications can be built easily. Screens are defined for end-user applications simply by editing tableBASE tables. Once a user application is completely defined to the system, it can be made available to end users by updating the appropriate security tables.

```
tablesONLINE 6.0.1 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 6-1: Administrator Screen


```
tablesONLINE 6.0.1 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 6-2: Application Developers Menu

tablesONLINE Application Developers

Screens can be built for end users without requiring detailed programming or knowledge of the system's I/O protocol. To build screens, simply edit tableBASE tables.

From the **Application Developer's Menu** select option D—DEFINE TABLE— to access the **Define Table and View** screen, as shown in Figure 6-3.

Options 1–7 of the **Define Table and View Screen** provide all the features necessary to set up the environment for end users. Help tables, PF keys, library access authority, and the commands available to end users are defined by filling in the blanks on the screens that appear for each option on the main menu.


```
tablesONLINE 6.0.1 Administrator ------ ----- Define Table and View -----------
COMMAND ===>


To select, enter number/symbol on command line:

D    DEFINE ALL           - Define All Table Elements (View and Data)
P    PRINT VIEW           - Submit a Batch Job to Print a View
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 6-3: Define Table and View Screen

Build a Table

Select Option D—DEFINE ALL—from the **Define Table and View Screen** (see Figure 6-3) to define a table.

Option D—DEFINE ALL— is equivalent to options 1, 2, and 3 combined. These three options prompt for all the parameters necessary to define a table. Key strokes are saved by executing these options in order—the system calculates some of the required values from previously specified information. For example, row size is calculated from the sum of all field sizes.

Option 4—-EDIT DISPLAY ORDER—allows you to change the display order of data.

Option 5—BROWSE VIEW—allows you to browse the different table definitions you have just created.

Option 6—CREATE ALTERNATE—allows you to create alternate definitions of the same table to give multiple Views of one table.

Option 7—RESTRUCTURE TABLE—enables you to restructure a Data Table.

Option 8—GENERATE COPYBOOK—generates a COBOL copybook.

Option 9—DEFINE M2M (many-to-many)—provides access to the M2M table where you can define the many-to-many table relationships.

Utilities

tablesONLINE provides all the utilities necessary for managing tables (see Figure 6-4). Select Option U—UTILITIES—from the **Application Developer's menu** to access the **Utility menu**, as shown in Figure 6-4.

```
tablesONLINE 6.0.1 Administrator ------ Utility menu -------------------------
COMMAND ===>


To select, enter number/symbol on command line:

P    PRINT TABLE          - Submit a Batch Job to Print contents of a table
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 6-4: Utility menu

Any selection from the **Utility menu** (see Figure 6-4) will display subordinate screen(s) that prompt for additional information.

tablesONLINE End User's Menu

The End User's menu is entirely customizable (see Figure 6-5). The application developer determines the functions that are available on this screen. End-user screens can vary from user to user, or for groups of users. This simplifies end-user tasks and provides 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    MONTHLY             - 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 6-5: End user's menu

Functions can be added to or deleted from an end-user menu, so additional functions can be provided to end users as they become more experienced.

Modifying tablesONLINE

tablesONLINE is a table-driven system that can be extensively reconfigured by editing the tables that control it. Reconfiguration options are:

- 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 other operations appropriate to the applications
- controlling access to tablesONLINE or other programs
- controlling access to tableBASE table data or other data

Extensions to Editing via User Exits

tablesONLINE provides an interface to user-written exit programs that 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 require 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 handle any data that fits logically into tabular format, like data in VSAM files or database records. With such extensions, tablesONLINE becomes a powerful data entry and data maintenance tool, even for applications that 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 data processing resources, machine power, and staff time, without sacrificing user convenience.

tablesONLINE for Data Entry

tablesONLINE can 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 that describe the data expected. These Views are created by the application developer.

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

- automatic validation whenever the field is edited
- automatic translation between the stored form of the data and the display representation
- labelling of data with the field name from the View
- data validation
- the application of user-specified display attributes to data whenever it is displayed

The following is a list of comprehensive data validation capabilities (checking field values against View data definitions) 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 data validation requirements can be addressed by these built-in features, but for those special cases, exit programs can provide additional, customer-tailored validation.

Other Special Features of tablesONLINE

tablesONLINE incorporates many features (see Table 6-1) to make it convenient for you to work with tables:

Table 6-1: tableONLINE Special Features

Feature	Convenience
Context sensitive help	This help feature can be customized for local usage
Scrollable menus and tables	Users can scroll up, down, left, and right
Freeze keys	Freeze keys allow users to fix key fields on the screen while scrolling up, down, left, or right on other fields of data
COBOL code generator	This feature generates COBOL field definitions from table descriptions
Customizable PF keys	The PF keys can be customized for local usage
Extensive data conversion and validation routines	Assists with application development
Windowing	Allows multiple sessions at one time

7 System Specifications

Hardware	IBM mainframe
Operating System	z/OS
Compatible environments	TSO/ISPF, CICS TS, IMS TM, DB2 SPAS
Accessible From	All common languages using standard IBM protocol, including: C, C++, COBOL, PL/1, Assembler, Fortran
Library Requirements	The space required for a tableBASE library with default block size 3120 is approximately the sum of: • eight blocks • one block for every 20 tables • one block for each generation of each table • enough space for each table's data, including the number of rows multiplied by row size (in bytes), plus the index size (8 bytes for every row) • free space equivalent to the largest table

