# tableBASE

# Concepts and Facilities Guide

## Release 6.0.3 / Release 6.1.0

Document Number: TBM009-R6.0.36.1.0.v2.0



Publication History

Maintenance Release 6.0.1 - May 2004
Service Pack 2 Release 6.0.1 Version 1.1 - November 2004
tableBASE Release 6.0.2 Version 1.0 - May 2005
tableBASE Release 6.0.2 Version 1.1 - March 2006
tableBASE Release 6.0.3 Preliminary - February 2009
tableBASE Release 6.0.3 Version 1.0 - April 2009

tableBASE Release 6.0.3 / Release 6.1.0 Version 2.0 - February 2010

Technical Support Hotline: (613) 523-5588

DataKinetics Ltd.
202 - 2460 Lancaster Road
Ottawa, ON
Canada K1B 4S5

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

Facsimile: (613) 523-5533

Email: tableBASE@dkl.com

http://www.dkl.com

# Table of contents



 

 

 

 

 

 

# Preface

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

## Who this Guide is For

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 tableBASE VTS, tableBASE Process Manager 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.

## 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.

The following terms are used 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 2 Gigabyte 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. A VTS-TSR is a shared Data Space that provides a public location to share tables among application regions. |
| **tableBASE VTS** | The optional component 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 TSRs are managed by the VTS Agent program. If tableBASE Process Manager is installed, VTS TSRs are managed by VTS Managers, and the VTS Agent is not required but still available for transition purposes. |
| **tableBASE Process Manager** | The optional component that extends the functionality of the tableBASE VTS shared TSR capabilities. |
| **TPM** | tableBASE Process Manager—the tableBASE Process Manager top tier component. |
| **VTS Manager** | The middle tier of tableBASE Process Manager which manages VTS-TSRs. |
| **TPVM** | This is synonymous to a VTS Manager. It is the middle tier of tableBASE Process Manager which manages VTS-TSRs. |
| **Catalog** | A catalog contains definitions of managed items in the next tier down in the hierarchy; i.e., TPVM definition information is contained in the TPM catalog, and VTS-TSR definition information is contained in the TPVM catalog. The catalog is contained within the LDS associated with the TPM / TPVM. |
| **Cataloged VTS** | A VTS-TSR that is managed by a user-defined VTS Manager under tableBASE Process Manager, as opposed to the *compat* VTS Manager. |
| **LDS** | Linear DataSet used for tableBASE Process Manager. |
| **Alias name** | An alternate name for a VTS-TSR that can be created, assigned, and used in lieu of the name assigned at VTS-TSR definition. |

 

| | |
|---|---|
| **VTS switch** | The action of switching an alias name from one VTS-TSR to another. This is a feature of tableBASE Process Manager. |
| ***compat* VTS Manager** | The *compat* VTS Manager is the default VTS Manager that runs under tableBASE Process Manager. |

## 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 tableBASE VTS product to enhance tableBASE.

Chapter 6 details the system specifications needed to use the optional tableBASE tableBASE Process Manager product to enhance tableBASE.

Chapter 7 provides an overview of the menu system and table editor of the optional tablesONLINE product.

Chapter 8 details the system specifications needed to run tableBASE.

Chapter 9 outlines the tableBASE customer support and training options available to all tableBASE users.

## 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.

## 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*

 

# 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.

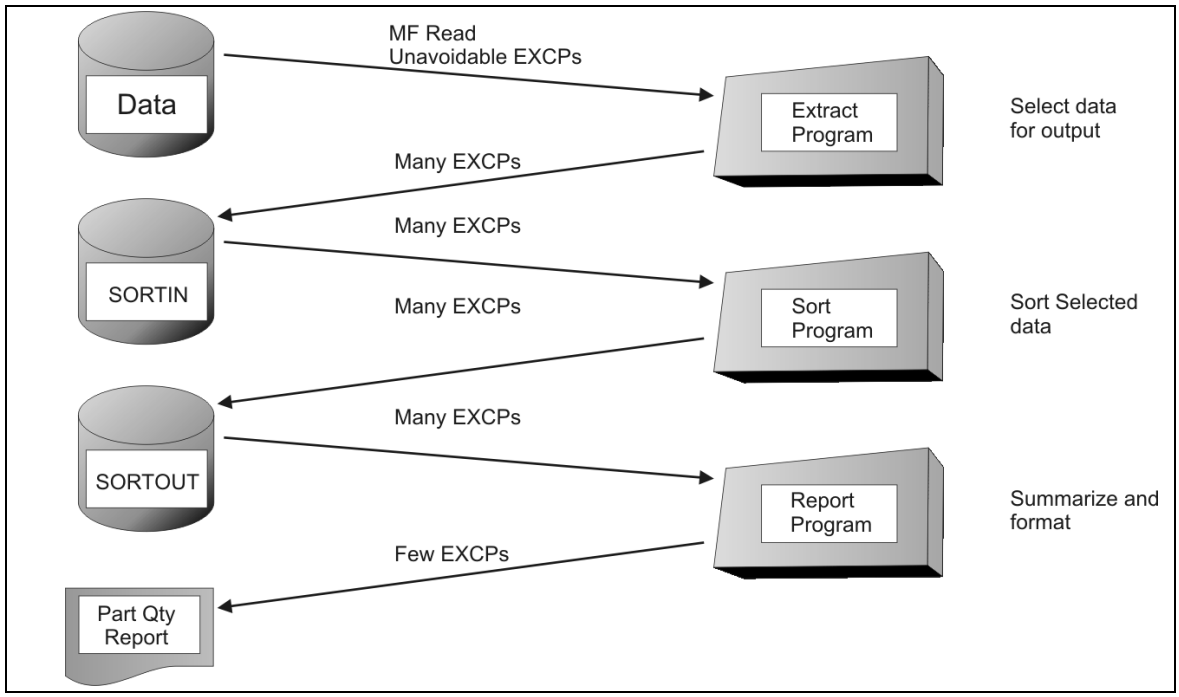


**Figure 1-1: Traditional approach used for transaction processing**

 

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.

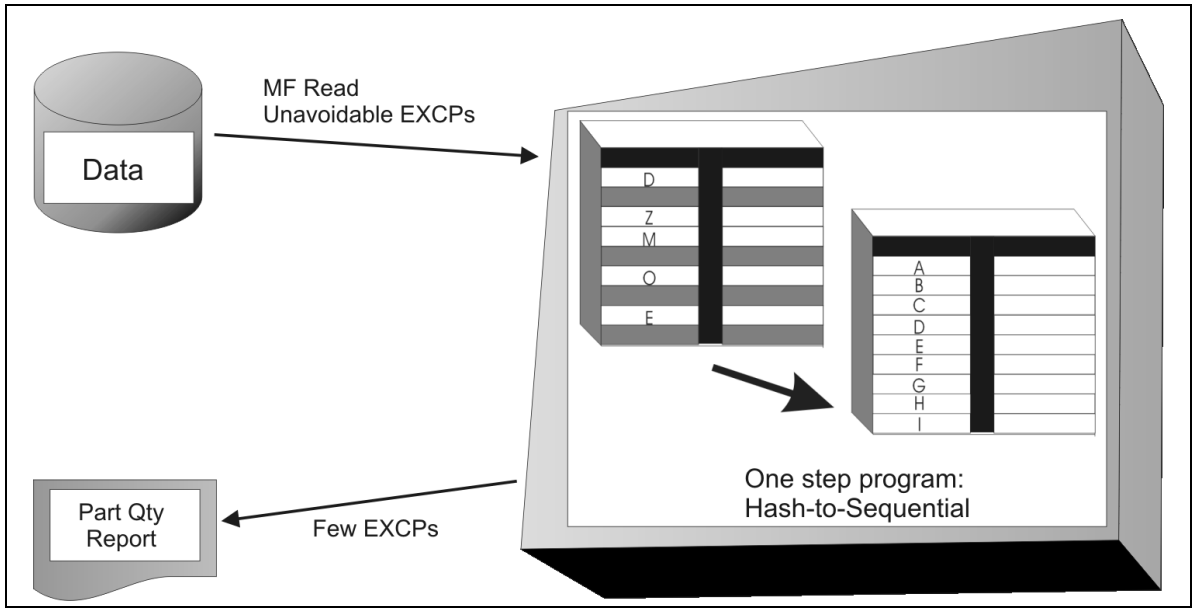


**Figure 1-2: tableBASE approach used for transaction processing**

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 DMBS 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.

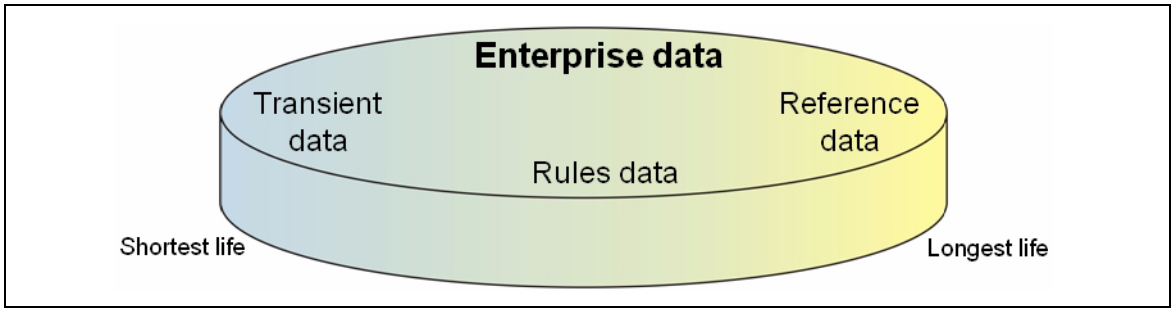


**Figure 1-3: Types of data**

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.

# tableBASE Base Product and Components

The base product is available for z/OS and consists of the tableBASE nucleus and batch interface. Additional interfaces are available; optional components for tableBASE are:

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

## CICS TS and IMS TM

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 component that 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.

## tableBASE VTS

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

The optional tableBASE VTS component allows access to shared, in-memory tables across multiple CICS or IMS regions, from both batch and online environments. tableBASE 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. tableBASE 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.

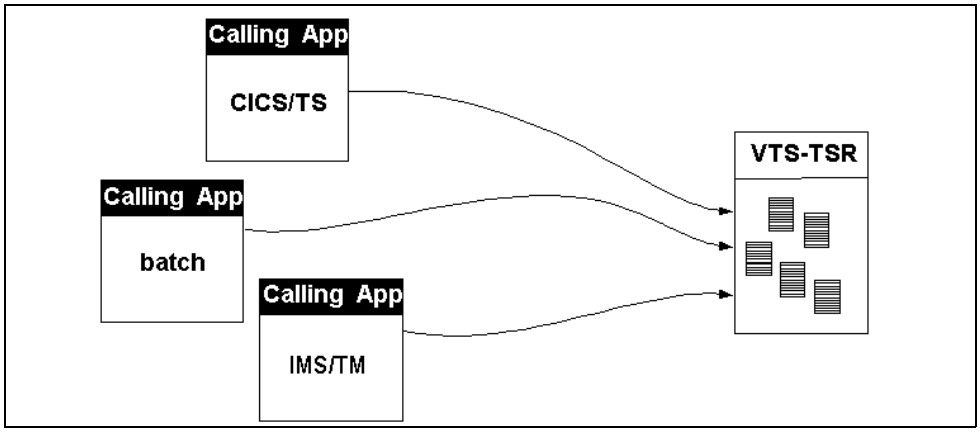


**Figure 1-4: Calling application accessing VTS table data**

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. tableBASE 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.

## tableBASE Process Manager

tableBASE Process Manager is a performance-enhancement product which augments the tableBASE VTS and tableBASE core products. tableBASE and tableBASE VTS provide the increased speed of mainframe applications and simplicity of maintenance that come with using table-driven applications. tableBASE Process Manager provides further improvements in performance, flexibility, security and manageability.

### Problems for enterprise IT

As the workload processing demands on z/OS systems continue to escalate, today's world-class enterprises keep searching for new ways to handle the skyrocketing growth in online transactions. IT professionals are faced with one or more of the following related problems:

- striving to keep up with an ever-increasing transaction load
- wanting better performance out of tableBASE and tableBASE VTS
- contemplating buying more hardware and software before planned upgrade
- needing more flexibility, control, speed, switching from test to production systems
- looking to provide increased security for sensitive data
- worrying about data integrity for VTS tables within a complex environment

 

## Solutions offered by tableBASE Process Manager

tableBASE Process Manager provides a new and higher level of data and table management. It is a three-tiered system based on the table-sharing capabilities of our tableBASE VTS product, and it has built-in features that will permit the tableBASE family of products to capitalize on the advanced multithreading capabilities inherent in today's mainframe technology.

tableBASE Process Manager incorporates Linear Data Sets (LDSs), a standard IBM technology, into its architecture, giving you the means to increase the performance, flexibility, manageability and security of your applications without ever touching a line of application code. With tableBASE Process Manager, the LDS contains an image of the shared table space regions, which means that table data is paged in as needed, eliminating entirely the initial overhead otherwise required to load tableBASE tables into memory.

tableBASE Process Manager further reduces start-up times by the use of its catalogs, where it stores all required information about the tableBASE VTS-TSRs that it managed at the time of system shutdown.

A VTS switching feature permits seamless 24/7 switching from one set of tableBASE tables to another while applications (and tableBASE VTS-TSRs) are up and running, without affecting application performance. Started transactions will complete with the tableBASE tables that are being taken offline, and new transactions will start up with the set of tables that are being brought online.

## Benefits of tableBASE Process Manager

Using the power of the features described above, tableBASE Process Manager can offer you immediate improvements to your current tableBASE / tableBASE VTS configuration in the areas of system performance, data security and enhances maintainability and flexibility.

### Performance

System performance gains are realized through the use of the base tableBASE product; further performance gains are made possible by adding the tableBASE VTS product to your configuration. Still more performance can be realized by employing tableBASE Process Manager. tableBASE Process Manager improves your system's performance by:

- eliminating the need to open tables and rebuild indexes after system shutdown
- allowing you to update libraries without taking down your production application
- increasing transaction throughput with shared read-only tables, eliminating enqueues and table locks where not required
- reducing start-up time after system termination with a warm start
- synchronizing live updates to active tables

 

**Manageability**

The tableBASE and tableBASE VTS products provide maintenance tools that allow you to manage your table-based data. tableBASE Process Manager improves your system's manageability by permitting:

- on-demand and/or pre-scheduled table data updates
- parallel testing capabilities to safely prove enhancements before going to production
- application-appropriate data access via read-only or read/write table space regions
- ability to change shared table data without changing the shared table name used by the application

**Management Flexibility**

The tableBASE and tableBASE VTS products provide some flexibility, but tableBASE Process Manager takes management flexibility to a new level, allowing you to:

- share LDSs, and therefore data, across LPARs
- run more than one maintenance level of tableBASE (e.g., production, test, Q/A) on the same LPAR
- define multiple VTS-TSRs under a single management hierarchy
- designate individual VTS-TSRs as read-only or read/write
- refresh table data content without interrupting the running transaction or application
- define, start up, shut down or delete table space regions and tableBASE Process Managers as operational demands require

**Security**

The use of the tableBASE and tableBASE VTS products provides enhanced performance and some data management capabilities, but do not offer much in terms of securing your table-based data. tableBASE Process Manager improves your system's security by employing:

- standard IBM resources such as RACF to secure access to LDSs
- read-only table spaces to add another layer of protection to corporate data
- segregation of update applications to specific read/write table space regions

 

# tableBASE History and Road Map for the Future

tableBASE is a mainframe programming tool developed by DataKinetics Ltd. It provides the reliability that you demand from a mainframe tool serving mission-critical applications. New releases of tableBASE always provide backwards compatibility with at least two previous releases. Planned releases for tableBASE involve the addition of new features and functionality, and keep tableBASE current with new hardware and software technology.

An optional maintenance agreement is available to all tableBASE customers. Maintenance customers have free access to updates, customer support, and access to the Customer Care section of the DataKinetics Ltd. corporate Web site.

DataKinetics Ltd. prides itself on excellent customer support. Friendly, helpful support staff is available 24x7. DataKinetics Ltd. believes in working with our customers to provide the solutions they need. DataKinetics Ltd. is also available to develop custom solutions, assist with optimizing tableBASE, or provide instructor-led or computer-based tableBASE training.

The DataKinetics Ltd. road map for the future includes plans for growth in the market, driving product improvements and new product offerings to help support mainframe users well into the future.

 

# 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 Virtual Table Share (VTS) component 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-TSR 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.

 

A function called tableSPACE can be invoked to limit and manage the total amount of space to 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 tableBASE VTS optional application. A shared TRS 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.

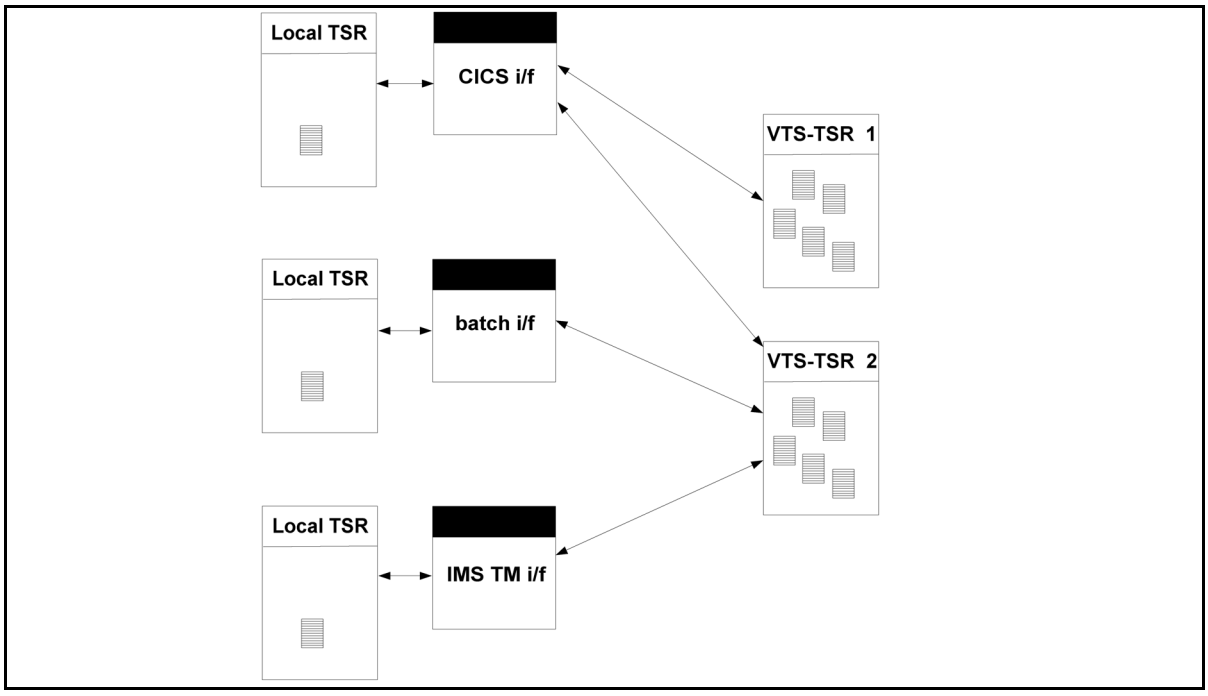


**Figure 2-1: Local TSR and Updateable VTS-TSR**

### Read-only VTS-TSR

The VTS-TSR can be Read-Only when using the tableBASE Process Manager optional component. 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.

 

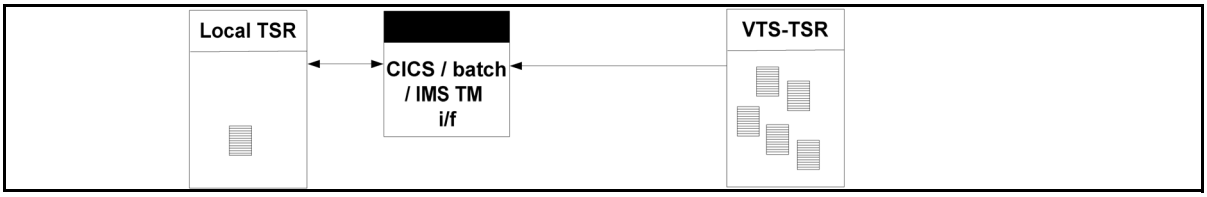


**Figure 2-2: Read-only VTS-TSR**

## 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
- Address Tree 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
- Address Tree 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.

### Address Tree Binary Search

The address tree 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 address tree 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 | Address Tree 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.

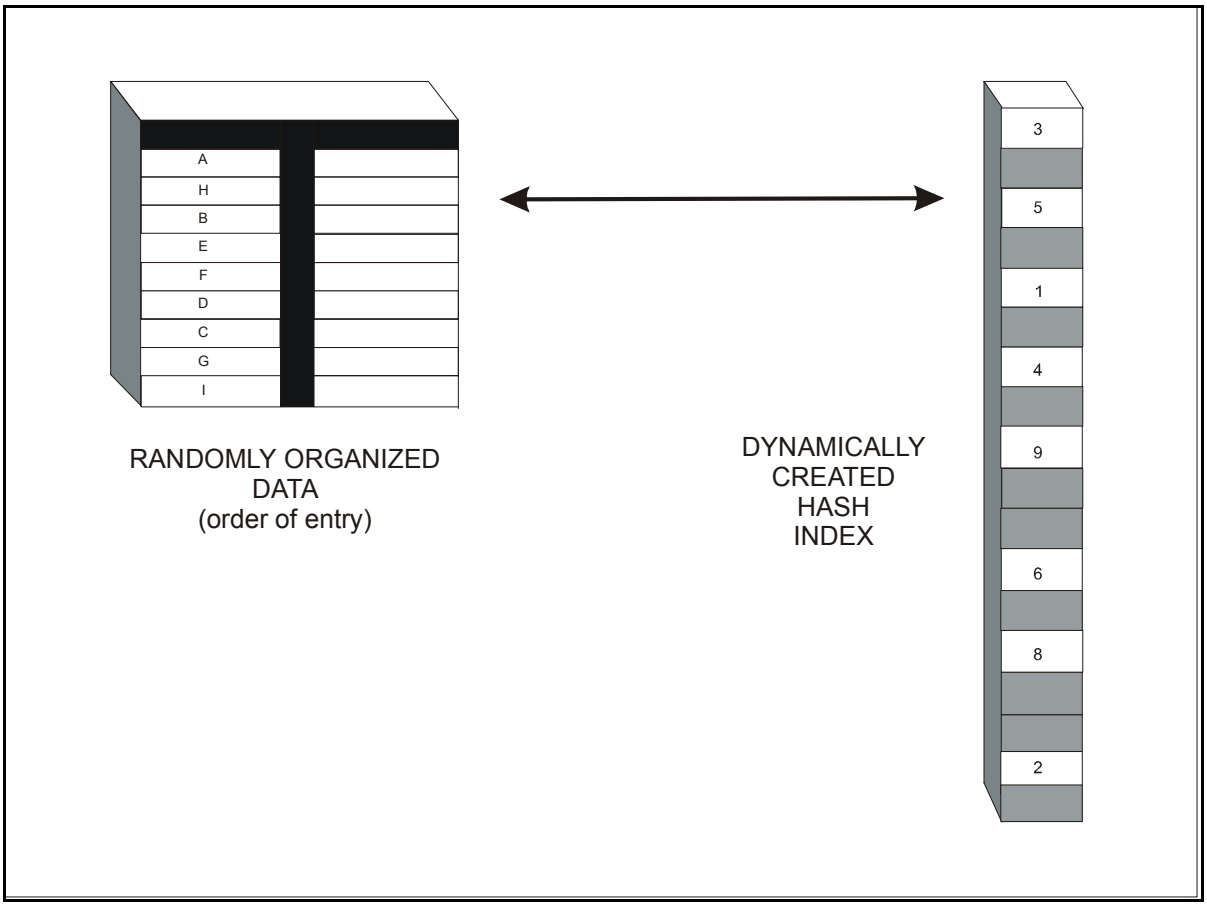


**Figure 2-3: Dynamically Created Indexes Save Space and Process Faster**

 

## 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.

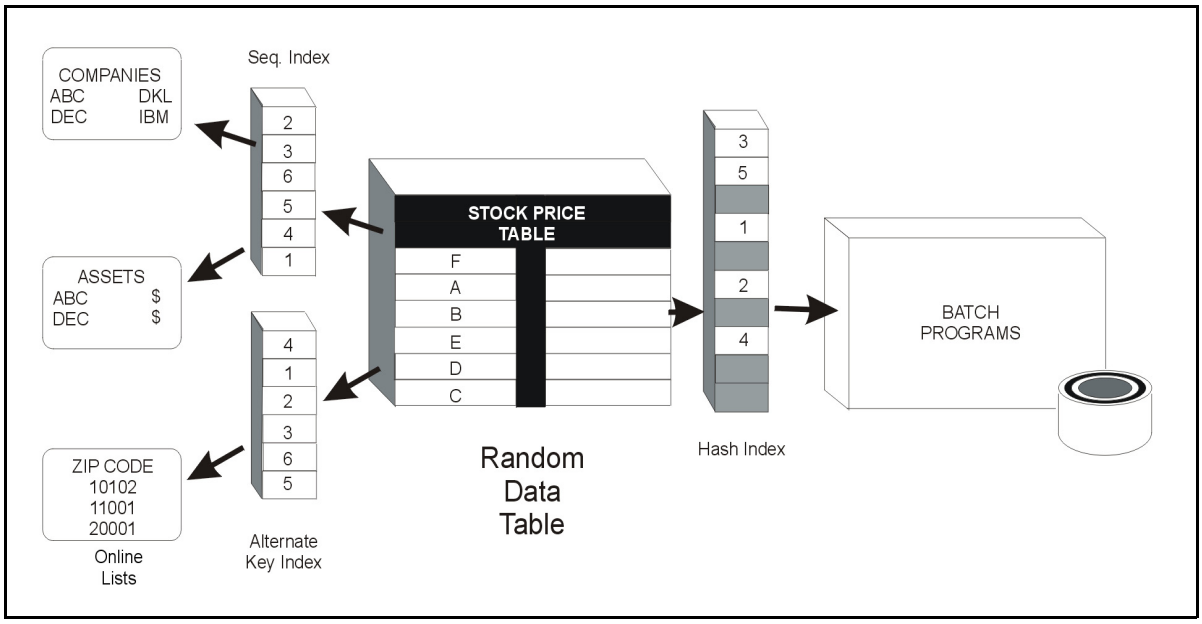


**Figure 2-4: Alternate Indexes**

# 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.

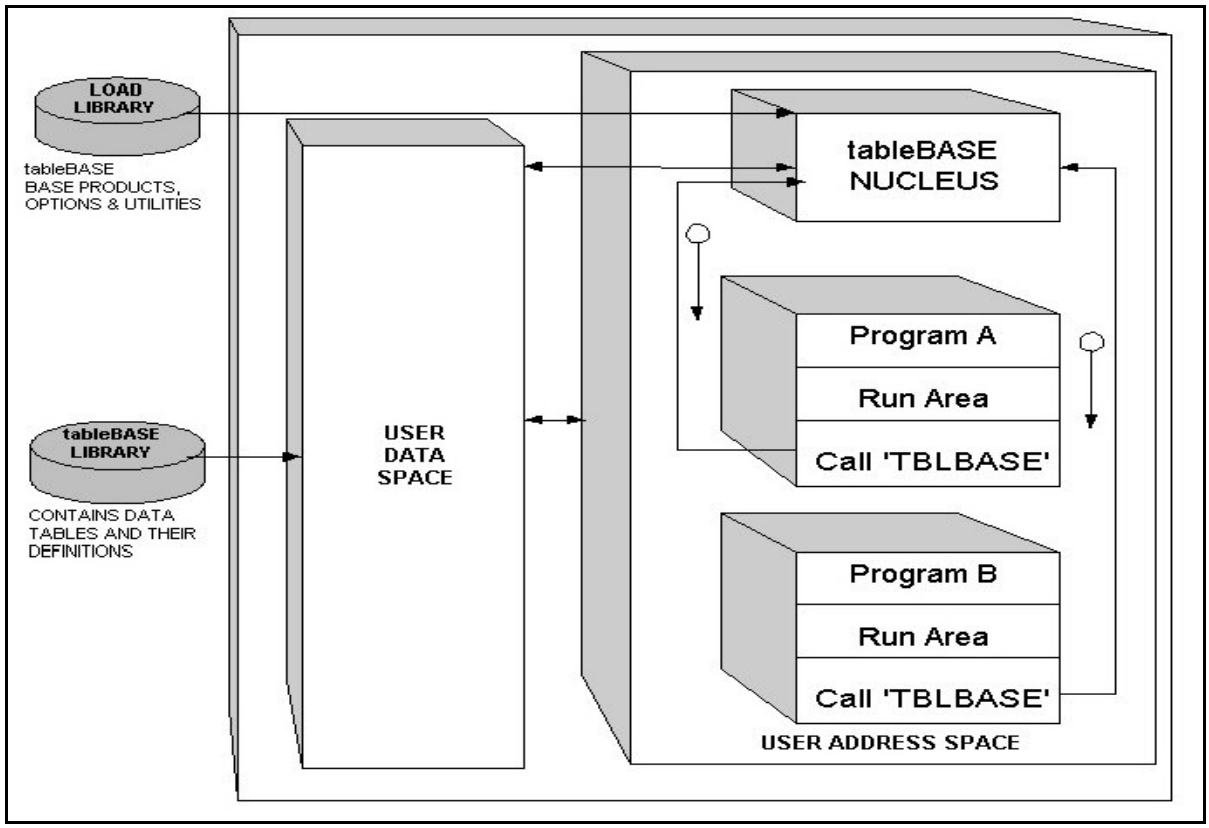


**Figure 2-5: Accessing Tables**

 

# 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.

 

# tableBASE Process Manager

tableBASE Process Manager provides a multi-tiered VTS management structure that makes VTS data management efficient, flexible and effective. Running on z/OS, the structure permits multiple independent VTS management environments, each controlled by a dedicated VTS Manager. The environments are managed simultaneously on the same LPAR.

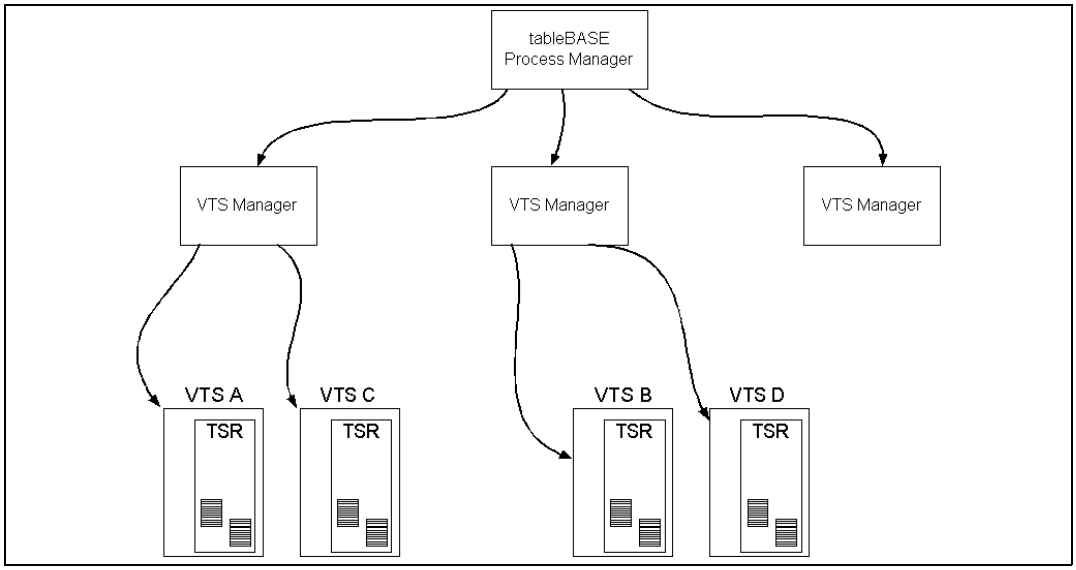


**Figure 2-6: tableBASE Process Manager: VTS management structure**

As shown in Figure 2-6, the tableBASE Process Manager itself manages one or more VTS Managers. The VTS Managers, in turn, manage individual VTS-TSRs.

## tableBASE Process Manager components

The tableBASE Process Manager product consists of three main component types: the tableBASE Process Manager, the VTS Managers, the individual VTS-TSRs.

- **tableBASE Process Manager component**: This is the top-tier module, responsible for the management of the second tier. It is the first component initialized, and is the system-wide management component on a given LPAR.
- **VTS Manager component**: This is the second-tier component, responsible for the management of one or more individual VTS-TSRs. There is one default VTS Manager (called *compat*), and there can be up to 14 user-definable VTS Managers.
- **VTS-TSRs**: These are exactly the same VTS-TSRs that were used in tableBASE Release 6.0.3. The only differences are that they are now managed by VTS Managers (rather than the old VTS Agent), and can be loaded either from a tableBASE library, or from a VSAM LDS (see "Pre-loaded VTS-TSRs" on page 43). There can be up to 64 user-defined VTS-TSRs associated with a VTS Manager.

 

# tableBASE Process Manager features

## VTS initialization enhancement

### Pre-tableBASE Release 6.1.0 VTS initialization

Between tableBASE Release 5.0.2, and Release 6.0.3, a VTS-TSR is managed by a program called the VTS Agent. This program's function is to wait for maintenance commands such as "shut down."

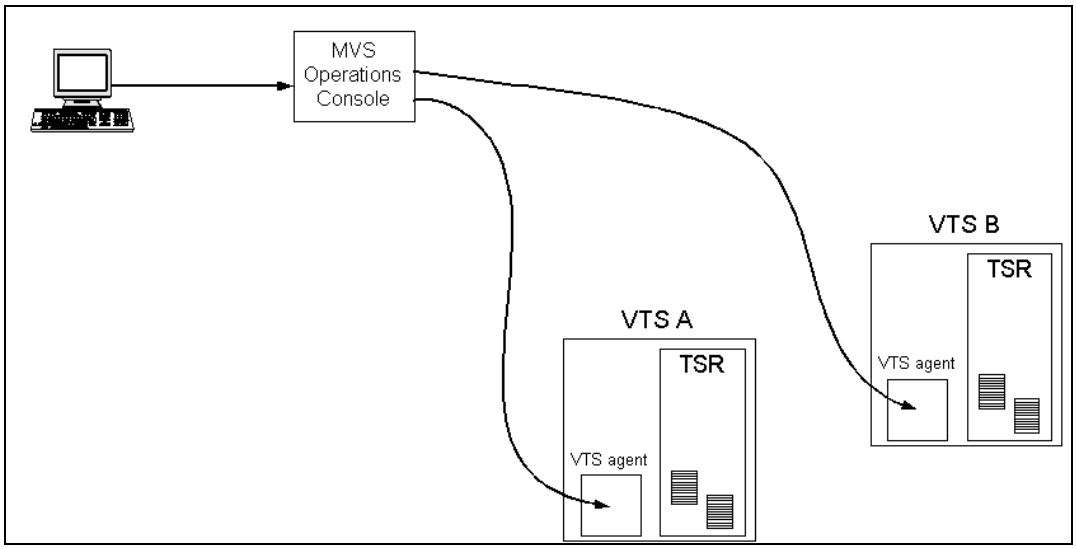


**Figure 2-7: User interaction with the VTS Agent**

The user interaction through the MVS operations console (or from a TSO session) with the VTS Agent is the mechanism by which a VTS TSR is managed. Typically, a VTS Agent is started using a JCL script, and stopped by an operator. Operational personnel must do this manually.

In the scenario shown in Figure 2-7, there could be business reasons to start VTS A every day at 8 AM, and to shut it down at 5 PM; and to start VTS B at 7 PM and shut it down at 4 AM. To accomplish this, a mainframe operator must be available to implement these actions, or each activity would have to be scheduled independently.

After shutting down a VTS-TSR, there is no way to return to the last used state of the VTS-TSR at shutdown. When you start a VTS-TSR up again, data is rebuilt within the VTS-TSR row-by-row from library-based data, and index(es) are rebuilt as required.

### Improved VTS initialization

tableBASE Process Manager, being fully compatible with standard MVS scheduling systems, introduces a new level of flexibility to the VTS startup and shutdown processes. VTS-TSRs can be scheduled for startup and shutdown in groups, and at different times. This eliminates the need to startup and shutdown VTS-TSRs individually; however, the capability is still retained.

 

When a VTS-TSR is started up using tableBASE Process Manager, the data within is accessed from the corresponding mapped LDS—the result is that VTS data is available immediately, and in the exact state it was when the VTS-TSR LDS image was created.

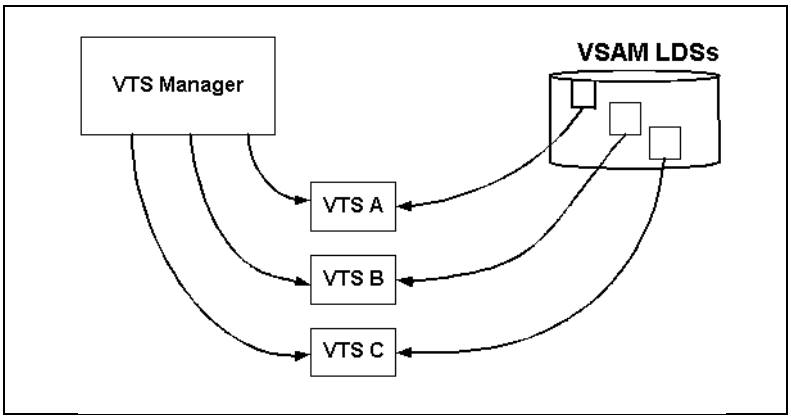


**Figure 2-8: Improved initialization**

The scenario is improved using tableBASE Process Manager—The VTS-TSRs, under the management of a VTS Manager can be started / shut down together or independently. Startup and shut-down is now a job handled by a new utility (See "Using tableBASE Process Manager" in the *tableBASE Batch Utilities Guide*). On startup, VTS-TSR data spaces are loaded immediately with the latest data via their optional links to the LDS. (For more on this, see "Pre-loaded VTS-TSRs" on page 43.)

## Pre-loaded VTS-TSRs

Using tableBASE Process Manager, VTS-TSR tables can be mapped to Linear Data Sets (LDSs). On VTS-TSR initialization, the link to the LDS is set up, and data is available immediately to calling applications.

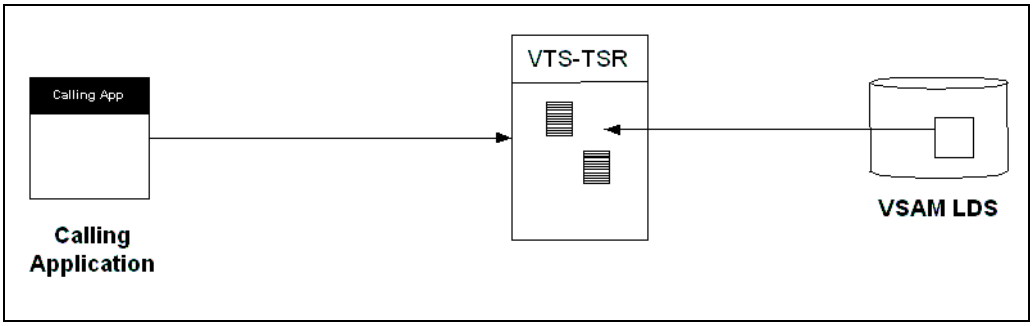


**Figure 2-9: VTS-TSR mapping to VSAM LDS**

The use of LDSs in this way improves performance by eliminating the time previously required at VTS-TSR startup to load table data row-by-row from the DASD-based library, and to rebuild all indexes. I/O operations are performed only as needed, using the system paging facility.

 

## Read-only VTS

It is now possible to create read-only VTS-TSRs. Read/write VTS-TSRs, useful when updating tables is a requirement, are less desirable when that data is intended for table-driven decision processing. Access time can be burdened by needless queuing and table locking—read-only VTS-TSRs eliminate the need for this type of overhead processing. Using read-only VTS access, such costly processing can be completely eliminated.

While read/write VTS-TSRs provide a certain amount of flexibility, they are not the best choice for use across multiple LPARs. It would be difficult, or impossible to manage changes to a single LDS mapped to a VTS-TSR that appears on multiple LPARs. However, a read-only VTS-TSR can easily be managed across multiple LPARs.

Once loaded, no changes to the definition, index(es) or data contents will be allowed; the update commands are disabled. The mapped LDS will not be updated.

Note that some features (such as "VTS initialization enhancement" on page 42) are available only for read-only VTS-TSRs.

## Change control improvements

### Legacy change control workflow

Change control within installations using tableBASE Release 6.0.3 or earlier, rely on simulation and approximations in the update and test phases. Working in the same LPAR as the production environment, it is not possible for the update/test environment to use the same naming conventions for VTS-TSRs. For this reason, custom test software must be used to test the new VTS-TSR-based data.

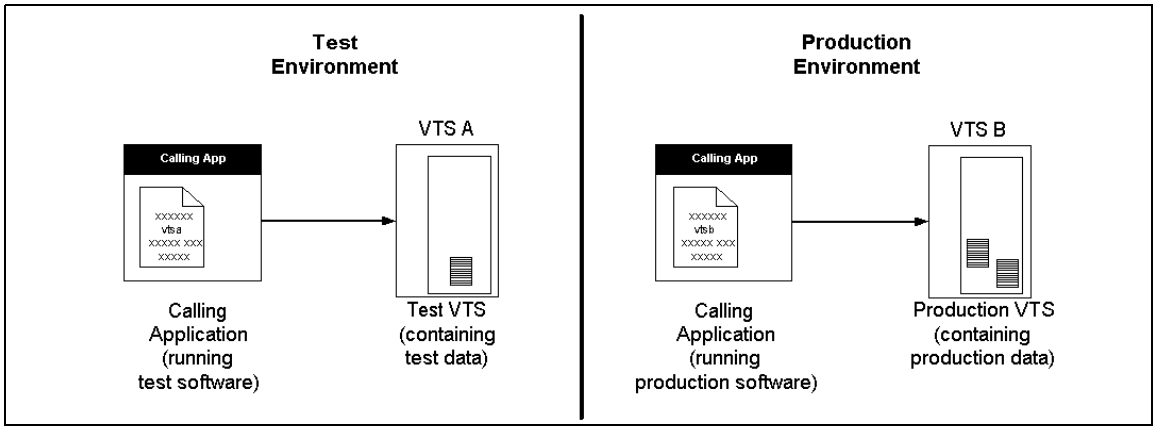


**Figure 2-10: Change control using simulations**

### Improved change control workflow

Using tableBASE Release 6.1.0 and tableBASE Process Manager, you can use the same naming conventions in the test/update environment as those used in the production

 

environment. Multiple VTS Managers provide multiple independent VTS-TSR environments within a single LPAR. In this way, different VTS-TSRs can have the same name, as long as they are controlled by different VTS Managers, making it possible for you to use your VTS names and production software in the testing phase as well.

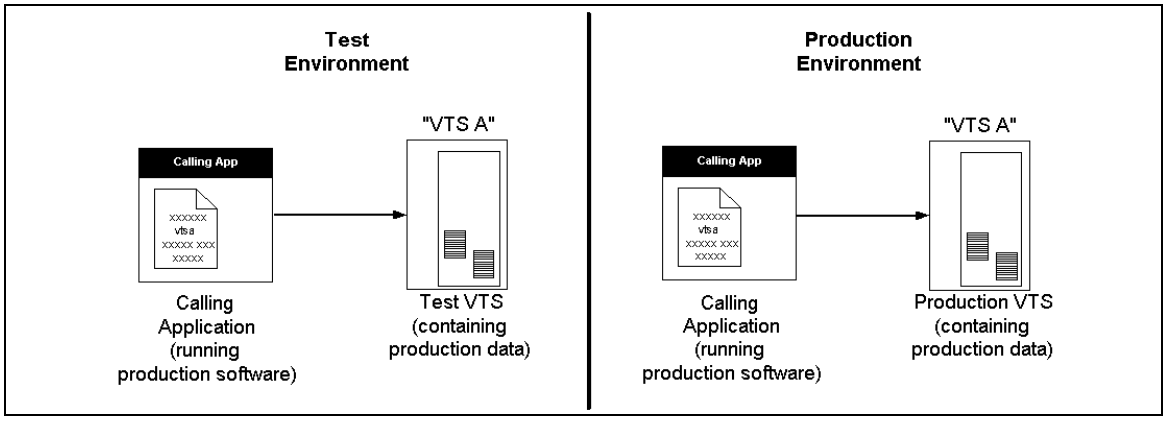


**Figure 2-11: Change control using proper naming conventions and data**

Rather than relying on simulations, your change control can be much more reliable, and will stand up much better to scrutiny.

Integral to this feature are "Update tables in the background" (see below), "Alias names for VTS-TSRs" (see page 47) and "Switching VTS-TSRs" (see page 46).

### Update tables in the background

Updating VTS table data using tableBASE Release 6.0.3 or earlier can pose some challenges. First, if data updates have to be performed during periods of heavy VTS table use, calling applications will experience significant delay, as table locking will be in effect during the update.

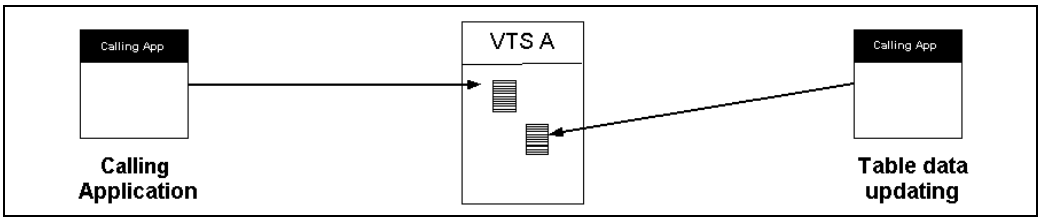


**Figure 2-12: Problems with simultaneous read access and updating**

If the VTS-TSR is shut down to implement table data updating, the calling applications will also experience delay, since, when the VTS-TSR is started up again, table data must be rebuilt row-by-row from libraries.

Using tableBASE Process Manager, VTS-TSRs can be updated in the background without affecting the performance of calling applications, and without the delays

 

associated with VTS startup. In effect, you can synchronize live updates to your active tables.

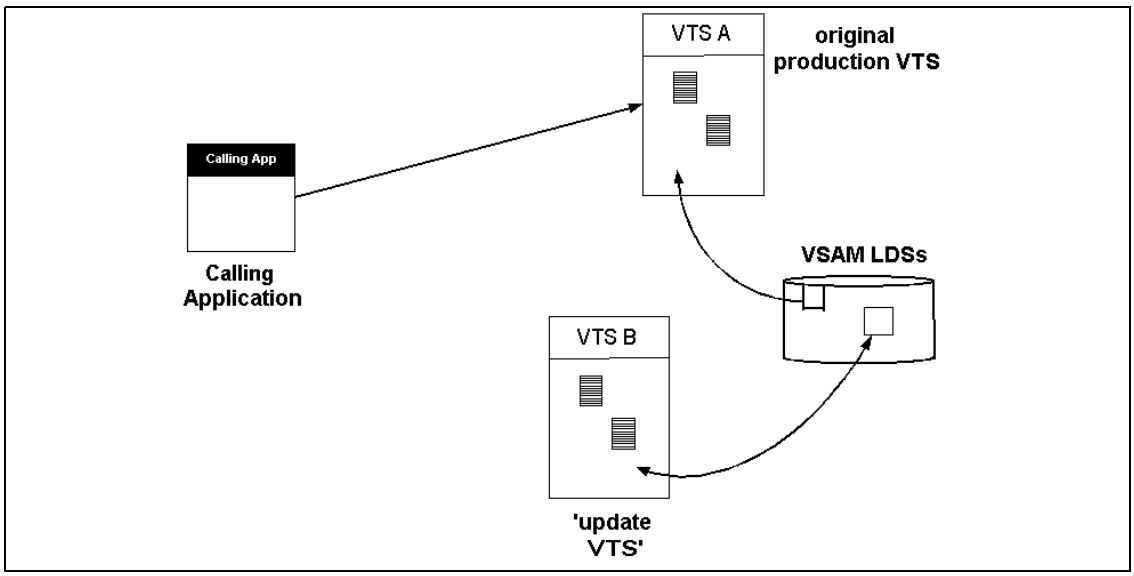


**Figure 2-13: Data updates managed in the background**

An "update VTS" can be created with a different name (VTS B), mapped to a different LDS. Access to the production VTS (VTS A) is not affected.

### Switching VTS-TSRs

When background updates are complete, the VTS-TSR containing the original data (VTS A) is replaced by the new VTS-TSR containing the updated data (VTS B). Calling applications (batch programs, transactions, etc.) still accessing VTS A will be allowed to complete their tasks. New tasks will access the updated data (VTS B).

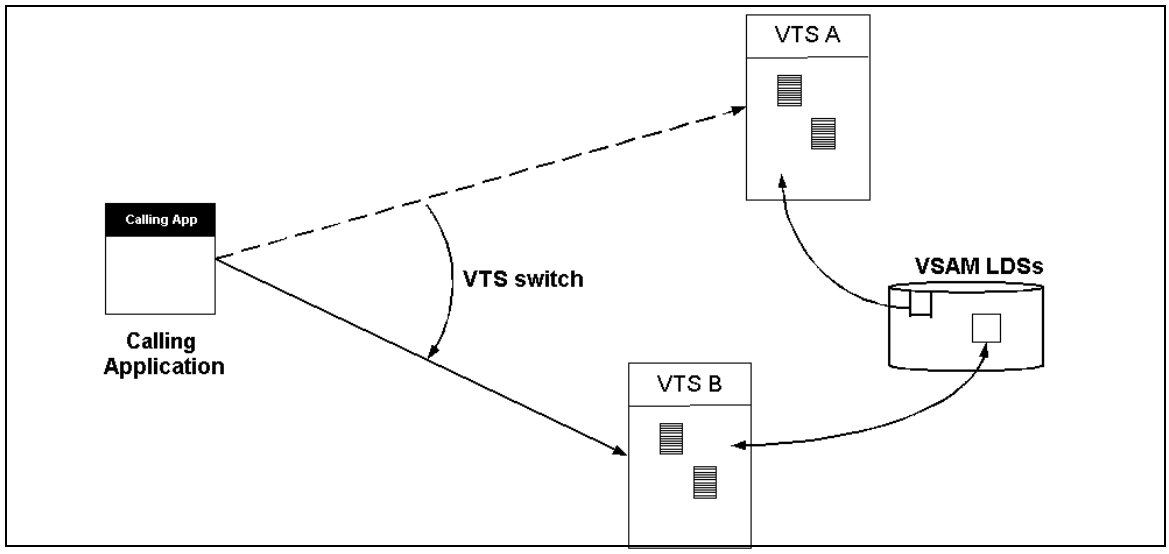


**Figure 2-14: Switching from old data to updated data**

 

The switch actually switches an alias name from one VTS-TSR to another; it determines which VTS-TSR is accessed by new tasks. It does not start or stop the VTS-TSR.

The switching of VTS-TSRs can be scheduled to execute at a specific time and date. This allows for immediate updating of data to align with external business triggers.

### Alias names for VTS-TSRs

A critical feature which supports the "Change control improvements" feature is alias names for VTS-TSRs. Calling applications will access VTS-TSRs using an alias name; the actual name of a VTS-TSR is unknown to the calling application. This allows VTS-TSRs to be switched—from an older VTS-TSR containing old data, to a new VTS-TSR containing updated data. All that is really switched is the alias. The alias name used by the calling applications is disassociated from the old VTS-TSR, and associated with the updated VTS-TSR.

See "Behind the scenes" on page 50 for information on how this and other features are made possible.

## Security enhancements

tableBASE Process Manager improves table data security in several ways:

- **VTS-TSR to LDS mapping**: Recall that tableBASE Process Manager maps VTS data to VSAM LDSs, providing a new, more secure data environment. Since LDSs are secured by standard IBM RACF services, your VTS-TSRs, by extension, are now so secured. While it is difficult to provide security for your current VTS data, installing tableBASE Process Manager will provide a level of RACF security to all of your VTS-TSRs by default.
- **Limit access to your data**: Recall also that mapping your VTS tables to LDSs makes it possible to run identical copies of VTS-TSRs across your enterprise—you can now control where and when your corporate data is used. Limiting the access to your sensitive data in this way improves your overall data security. For example, while VTS data is in use by calling applications (VTS A in Figure 2-13), mainframe support personnel can work on data updates in the background (VTS B in Figure 2-13). Only these personnel have access to the update data. Their activity has no effect on the current usage, and calling application users have no access to the update data.
- **Secure change control**: tableBASE Process Manager provides improved workflow for change control. Its multi-tiered management features allows you to create a structure that permits multiple environments to coexist on the same LPAR. Personnel responsible for table changes can be updating tables in a secure 'updating environment' (VTS B in Figure 2-13) at the same time that other users are using the existing tables in a secure "production environment" (VTS A in Figure 2-13). The same VTS names and table names can be used in all environments, reducing confusion and adding a new level of quality to the process.

 

Control and security are provided by this process. Further security comes from controlling who sees the data, and when they are permitted to see it.

When table changes are implemented and approved, you can seamlessly switch them into the production environment. This means that you can schedule table changes and control when they become available across your enterprise. The resulting level of control helps you maintain data consistency among all users.

- **Read-only VTS data**: Another level of security comes from read-only VTS-TSRs. In addition to your existing read/write VTS-TSRs, you can now restrict access to some of your shared VTS data by designating it as read-only. This is especially important if you use tableBASE for table driven decision processing. Read-only VTS-TSRs are secure in that they can not be changed—there are no update capabilities associated with them.

## Sysplex operation

Currently (tableBASE Release 6.0.3), it is possible to manage your VTS table-based data on multiple LPARs within a Sysplex, but it is a precarious management task. VTS-TSRs are read-write, meaning that it is a challenge to maintain data consistency across LPARs. With tableBASE Release 6.1.0, you can use tableBASE Process Manager to organize your read-only data on a single LPAR, multiple LPARs, or all of your LPARs in a sysplex. In this way, you can ensure that data used on different LPARs is guaranteed to be consistent.

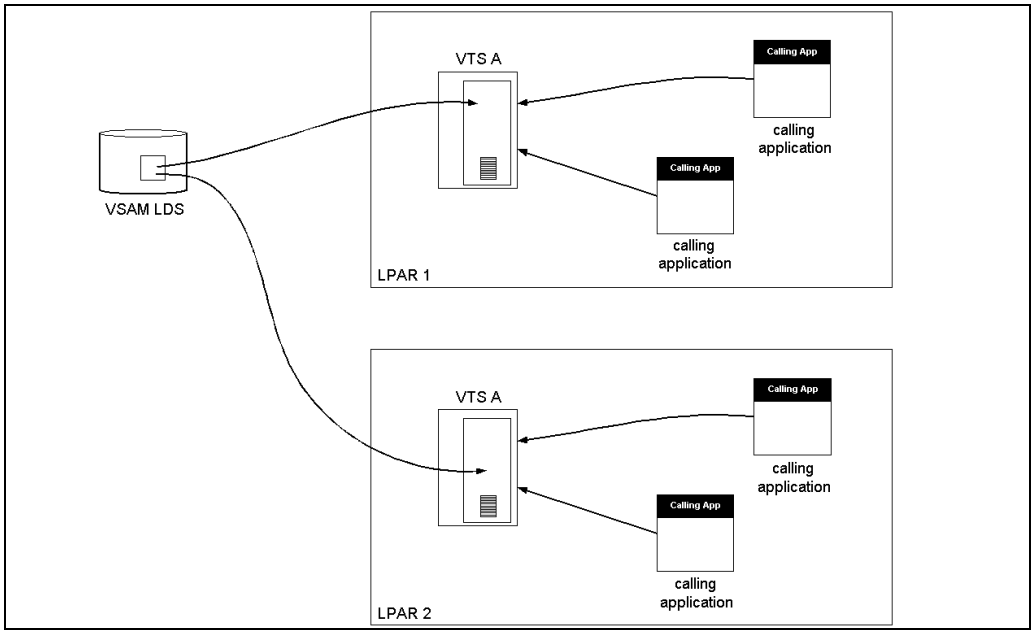


**Figure 2-15: Controlling data across LPARs**

Since a VTS is mapped to a single LDS, each instance within each LPAR is consistent with the hardened LDS. The result is data consistency across LPARs.

 

**Note:** In a multi-LPAR configuration, there will be a dedicated tableBASE Process Manager installed on each LPAR.

## Backward compatibility

A special VTS Manager, called *compat*, provides backward compatibility for VTS-TSRs created with tableBASE Release 6.0.3 or earlier. Existing VTS Agent JCL will run seamlessly, and existing VTS-TSRs will run as expected, managed by the *compat* VTS Manager.

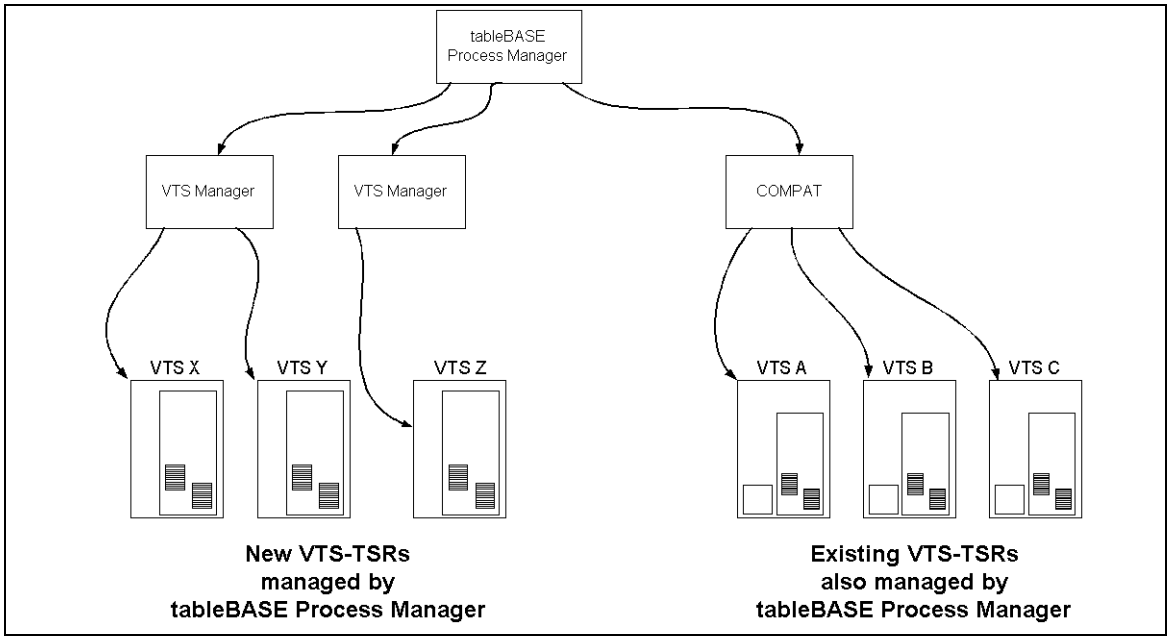


**Figure 2-16: Pre-existing VTS-TSRs maintained by tableBASE Process Manager**

You can benefit from the new, more efficient tableBASE Process Manager environment while still maintaining 100% of your current VTS data architecture. You can create new VTS-TSRs in the new environment, and gradually migrate your current data into new custom environments at a controlled pace. Or, you can leave your pre-existing VTS-TSRs as they are—they will operate as expected.

Within the tableBASE Process Manager architecture, pre-existing VTS-TSRs can be auto-started, but many of the new tableBASE Process Manager features will not apply to them.

 

## Behind the scenes

### Alias names and VTS-TSR switching

An alias name list is kept in the VTS Manager catalog. Normally, VTS-TSR names do not change—they retain the name given to at definition. Alias names are assigned to a VTS-TSR. Calling applications will use the alias name only. Alias names make possible the improved change control and switching features that are part of the tableBASE Process Manager package.

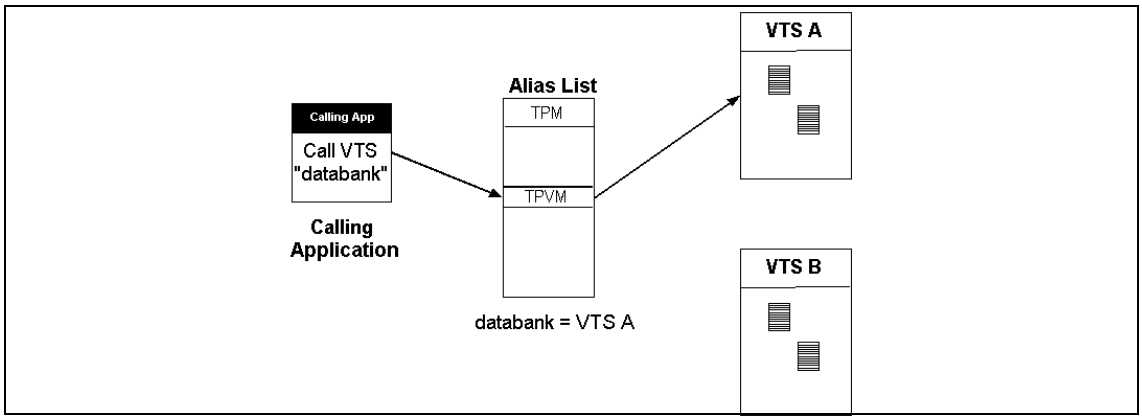


**Figure 2-17: Alias name for VTS A**

In the above scenario, the alias name *databank* has been assigned to the VTS-TSR VTS A. When the calling application makes a call to *databank*, tableBASE searches for the name within the applicable GCA structure, where a pointer exists to the actual VTS-TSR (VTS A).

When a VTS switch takes place—the alias name is unmapped from VTS A, and remapped to VTS B. The result is that the alias name *databank* now points to VTS B:

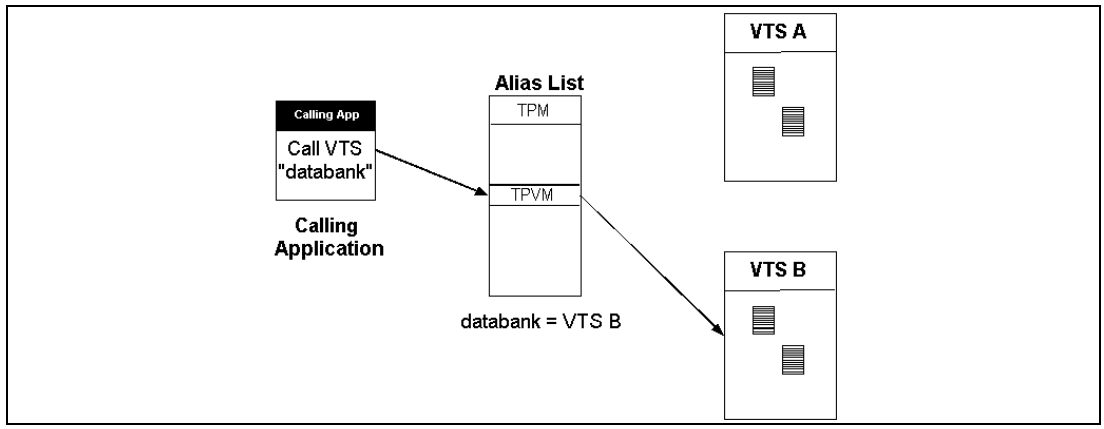


**Figure 2-18: Alias name switched**

 

From this point on, new calls to the alias name *databank* will now access VTS B. Note that applications using VTS A at the time of the switch are permitted to complete their transactions. Subsequent calls will go to VTS B.

This feature allows for changes or updates to table data without affecting program code in any way. Even replacing the data tables outright requires no change to program code.

Note that alias names are not available with the backward-compatibility VTS Manager, *compat*.

### Alias name rules

It is important to understand that the alias name acts as a pointer only. The VTS-TSR defined characteristics are properties of the VTS-TSR, and not the alias name. For example, a VTS-TSR can be defined as read-only with auto-start and auto-shutdown; these are properties of the VTS-TSR, and not of the alias name. Moreover, the VTS-TSR properties are not inherited by the alias name, nor does a switch transfer properties from one VTS-TSR to another.

The following rules apply to alias names:

- You must define an alias name before you switch to it (there is no implicit alias definition).
- An alias name must be unique—it cannot be the same as an existing VTS-TSR or another alias name.
- A new VTS-TSR name, similarly, must be unique—it too, cannot be the same as an existing VTS-TSR or another alias name.
- If a VTS-TSR is associated with an alias name, it must be shut down using the alias name, as opposed to the defined VTS name.
- An alias name can be assigned to only one VTS-TSR at a time.
- A VTS-TSR can be assigned only one alias name at a time.
- Accessing a VTS-TSR via an alias name will continue to be valid until the alias name association is changed, or until the VTS-TSR is shut down.
- If a VTS-TSR is to change from one alias association to another, two switches will be required: one switch to de-assign the first alias name association, and a second switch to apply the new alias association.
- Changing an alias from one VTS-TSR to another requires only a single switch.
- Switching an alias name from one VTS-TSR to another requires that both VTS-TSRs be up and running.
- If a VTS-TSR has an associated alias name, all tableBASE accesses must be made using the alias name; attempts to use the original defined VTS-TSR name will fail.
- Alias names and VTS-TSR switching do not apply to R/W VTS-TSRs.

 

### VTS-TSR switching process

Here is the sequence of events for a VTS-TSR switch:

- A VTS-TSR is defined (called *VTSA*, for example).
- The VTS-TSR is started.
- An alias name is defined (called *databank*, for example).
- A switch is performed to associate the alias name with the VTS-TSR.
  (in effect, you're switching from nothing, to *VTSA*).
- Users or calling applications will access *VTSA* using the name *databank*.
  (The users/calling applications need never be aware of the name *VTSA*).
- While users/calling applications are using the data within *VTSA*, another version of the data can be created separately. The containing VTS-TSR is defined and named *VTSB*.
- *VTSB* is started.
- The *databank* alias is switched from *VTSA* to *VTSB*
  (in effect, it is unassigned from one, and re-assigned to the other).
- The next transaction application that requests *databank* will actually access *VTSB*.

Note that alias name switching is not available with the backward-compatibility VTS Manager, *compat*.

 

### Catalog usage

Catalogs are used by the tableBASE Process Manager and the VTS Managers to store information about the layer below. They are mapped to LDSs, and provide instant information on startup.

The tableBASE Process Manager catalog information includes: VTS Manager information (VTS Manager names and attributes), auto-start information, LDS information and more. When the tableBASE Process Manager starts, it accesses its catalog, and acquires this information instantly.

Similarly, each VTS Manager has a dedicated catalog which contains information including: VTS names and attributes, auto start information, LDS information, and alias definitions.

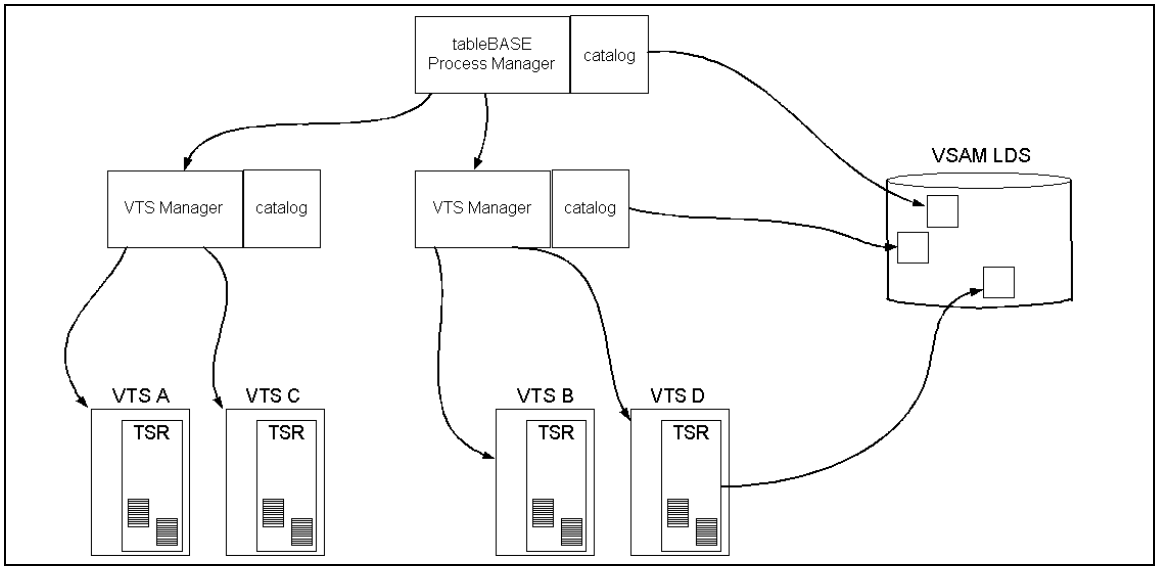


**Figure 2-19: Catalogs are mapped to LDS**

The tableBASE Process Manager catalog LDS is mapped to the data space owned by the tableBASE Process Manager address space. The tableBASE Process Manager catalog dataspace is built when the tableBASE Process Manager is initialized (see "Details of tableBASE Process Manager initialization" in the *tableBASE Administration Guide*).

#### VTS-TSRs mapped to LDSs

As shown in Figure 2-9 and above, in Figure 2-19, VTS-TSR dataspaces are also mapped to LDSs, but this is optional. However, the VTS-TSR to LDS mapping is the basis for many tableBASE Process Manager features. For read-only VTS-TSRs, the LDS must be pre-loaded, and all tables and indexes must be build prior to VTS startup.

 

#### LDS use across a SYSPLEX

Note that the tableBASE Process Manager LDS catalog can be shared across a SYSPLEX (all LPARs within a single GRS-plex). Similarly, VTS Manager LDS catalogs and VTS-TSR LDSs can be shared using read-only VTS-TSRs across multiple LPARs.

# 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 57.

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, TPDRIVER, 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.

### TPDRIVER

This is the tableBASE Process Manager Service Request Manager, which provides the capability to dynamically define, start, stop, and delete VTS-TSRs, VTS Managers and other management components. Typically, TPDRIVER will be submitted as a normal batch job and, as such, will provide final return codes.

### 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.

### DK1P1MGR

This tableBASE Process Manager module is responsible for managing both product initialization as well as product termination / recovery. It also loads other modules which are used during initialization or as part of normal processing.

### RM

The tableBASE Process Manager Request Manager (RM) works in conjunction with the TPDRIVER utility. It invokes applicable service routines on behalf of the tableBASE Process Manager or a VTS Manager, and passes back the RETURN / REASON codes.

### 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*.

## Internal terminology

When commands and JCL parameters need to make reference to the tableBASE Process Manager or a VTS Manager, special terms are used; below is a summary of these.

### TPM

TPM is a short (or code) name for the tableBASE Process Manager module. This term is used within JCL scripts and command parameters.

### TPVM

TPVM is a short (or code) name for the VTS Managers. This term is used within JCL scripts and command parameters.

### *compat*

This is the backward-compatibility VTS Manager. It is always auto-started, and is never auto-shutdown.

 

 

# 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.

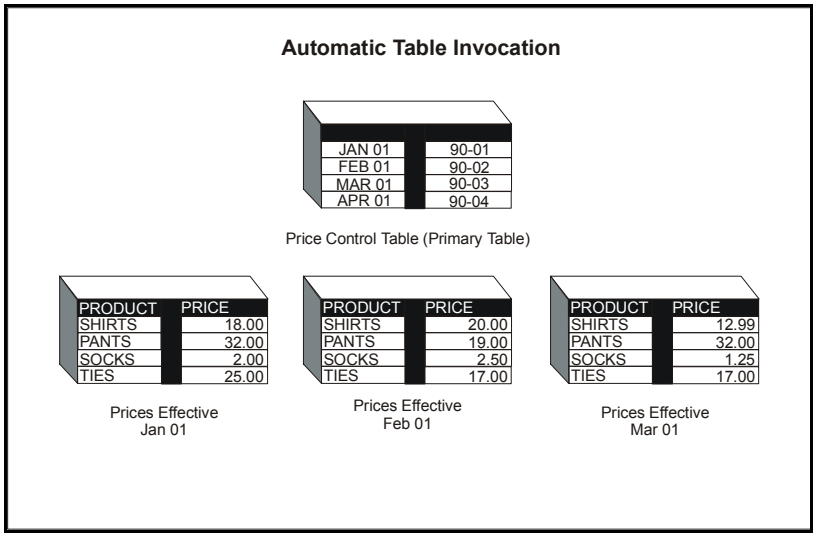


**Figure 4-1: Indirect Access to Tables for Date Sensitive Data**

## 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 tableBASE VTS

tableBASE VTS, 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 tableBASE 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 "tableBASE VTS" on page 24, with more detail under "tableSPACE Region" on page 29.

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 tableBASE 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.

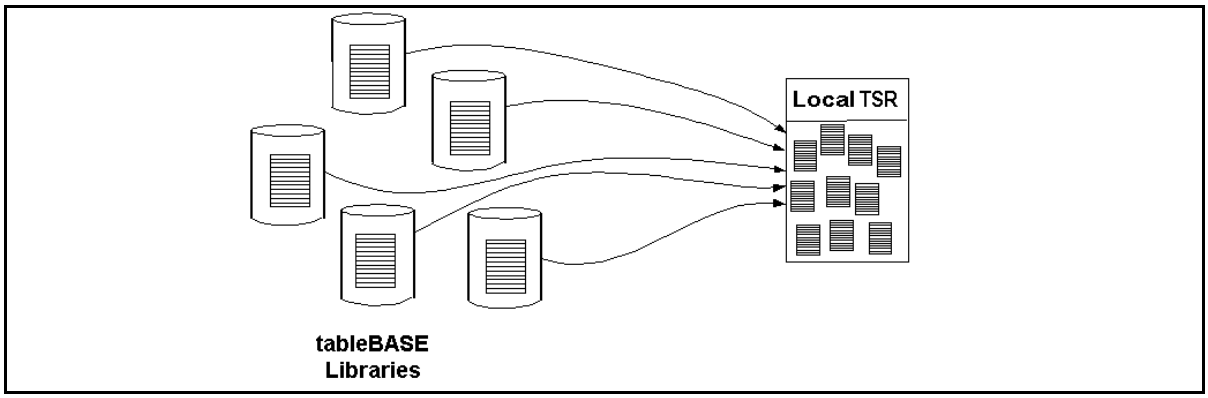


**Figure 5-1: Performance issues at initialization**

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).

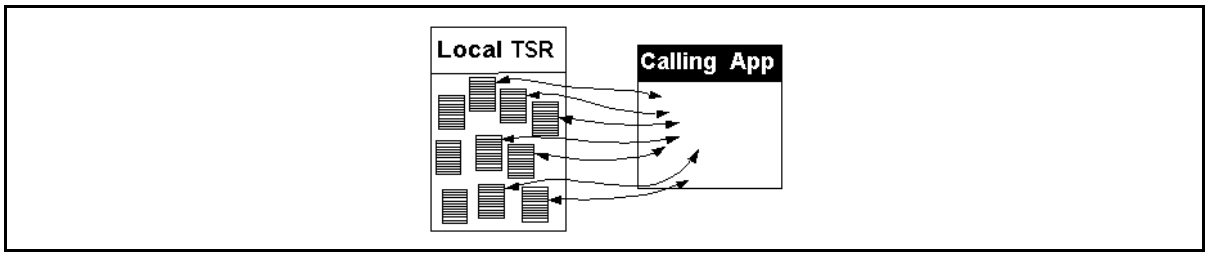


**Figure 5-2: Performance issues at run-time**

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

 

## tableBASE VTS provides the solution

The tableBASE VTS option 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).

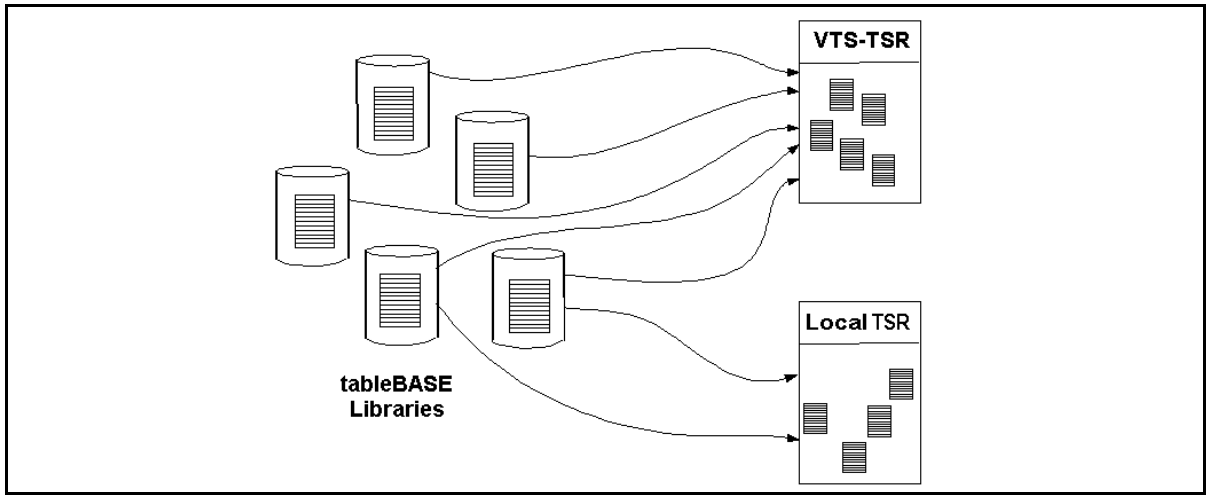


**Figure 5-3: VTS-TSR off-loads the Local TSR**

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.

 

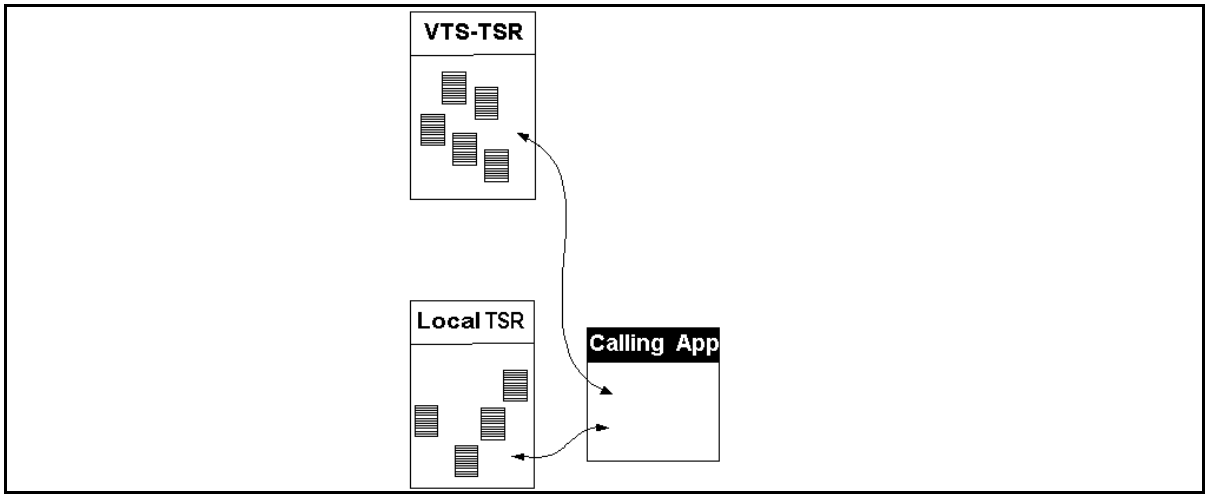


**Figure 5-4: Improved performance with fewer tables per TSR**

## More options with tableBASE 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 tableBASE 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.

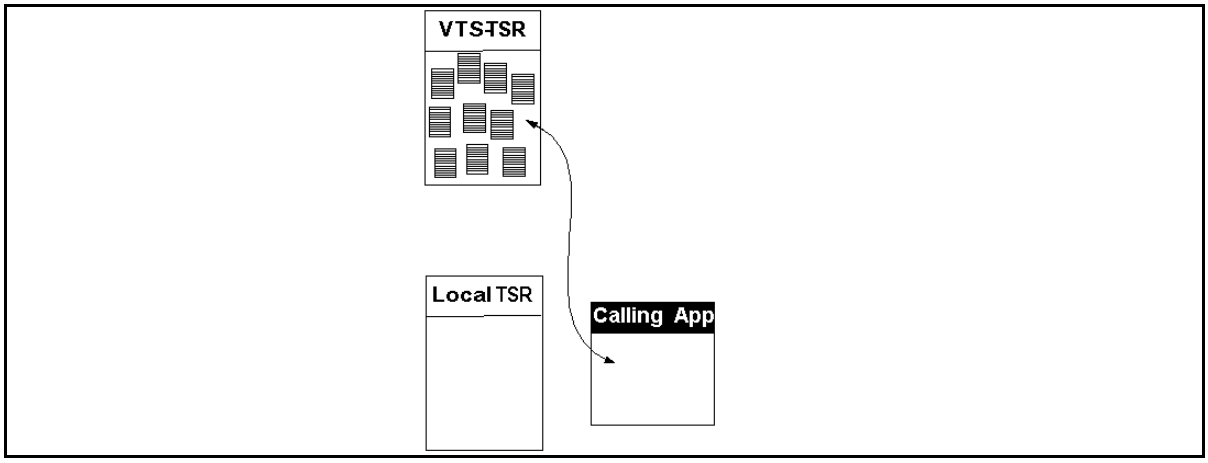


**Figure 5-5: No tableBASE data in local TSR**

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, tableBASE VTS has inherent compatibility features that allow access from multiple environments simultaneously, allowing for diversified applications.

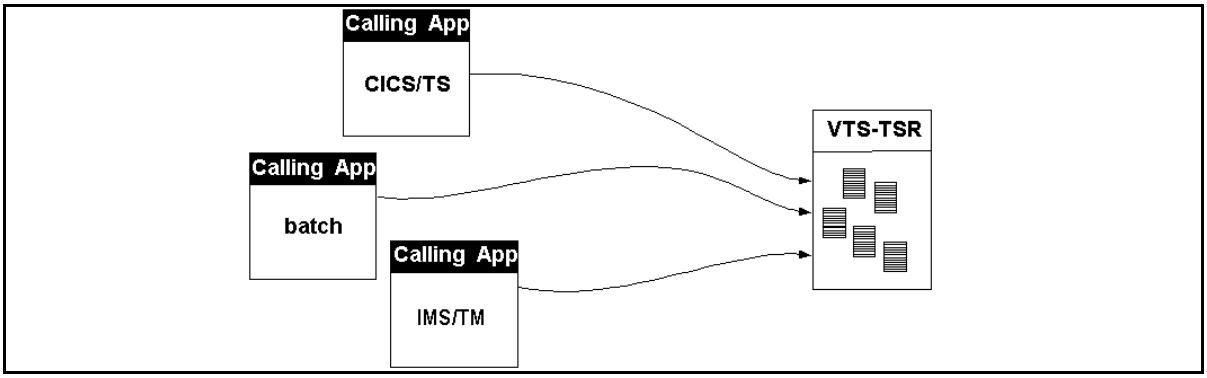


**Figure 5-6: Multiple applications accessing shared tableBASE data**

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).

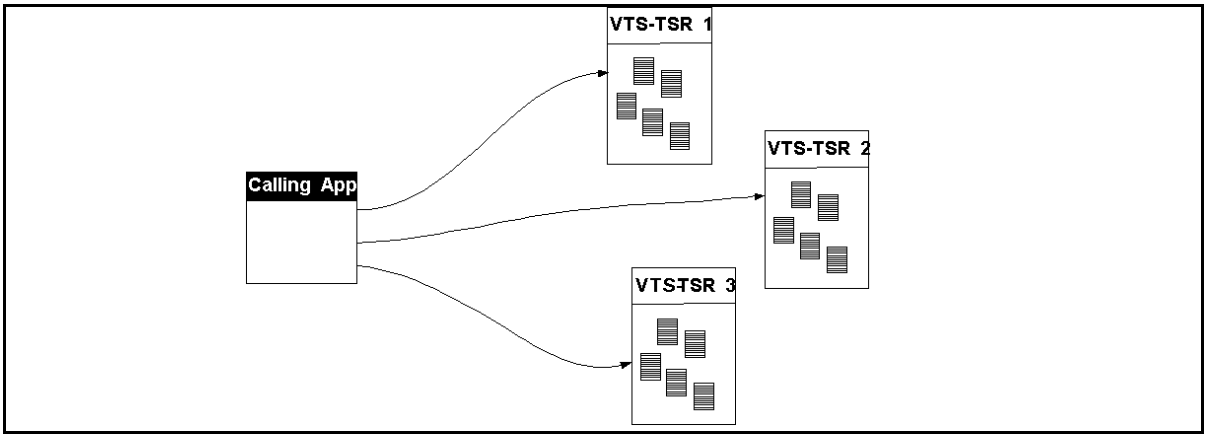


**Figure 5-7: Multiple VTS-TSRs**

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.

 

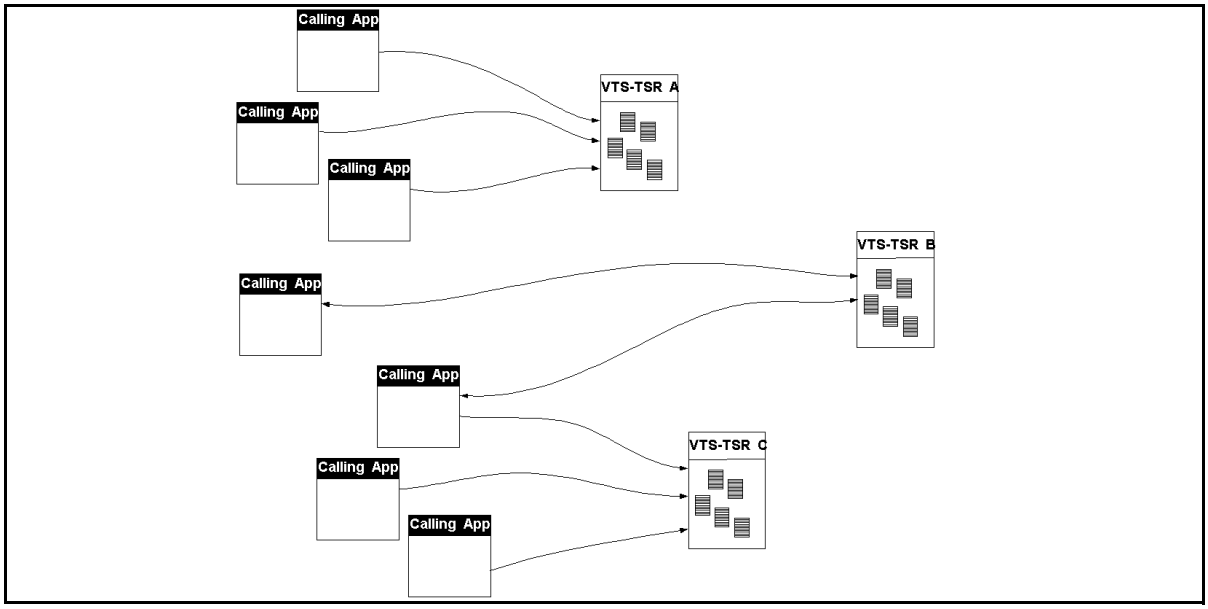


**Figure 5-8: More tableBASE VTS solutions**

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 tableBASE 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 tableBASE VTS, 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 tableBASE 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 tableBASE 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.

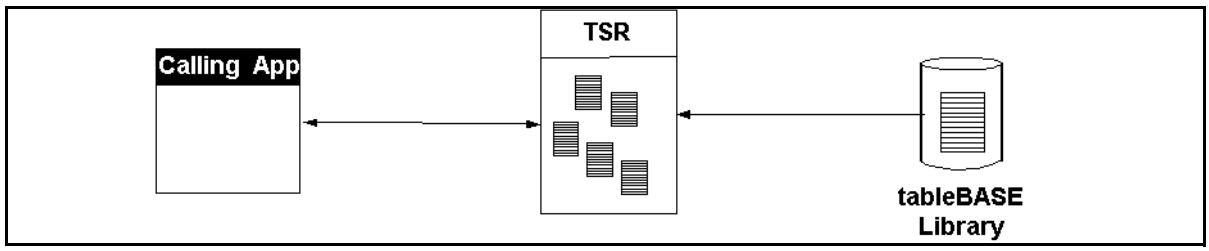


**Figure 5-9: TSR access using a single region**

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.

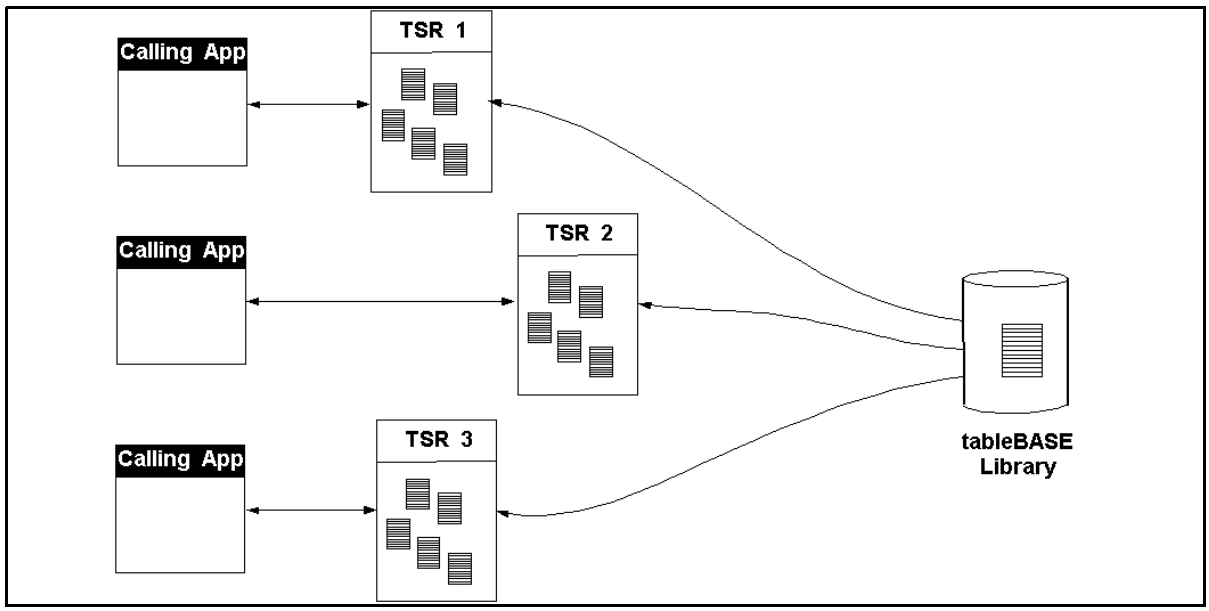


**Figure 5-10: TSR access using multiple regions**

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

 

## tableBASE VTS enhances data integrity

tableBASE 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 tableBASE VTS, all regions can access the same data from a shared TSR; a VTS-TSR.

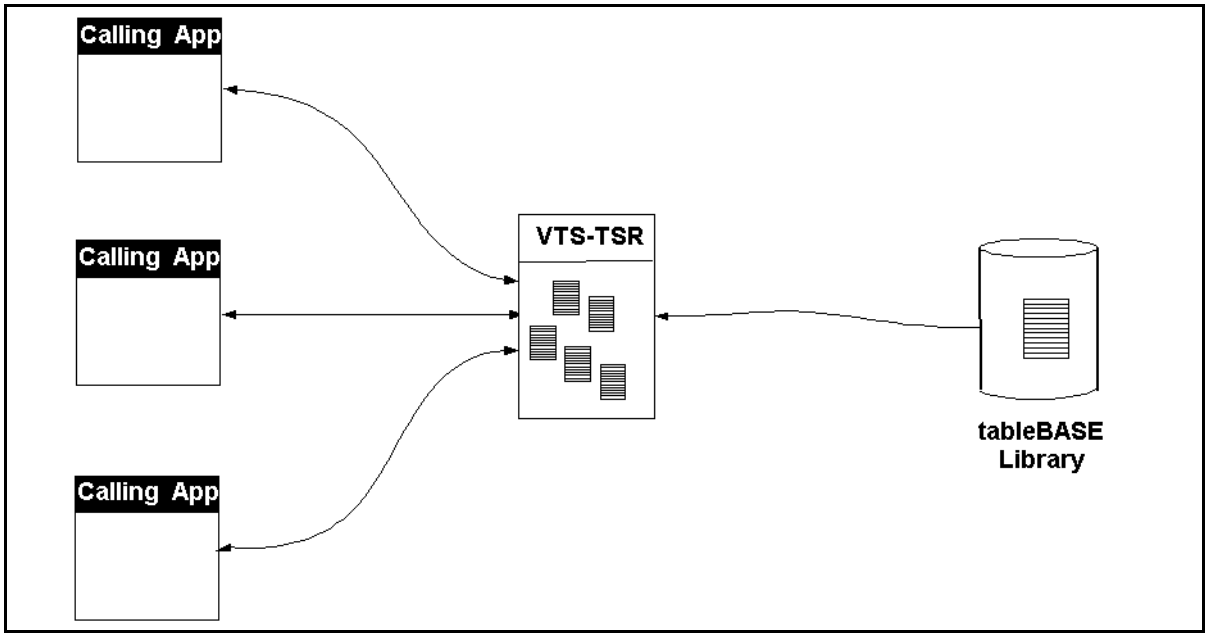


**Figure 5-11: Shared TSR access using multiple regions**

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.

 

# Message queuing using tableBASE VTS

DataKinetics provides a messaging technique that is elegant, and easy to implement. If you are looking for an efficient way to link your business processes together within your mainframe architecture, DataKinetics' tableBASE VTS provides an attractive solution.

## Legacy business process linking

IBM mainframe technology provides a tried-and-true, reliable process for integrating business processes. Typically, it consists of real-time processes running during business hours, and batch jobs running after hours.

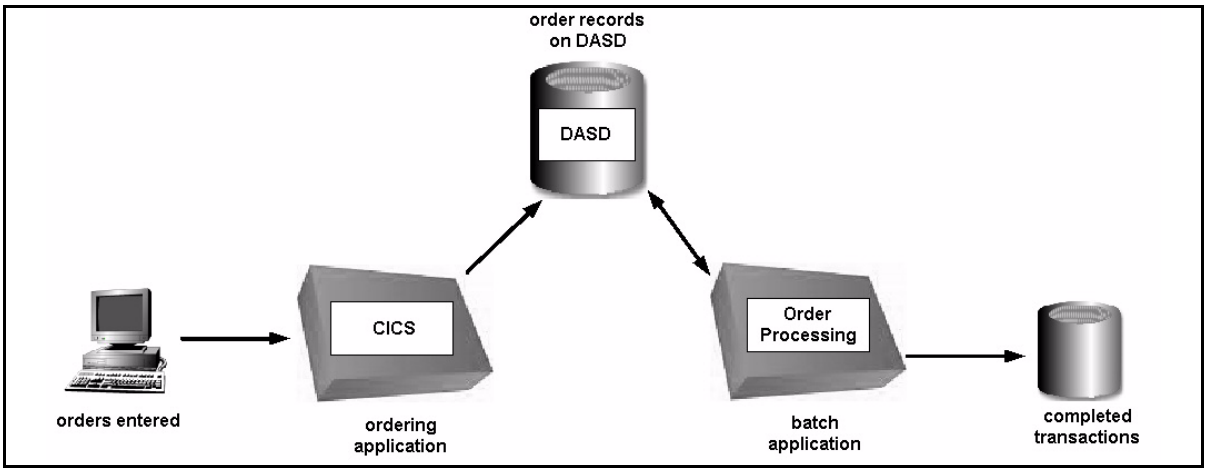


**Figure 5-12: Typical mainframe order-entry and order-processing scenario**

The image above depicts a typical mainframe-based order-taking and order-processing solution. During working hours, orders are received, then entered by users using a CICS interface. They are saved immediately as records on a DASD device. During off-hours, a batch order-processing program sources the orders from DASD, processes them, and saves completed transactions.

This solution has been used for years by many companies in many business segments including insurance sales, retail sales, finance and banking, using similar processing.

When orders increased, more order-entry users were added, and when order storage became scarce, more DASD devices were incorporated into the solution.

 

## When things get busy

While mainframe technology scales nicely in this regard, a bottle-neck occurs when the time needed to complete the off-hours batch processing begins to approach or exceed the time available. When web-based ordering comes into the mix, the situation worsens.

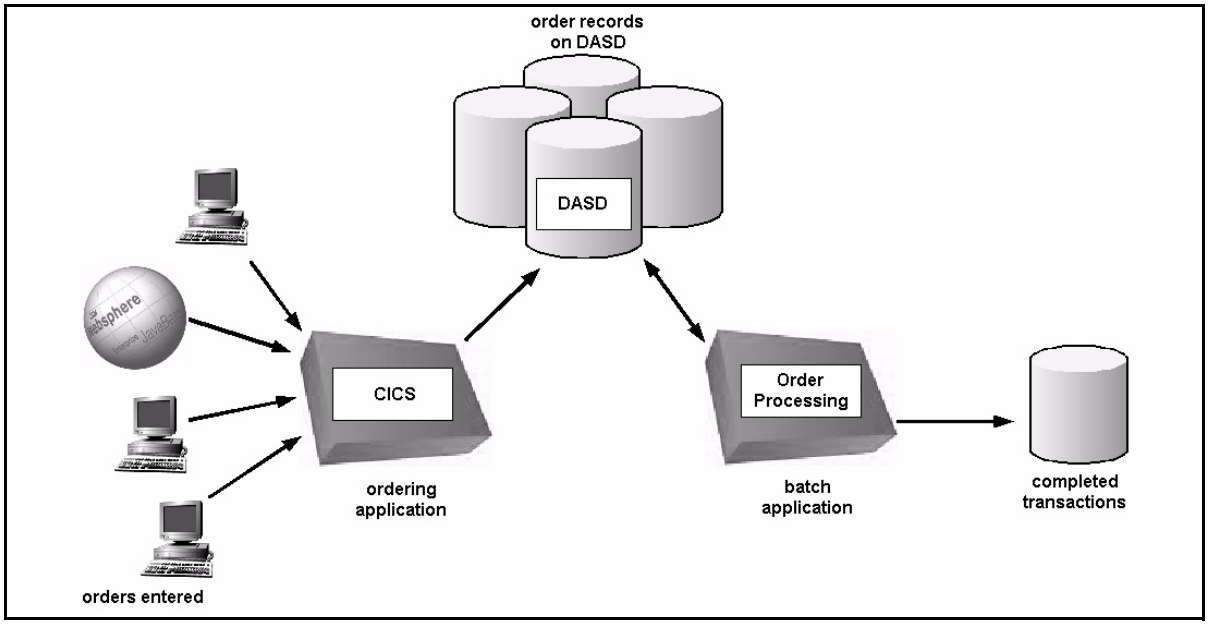


**Figure 5-13: Scaled-up mainframe order-entry and order-processing**

The above image depicts the worsening condition. Built-in IBM scalability allows you to increase the number of CICS sessions, add DASD resources, and to use multiple order-processing batch applications. More difficulties ensue, however, as it becomes tricky to synchronize the orders on multiple DASD devices, requiring additional, and sometime complex processing. Finally, the entire architecture is challenged with 24x7 web-based order input.

 

## The tableBASE solution

DataKinetics offers a simple, elegant solution to this problem, and it doesn't include massive investment in increased hardware spending. The solution is tableBASE and tableBASE VTS.

tableBASE VTS allows you to build a shared, in-memory order queue that can run 24x7.

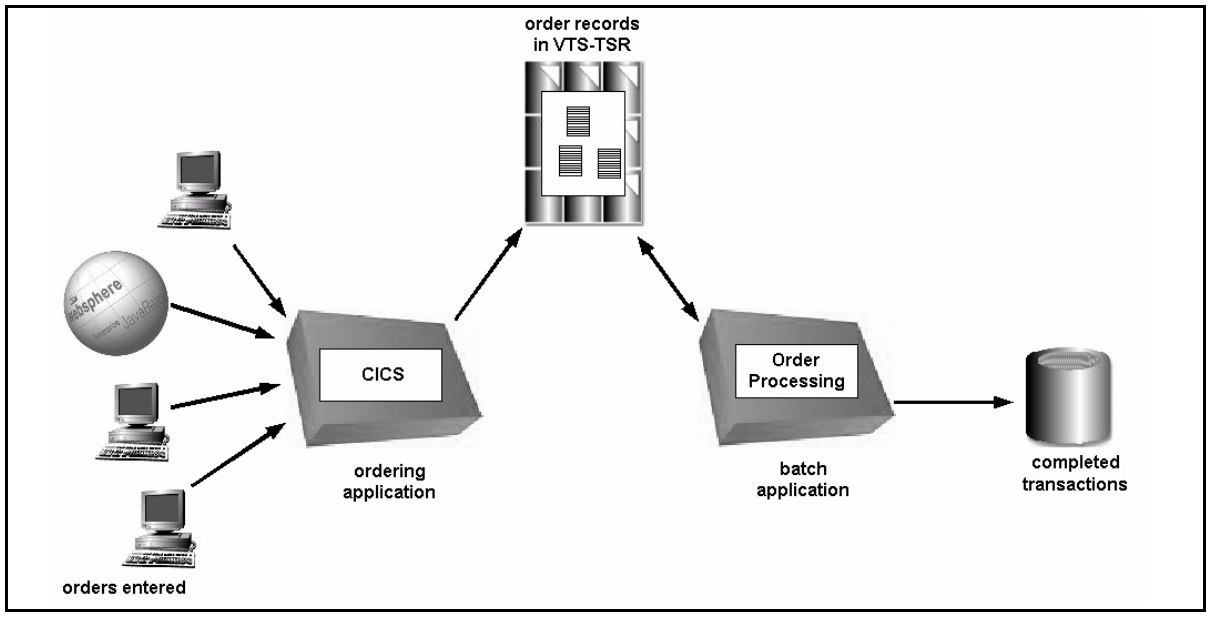


**Figure 5-14: tableBASE mainframe order-entry and order-processing solution**

Using the tableBASE order queue solution, orders can be taken by users, or from the internet 24x7. Each order is saved to a shared in-memory tableBASE table in real time. Order processing can be run 24x7 as well—tableBASE manages the queue table for you.

You no longer have to run your batch processing in off hours, and you don't have to make large upgrade investments. All you need to do is to install tableBASE, configure a VTS-TSR in memory (just once), issue three tableBASE commands, resulting in efficient 24x7 operation.

 

 

# 6 Using tableBASE Process Manager

tableBASE Process Manager (along with tableBASE VTS) augments and the tableBASE core product by providing vastly improved tableBASE table change control. Using tableBASE Process Manager, change control can be performed completely in the background while Production tables are in use. When ready, updated tables are switched for the Production tables currently in use without interrupting ongoing transactions. Down time is not necessary.

tableBASE Process Manager concepts are introduced in "tableBASE Process Manager" on page 25, and detailed in "tableBASE Process Manager" on page 41. These concepts include:

- VTS-TSR mapping to LDS
- Background table updates
- Real-time table switching

This section provides an example showing how you can use tableBASE Process Manager to streamline your change control. The following is covered:

- Contemporary change control
- Change control using tableBASE Process Manager
- The Production environment
- The Maintenance / Test environment
- The cut-over

 

# tableBASE Process Manager change control usage example

## Contemporary change control

Your change control procedures now consist of something similar to:

- change control employing a Maintenance / Test environment on a different LPAR
- change control using special test naming conventions
- no change control

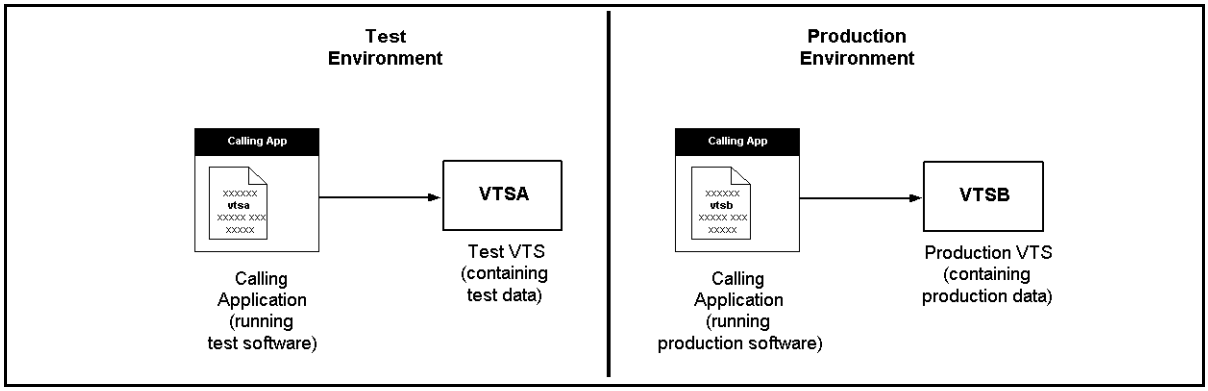


**Figure 6-1: Change control scenario using special test naming conventions**

A Maintenance / Test environment similar to the one depicted above is not completely representitive of the Production evironment; it does not use the Production environment's naming conventions.

## Change control using tableBASE Process Manager

Using tableBASE Process Manager, you can perfectly mirror your Production evironment, including naming conventions—on the same LPAR.

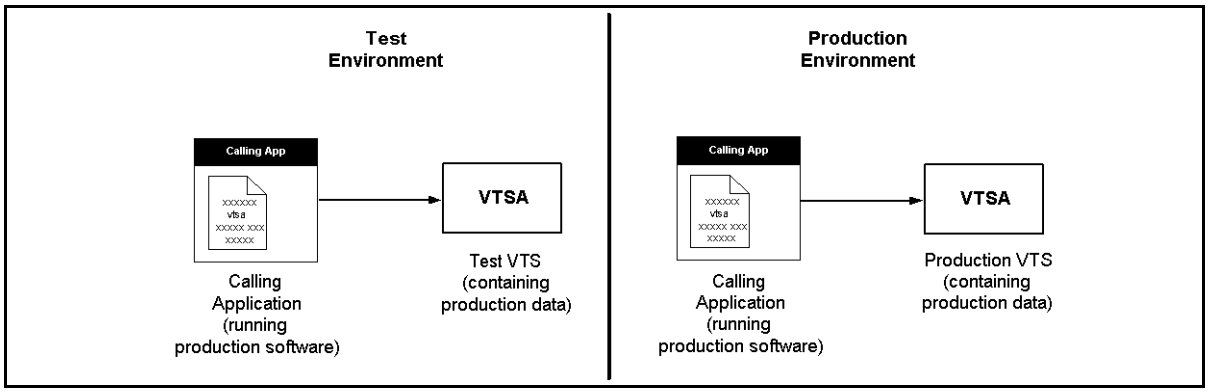


**Figure 6-2: Change control scenario using tableBASE Process Manager**

 

## Behind the scenes

### The Production environment

Figure 6-3 shows a more complete depiction of the Production environment. It includes a VTS Manager called VMAN1, a VTS-TSR called VTSA—it is read-only, and has an alias name called DTA1. The calling applications access VTSA (1) using the alias name. Finally, VTSA is mapped to LDS1 (2).

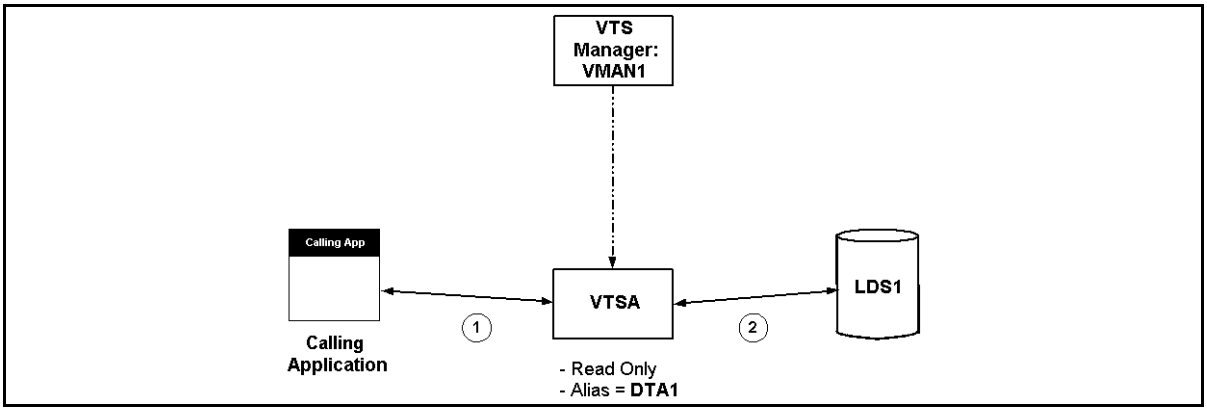


**Figure 6-3: Components in the Production environment**

Data has been sourced to the LDS from a previous LDS copy, or from tableBASE Libraries. The environment is an up-and-running Production environment. LDS1 can source its data in one of several ways: from an older LDS containing older data, via IDCAMS, from tableBASE libraries, via VTSA or from user interaction, via VTSA. VTSA table data is sourced from LDS1 at startup, via the VTS-to-LDS mapping.

All this is implemented by the following:

- Allocate the VTS Manager LDS using IDCAMS:
```
DEFINE CLUSTER(NAME(ACME.VMAN1LDS) RECORDS(032) SHR(3,3) LINEAR)
```
- Define and start the VTS Manager VMAN1:
```
DEFINE TPVM,NAME=VMAN1,PROD=ACMEPD1,LDS=ACME.VMAN1LDS
START TPVM,NAME=VMAN1
```
- Allocate the VTS-TSR LDS using IDCAMS:
```
DEFINE CLUSTER(NAME(ACME.LDS1)- RECORDS(2560) SHR(3,3) LINEAR)
```
- Define and start the VTS-TSR VTSA:
```
DEFINE VTS,NAME=VTSA,PROD=ACMEPD1,TPVM=VMAN1,LDS=ACME.LDS1,AUTOSTART=Y,
TSRACCESS=RO
START VTS,NAME=VTSA
```
- Set up the alias, and assign it to VTSA:
```
DEFINE ALIAS,NAME=DTA1,TPVM=VMAN1
SWITCH ALIAS,NAME=DTA1,TPVM=VMAN1,VTS=VTSA
```

 

**Notes:**

- The VTS Manager requires an LDS for its catalog– it is VMAN1LDS.
- The VTS-TSR must be read-only to use all of the change control features (alias, etc.)
- After assigning an alias to a VTS-TSR, the alias must be used (you will no longer be able to access it using the original defined name).

The SHOW command can be used to confirm/illustrate the setup:

```
--- Processing command # 1 ---
SHOW      VTS,NAME=VTSA,TPVM=VMAN1
Actual number of VTSes returned from LIST : 1
                      VTS# : 1
                  TPVM Name : VMAN1
                   VTS Name : VTSA
                 Alias Name : DTA1
                     Status : RUNNING
                    Version : V610
              Time Start/Stop : 2007-05-31 15:25:14.41
                    TSR size : 10M
                Job Step Name :
            Startup Procedure : ACMEPROC
            Update Time Stamp : 2007-05-31 15:05:10.00
        Updated by (username) : KEITH0
                 Startup Flag : Y
                Shutdown Flag : Y
                  Access Flag : RO
                      Use LDS : Y
                  ListOptions : Y
         Max number of tables : 20
                     LDS Name : ACME.VTSLDS.LDS1
                  Description :
```

**`SHOW VTS,NAME=VTSA,TPVM=VMAN1`**

**Notes:**

- The VTS-TSR VTSA has the alias name DTA1 assigned to it
- It is associated with the VTS Manager VMAN1,
- It has the alias DTA1 associated with it,
- It has a status of RUNNING,
- It is auto-start,
- It is read-only (RO).
- It is mapped to the LDS LDS1

 

## The Maintenance / Test environment

### Prepare for updates

The first step in creating the Maintenance / Test environment is to make an image of the LDS. Note that this is standard IBM IDCAMS functionality. The LDS size/number of records in the new LDS should be the same as the that in the existing LDS (it definitely cannot be smaller). Figure 6-4 depicts what we are doing.

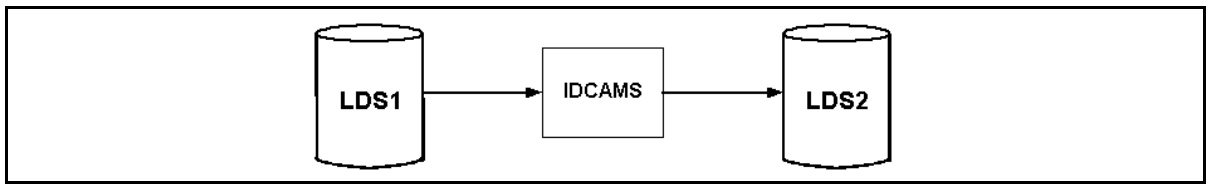


**Figure 6-4: Making an image of the Production LDS**

The first step is to create an image of the LDS used in the Production environment.

- First, we define the new LDS:

```
DEFINE CLUSTER(NAME(ACME.LDS2) RECORDS(2560) SHR(3,3) LINEAR)
```

- Then use it to copy the original:

```
REPRO INFILE(SOURCE) OUTFILE(TARGET)
```

The following JCL could be used:

```
//COPYLDS  JOB 0
//* DELETE & DEFINE NEW LDS
//DEFINE    EXEC PGM=IDCAMS
//STEPLIB   DD DISP=SHR,DSN=ACME.LOAD
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DELETE ACME.LDS2 CLUSTER
    SET LASTCC = 0
  DEFINE CLUSTER(NAME(ACME.LDS2) RECORDS(2560) SHR(3,3) LINEAR)
//* COPY LDS
//COPYLDS  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SOURCE    DD DISP=SHR,DSN=ACME.LDS1
//TARGET    DD DISP=OLD,DSN=ACME.LDS2
//SYSIN    DD  *
 REPRO INFILE(SOURCE) OUTFILE(TARGET)
/*
```

The bolded text does the actual copy. The IDCAMS output confirms success:

```
1IDCAMS  SYSTEM SERVICES                                   TIME: 15:56:3
  REPRO INFILE(SOURCE) OUTFILE(TARGET)                           0002570
0IDC0005I NUMBER OF RECORDS PROCESSED WAS 4
0IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
0IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

 

### Configure the Maintenance / Test environment

The next step is to recreate the Production environment within the Maintenance / Test environment.

This "Maintenance / Test environment" is on the same LPAR as your "Production environment"; in fact, all of these objects can live nicely within the same environment. We will create the same objects that we did for the Production environment:

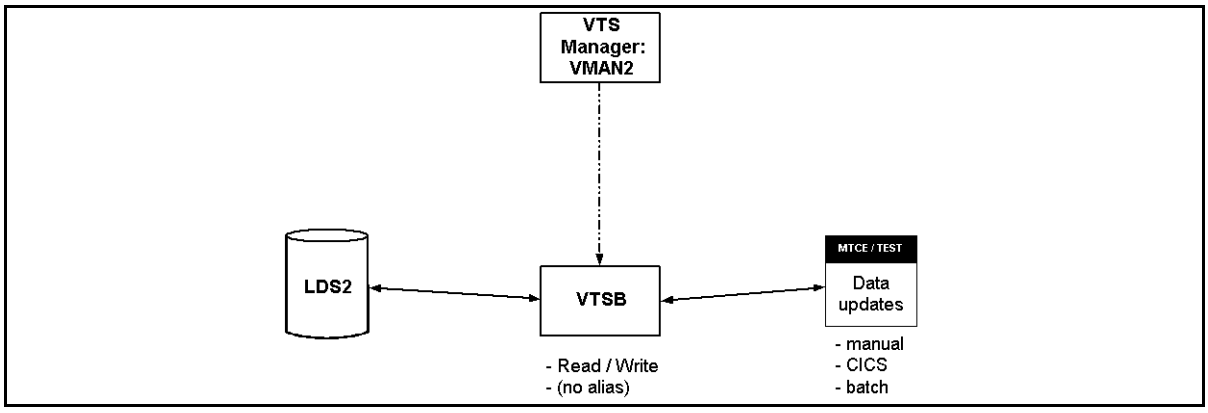


**Figure 6-5: Components in the Maintenance / Test environment**

Commands and procedures used are similar to those used to configure the Production environment.

We just completed the creation of LDS2 (see "Prepare for updates" on page 85), and copied into it the contents of LDS1, the Production LDS. A new VTS Manager (VMAN2) is needed to test the new data in LDS2 using Production naming conventions, and even Production software. It requires its own LDS for its catalog. VTSB must be read/write so that we can update it.

Now is the time to perform the table updates:

1. Updates are made to VTSB using normal tableBASE calls via CICS, batch, etc.

2. When updates are complete, VTSB is shut down (this will update LDS2)

```
SHUTDOWN VTS,NAME=VTSB,TPVM=VMAN2
```

After table updates are completed, there is no further need for VTSB; it can be deleted.

```
DELETE VTS,NAME=VTSB,TPVM=VMAN2
```

 

## Test the updated data

The next step will be to test the updated data before it goes into Production.

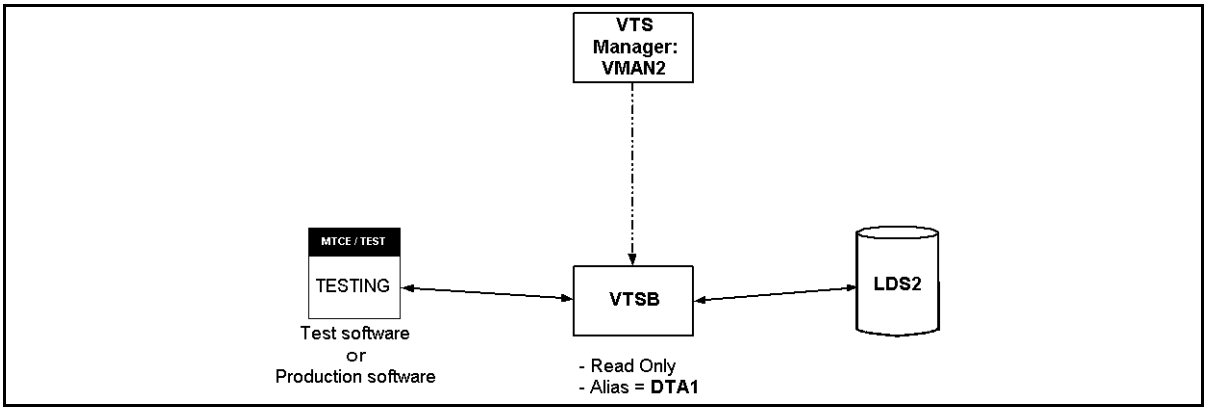


**Figure 6-6: Testing within the Maintenance / Test environment**

Under our Maintenance / Test VTS Manager, we'll need a read-only VTS-TSR, for testing. We'll call it VTSB again; that is what we will call the new, updated VTS-TSR that will be created later in the Production environment. (Note that the original one in the Maintenance / Test environment, VTSB was read/write- we needed a read/write VTS-TSR to update LDS2. Now we need a read-only VTS-TSR in the Maintenance / Test environment to simulate the Production environment.)

It will have the alias DTA1, as is the case in the Production environment. It will be mapped to LDS2.

Testing software will access the VTS-TSR, as is the case in the Production environment. In some cases, you could even use the Production software—this makes our testing more representative of the Production environment.

Now you're ready for cut-over…

 

## The cut-over

After all the preparations (the creation of an update VTS-TSR, the updating and test of data, and the mapping of the new LDS-based data to a new VTS-TSR), the Production environment is ready for cut-over.

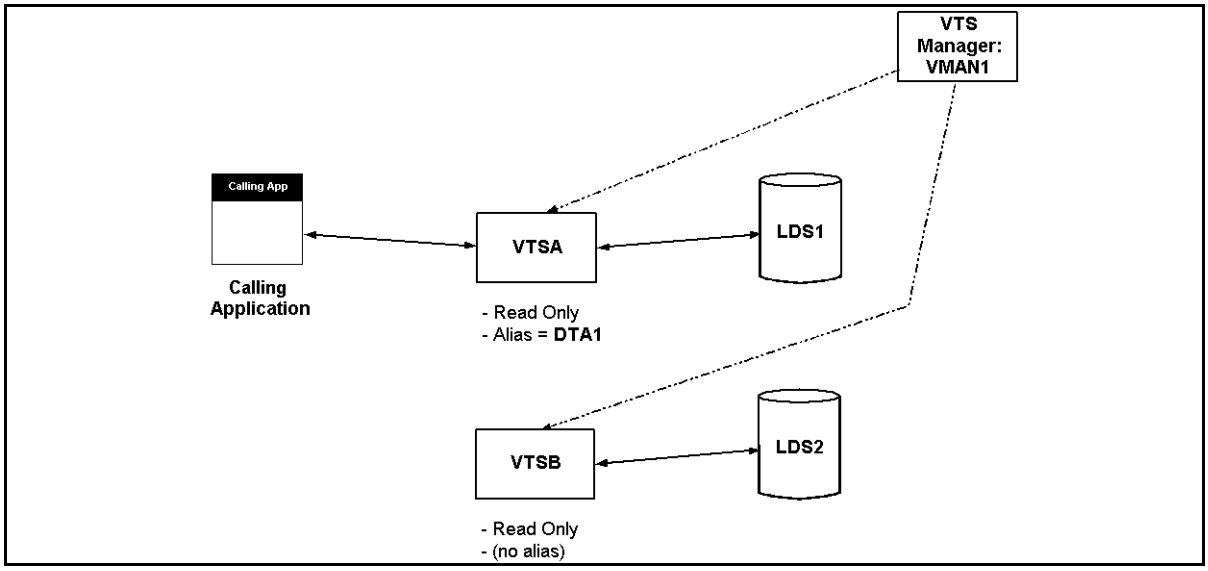


**Figure 6-7: Production environment ready for cut-over**

### Before the cut-over

Note that everything is on one LPAR, and that our existing Production environment is still in place, and functioning as before. So far, nothing has changed with respect to the existing Production data in VTSA / LDS1.

We will:

- Create a new read-only VTS-TSR, and call it VTSB

```
DEFINE VTS,NAME=VTSB,PROD=ACMEPD1,TPVM=VMAN1,LDS=ACME.LDS2,AUTOSTART=Y,
TSRACCESS=RO
```

- Map the new LDS, LDS2 to the new VTS-TSR.
  (as before; see page 83)

**Note:** It should be clearly understood that VTSB is a new VTS-TSR. It mirrors what we did in the Maintenance / Test environment.

To effect cut-over, the following command is used:

```
SWITCH ALIAS,NAME=DTA1,TPVM=VMAN1,VTS=VTSB
```

 

### After the cut-over

After the cut-over, new transactions will now access VTSB using the DTA1 alias. Existing transactions will continue accessing data in VTSA. VTSA and LDS1 can now be shut down / deleted / reallocated.

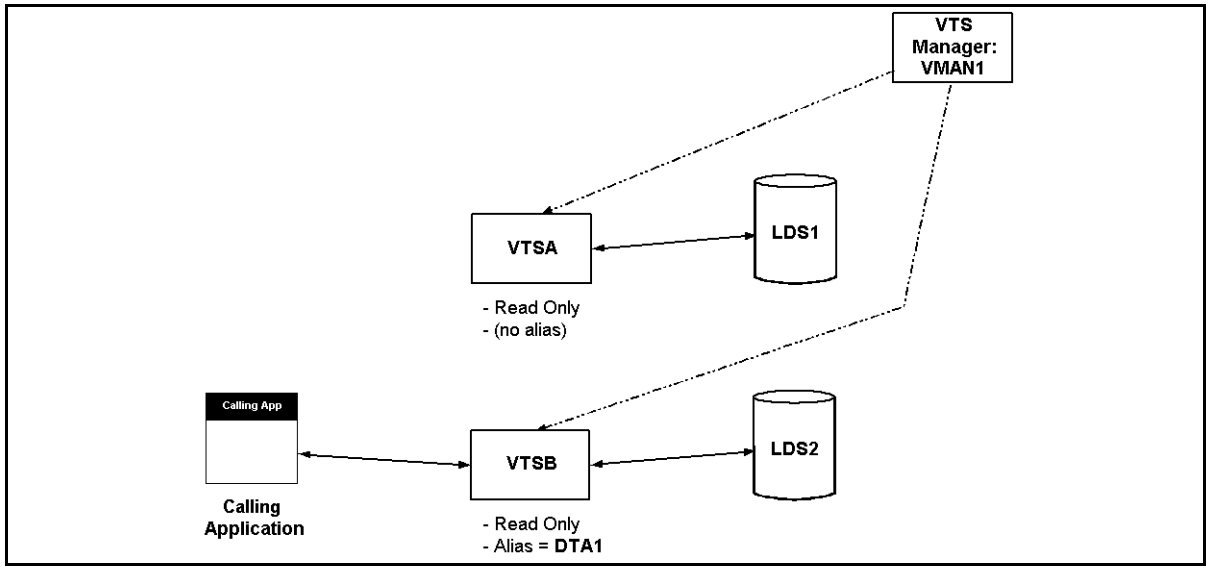


**Figure 6-8: Production environment after cut-over**

VTSB is now the Production VTS-TSR that is being used by calling applications.

From the perspective of the Calling Applications, nothing is changed– they are still referencing DTA1. For this reason, there is never a need to change application software as part of the change control process.

Data in VTSA will still be used by engaged calling application(s), until the end of the current transaction (even though the reference to DTA1 doesn't apply anymore). All new transactions will reference VTSB (via the DTA1 alias).

When all transactions using VTSA data are completed, VTSA can be shut down and deleted, and LDS1 can be re-allocated or archived.

 

## Summary

In summary, you have been shown simple change-control workflow; you have:

- made a copy of the current data
- updated that data in a distinct Maintenance / Test environment (on the same LPAR)
- tested the data
- introduced the new LDS into the Production environment
- replaced the existing data in the Production environment with updated data

# 7
# Using tablesONLINE

## Accessing and Maintaining tablesONLINE

tablesONLINE, an optional component of tableBASE, 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 7-1 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 7-1) contains selections for tablesONLINE access and how they are used. Selecting Option D—DEVELOP APPLICATION—opens the **Application Developer's Menu** (see Figure 7-2) 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 7-1: 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 7-2: 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 7-3.

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 7-3: Define Table and View Screen**

## Build a Table

Select Option D—DEFINE ALL—from the **Define Table and View Screen** (see Figure 7-3) 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 7-4). Select Option U—UTILITIES—from the **Application Developer's menu** to access the **Utility menu**, as shown in Figure 7-4.

```
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 7-4: Utility menu**

Any selection from the **Utility menu** (see Figure 7-4) 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 7-5). 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 7-5: 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 7-1) to make it convenient for you to work with tables:

**Table 7-1: 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 |

 

 

# 8
# 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:<br>• eight blocks<br>• one block for every 20 tables<br>• one block for each generation of each table<br>• 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)<br>• free space equivalent to the largest table |

 

# 9
# Training and Support

## Customer Support

### Telephone Hotline

tableBASE customers with a maintenance agreement obtain support by calling the 24-hour hotline-support telephone number (613-523-5588). The maintenance agreement lists the guaranteed response times for each problem severity level.

DataKinetics Ltd. strives for the highest quality customer support, providing almost immediate response times for support calls to the 24-hour hotline support telephone number during normal business hours, which are Monday-Friday, 8:00 a.m.-5:00 p.m. EST, and usually within one hour during off-business hours.

For less urgent requests, tableBASE customer support can be contacted by email at tablebase@dkl.com.

### Internet

The DataKinetics Ltd. corporate Web site at www.dkl.com contains a customer section that provides answers to frequently asked questions (FAQs) posed by tableBASE users. The Customer Login (Support) section is available to tableBASE maintenance-agreement customers.

A username and password are needed to access the Support section. Please consult the tableBASE administrator or tableBASE customer support for this information.

### Consulting Services

DataKinetics Ltd. is proud to be able to offer our customers the very best in technical consulting services. Our consulting team consists of highly experienced professionals, most of whom have been in the business of building and supporting high performance

 

mainframe applications and operating systems for 20 years or more. Our current consulting services offering includes:

### Upgrade services

We can assist you in upgrading your applications from an older release of tableBASE, to Release 6. Included in this offering is assistance in as many phases of your upgrade process as you need. We can help throughout the upgrade process/project, from upgrade planning, requirements analysis, specification, development, testing, installation, change control, staging, to implementation into your production environment. We can offer you this service either on-site, in the form of actual upgrade staff augmentation, or via secure remote access, on an as-needed basis.

### Installation services

We can assist you in your efforts to install the latest release of tableBASE (Release 6). Included in this offering is assistance in site preparation, software installation, and verification testing. We can offer you this service either on-site, or via secure remote access.

### Optimization services

We can assist you in fine-tuning tableBASE within your infrastructure. Having years of experience optimizing the product within a wide range of installations, our optimization consultants can help you get the most from your tableBASE installation. We will examine your current methods and practices, and recommend beneficial changes. Some of the things we can help you with are:

- Analyse your current access techniques, and adjust them to ensure that they are optimized for your specific systems
- Analyse your current search techniques, and adjust them to ensure that they are optimized for your specific needs
- Review your existing applications to determine if they can be fine-tuned to enhance performance
- Review your current needs to determine if you would benefit from customized exits.

We can offer you this service either on-site, or via secure remote access.

### Application development and upgrade services

We can assist you in your efforts to make your current application environment as efficient as possible. Some of the things we can help you with are:

- Perform code optimization
- Perform code conversion (from COBOL to Java, C/C++, etc.)
- Create custom application wrappers for one or more of your legacy applications

 

We can offer you this service either on-site, or via secure remote access.

### Ordering consulting services

For more information on our consulting services, please contact us -

- E-mail: info@dkl.com
- Phone: 1-800-267-0730 or 613-523-5500
- Web: www.dkl.com

## tableBASE Training

Let DataKinetics Ltd. help you learn more about the advantages of our products and how to best implement them. We can provide product training in a variety of formats for your convenience and to suit your particular purposes. We offer customized instructor-led courses in a classroom setting, either on-site or at our offices, pubic webinars, customized webinars built around your specific needs, and more. Our current training course offering includes:.

### Introduction to tableBASE

This course is targeted specifically at new tableBASE users.
Prerequisites: none.
This course covers the following:

- tableBASE overview
- tableBASE utilities
- Table organization, indexing and searching
- tableBASE programming
- Sharing tables/multi-user updating.

### tableBASE application development workshop

This course is targeted specifically at new tableBASE programmers.
Prerequisites: the Introduction to tableBASE course.
This course covers the following:

- tableBASE review
- table repository and tableBASE API
- retrieval and update programming
- table creation and indexing
- table organization / shared memory tables

 

### tablesONLINE/CICS Administration

This course is targeted specifically at new tableBASE administration personnel.
Prerequisites: the Introduction to tableBASE course.
This course covers the following:

- tableBASE review
- Features / architecture
- Security / session control
- Exit programming

### Introduction to tableBASE Process Manager

This course is targeted specifically at new tableBASE Process Manager users.
Prerequisites: the Introduction to tableBASE course.
This course covers the following:

- tableBASE review / new capabilities
- architecture / getting started
- definitions, startup, shut-down
- change control, TSR switching
- review and debug

### tableBASE support

This course is targeted specifically at new in-house, or third-party IT personnel that will be supporting tableBASE, along with their mainframe systems.
Prerequisites: the Introduction to tableBASE course.
This course covers the following:

- tableBASE technical overview
- how applications access and use tableBASE
- tableBASE libraries and TablesSpace Regions (TSRs)
- tableBASE in batch, CICS, IMS/TM and DB2/SPAS environments
- tableBASE VTS (shared TSRs)
- overview of tablesONLINE/CICS and tablesONLINE/ISPF
- debugging and diagnostic information
- tableBASE maintenance

 

### Ordering training services

For more information on our training services, please contact us -

- E-mail: info@dkl.com
- Phone: 1-800-267-0730 or 613-523-5500
- Web: www.dkl.com

## About DataKinetics

DataKinetics Ltd. is a multifaceted technology organization whose core competency is software excellence. Established in 1977, the company is a privately owned and consistently profitable Canadian company. For over 30 years DataKinetics Ltd. has been providing products and services that help Fortune 1000 companies get the most out of their mainframe computing architectures.

DataKinetics Ltd. strives to deliver high-quality products and high-value services that meet or exceed customer expectations. This goal is realized through management commitment, employee empowerment and the continuing improvement of our quality process.

DataKinetics Ltd. flagship product, tableBASE, is at the heart of mission-critical applications such as on-line banking, trade settlement, e-business, data mining, data warehousing, and unified billing for financial, insurance, government, retail and other organizations. DataKinetics Ltd. is committed to ongoing development of the tableBASE family of products to meet the changing needs of enterprise application development.

DataKinetics Ltd. offers training and consulting services to ensure tableBASE customers are able to take full advantage of the product's speed, adaptability and scalability features (to name a few) when using tableBASE in their applications. Product training is available through instructor-led courses. The DataKinetics Ltd. consultant, versed in the use of table-driven techniques, is a valuable asset whether the customer is building a new application or enhancing existing applications.

DataKinetics Ltd. customer support is renowned in the industry, and helps guarantee product reputation for stability and reliability. Our seasoned support team is dedicated to providing the highest level of service in the industry--every day, around the clock.

For more information, contact DataKinetics Ltd. at suite 202, 2460 Lancaster Road, Ottawa, Ontario, K1B 4S5, Tel.: (613) 523-5500 or (800) 267-0730, e-mail us at info@dkl.com, or visit the Web site at www.dkl.com.

 

 