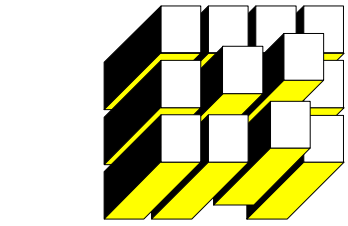
Accelerate Web Access to Legacy Data Sources with tableBASE™ and Java 2 Enterprise Edition (J2EE)

A White Paper

July 4th, 2000

Data Kinetics Ltd.

www.dkl.com



tableBASE™

Contents

Figures

© Data Kinetics Ltd. 2000

Management Overview

Many e-commerce applications rely on legacy sources for both data and business logic. Often, legacy access involves complex designs and cumbersome programming along with less-than-ideal performance. This White Paper shows you how to accelerate access to legacy data sources by combining the proven high performance of tableBASE with the power of Java 2 Enterprise Edition (J2EE) technology.

Data access requirements for large-scale enterprise Web sites often exceed the capabilities of embedded database solutions. Data Kinetics Ltd. has designed an Enterprise Information System (EIS) connector kit enabling enterprise architects and other solution providers to develop their own Enterprise JavaBean and/or CORBA-compliant applications using tableBASE. Using this facility, file-based legacy data sources can be easily Web-enabled by defining them as tableBASE data structures.

tableBASE combined with J2EE technology accelerates both “time to market” and “run time” performance.

“tableBASEÔ in a nutshell”

tableBASE is a high-performance, data access technology that manages and manipulates tables in memory in the IBM OS/MVS mainframe environment.

tablesONLINE is a flexible, interactive CICS front end for tableBASE that handles data entry and table editing tasks, complete with access controls and data validation services.

A new generation of the tableBASE technology, called net.TABLES, will soon be generally available for network distributed architectures such as Linux and Windows NT.

tableBASE can be utilized in several different ways by clients:

- ✓ online data reference for Transaction Processing applications;
- ✓ rules-based logic applications; and
- ✓ table-driven logic applications.

A tableBASE structure is a data source that can be viewed as a database but with a more flexible organization and much greater access performance.

With the new Web-enhancement toolkit, tableBASE users can break through the barrier between their legacy and e-commerce applications, bringing enterprise intelligence where and when it is needed with the processing power of the mainframe server.

© Data Kinetics Ltd. 2000

tableBASE: an Adaptive Technology

tableBASE enables its users to create powerful and flexible applications built upon the foundation of adaptive technology. **Adaptive technology** is a simple concept, yet provides a powerful advantage – the ability to create computer applications that can be readily modified to meet the rapidly changing conditions of our modern world. Essentially, it is based on the separation of business rules and processing logic. This is also known as **rules-based** or **table-driven programming**.

Tailoring the way various users view the same data is becoming more common and necessary, particularly in e-commerce applications intended for building long term customer relationships.

tableBASE, by means of adaptive technology, provides a lens through which users can master change with personalized views of corporate data.

tableBASE Solutions: Use Cases

tableBASE can manage a vast amount of information in memory at speeds up to a thousand times faster than disk. tableBASE makes the most of "above the line" MVS/XA memory space and shared MVS/ESA dataspaces. In the OS/390 world, tableBASE has proven its performance capabilities in both online and batch applications.

High-volume online data capture applications (such as bank card processing) running under a Transaction Processing monitor often have problems with throughput capacity. Mission-critical applications require low response times (below half a second) as well as high availability. Performance of online Transaction Processing applications is greatly enhanced by tableBASE memory management.

High-volume batch applications (such as data warehouse cube generation) may have difficulty completing within tight processing windows. Batch mode performance can be measured by elapsed execution time as well as CPU time and other expensive system resources. tableBASE also enhances performance of batch mode processes significantly.

tableBASE complements RDBMS technology that is best used for structured persistent data by allowing your corporate information to be accessed at memory speeds, that far exceed traditional VSAM or DBMS disk storage. tableBASE actually complements your investment in RDMS technology, letting you process corporate data as complete reusable sets to achieve optimal performance.

© Data Kinetics Ltd. 2000

Why Use Tables?

The principle of separating business rules from processing logic has long been used to create a wide variety of sophisticated computer applications, ranging from operating systems to stock trading. Separating rules from logic – by placing the rules in external tables - makes it easier to develop applications and, most importantly, gives you the power to rapidly modify the application to adjust to our constantly changing world.

With tableBASE you can readily develop table-driven, rules-based applications, thus improving your staff's productivity. And, because tableBASE manages the tables or rules in memory, it also turbo-charges your application's performance.

As documented in our free publication tableBASE Concepts and Facilities (available in PDF format at our Web site www.dkl.com), tableBASE provides definite performance benefits with the following kinds of data:

- **semi-stable data**;
- **program control data**;
- **constants and parameters**; and
- **transient data**.

Semi-stable Data Tables

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

Program Control Data Tables

Program control data tables form a special class of semi-stable data containing decision control information, or rules, for determining which routines should be executed under a particular condition or set of conditions. Control values of this nature have traditionally been hard coded in application programs for optimal performance, but this approach introduces a level of program complexity that hampers subsequent maintenance efforts.

With tableBASE, parameter sets that are often implemented in the form of program logic (or as installation or run-time options) can now be viewed as tables. Examples are report distribution lists, user and password tables and data validation tables.

© Data Kinetics Ltd. 2000

Constants and Parameters

Constants, literals and other parameters that consume a significant part of a program's working storage are suitable for implementation as tableBASE tables. This is another special case of semi-stable data, which should be entirely resident in virtual storage, but can be easily maintained through tableBASE facilities.

The typical approach is to place these constants in working storage defined with copy members at compile time. They are available to just a single task with no additional I/O overhead. This, however, can result in swollen demands for Dynamic Storage Areas, where interactive on-line usage requires multiple instances of the same constants for each transaction generated. Maintaining these constants is also time consuming. Programs have to be re-linked and re-tested for every change.

This data should be managed as tables rather than embedded in program logic. With tableBASE, the same single copy of these rows is instantly available to all tasks needing them with no I/O overhead. Constants are maintained separately from the programs that use them.

Transient Data Tables

Transient data lies at the other end of the spectrum from semi-stable data. It is volatile and forms the bulk of working storage in most online and batch applications. Transient data is not retained once a process is complete. It is typically 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.

The most frequent use of transient data in a batch environment is the summarization of large volumes of data. Examples include sales statistics, payroll summaries or funds transferred tables. The traditional approach to summarization involves creating a sequential extract of the subset of detail records and data fields needed, followed by the sorting of the extracted records into report order, and — finally — sequential processing of the summary information. Using tableBASE, all control totals are generated, re-indexed, and delivered in memory at speeds far exceeding media-based storage methods.

Alternate Table Organizations

With tableBASE, the concept of indexing differs from the traditional disk storage based model. **tableBASE index tables** are built dynamically when an indexed table is opened; they are not stored on the library with the data. And indexes may be reorganized dynamically at any time, in batch or online mode, without any I/O cost.

© Data Kinetics Ltd. 2000

When indexed tables are reorganized with tableBASE, only the indexes are affected; the data tables remain in their original physical organizations. tableBASE keeps both data tables and indexes entirely resident in memory.

Multiple views of the data are available simultaneously through **alternate indexes**. On-line accesses can be processed using various ad-hoc **hash indexes** for high speed random access, while batch jobs can refer to **sequential indexes** for list processing. For example: a particular job may need to reorganize a table dynamically for summarization and reporting, while another may cross reference a table by keying on two or more different fields in an interleaved fashion.

The tableBASE data model

Because tableBASE can represent tabular information of any kind — business data, parameters or decision rules — its data model is actually a "metamodel" (a generic model of possible models). In the Unified Modeling Language (UML), Class diagrams describe the structural view of a system.

Figure 1 shows the *tableBASE Class Model*. The **Class Name** is shown in the top "partition" of each box, **Class Attributes** in the middle partition and the more important **Class Methods** in the bottom partition. **Class Attributes** marked with asterisks indicate "key fields", while those marked with **(fk)** indicate "foreign keys" to other classes.

In tableBASE, tables can be understood as the aggregation of a set of rows. This is known as an "entirety-part hierarchy" relationship. In UML, "aggregation" classes are marked with a diamond symbol that terminates the "association line" leading to the corresponding "part" class. Asterisk symbols indicate the "cardinality" or "multiplicity" of a relationship or association linkage (for example, a **Table** "has" many **Row**s).

tableBASE enables the creation of tables with more than one row layout. When more than one row layout exists, row identifier fields (called View ID) specify the View to be used to interpret the field composition of rows with that View ID. View tables (which define the field contents of particular rows) are not mandatory.

© Data Kinetics Ltd. 2000

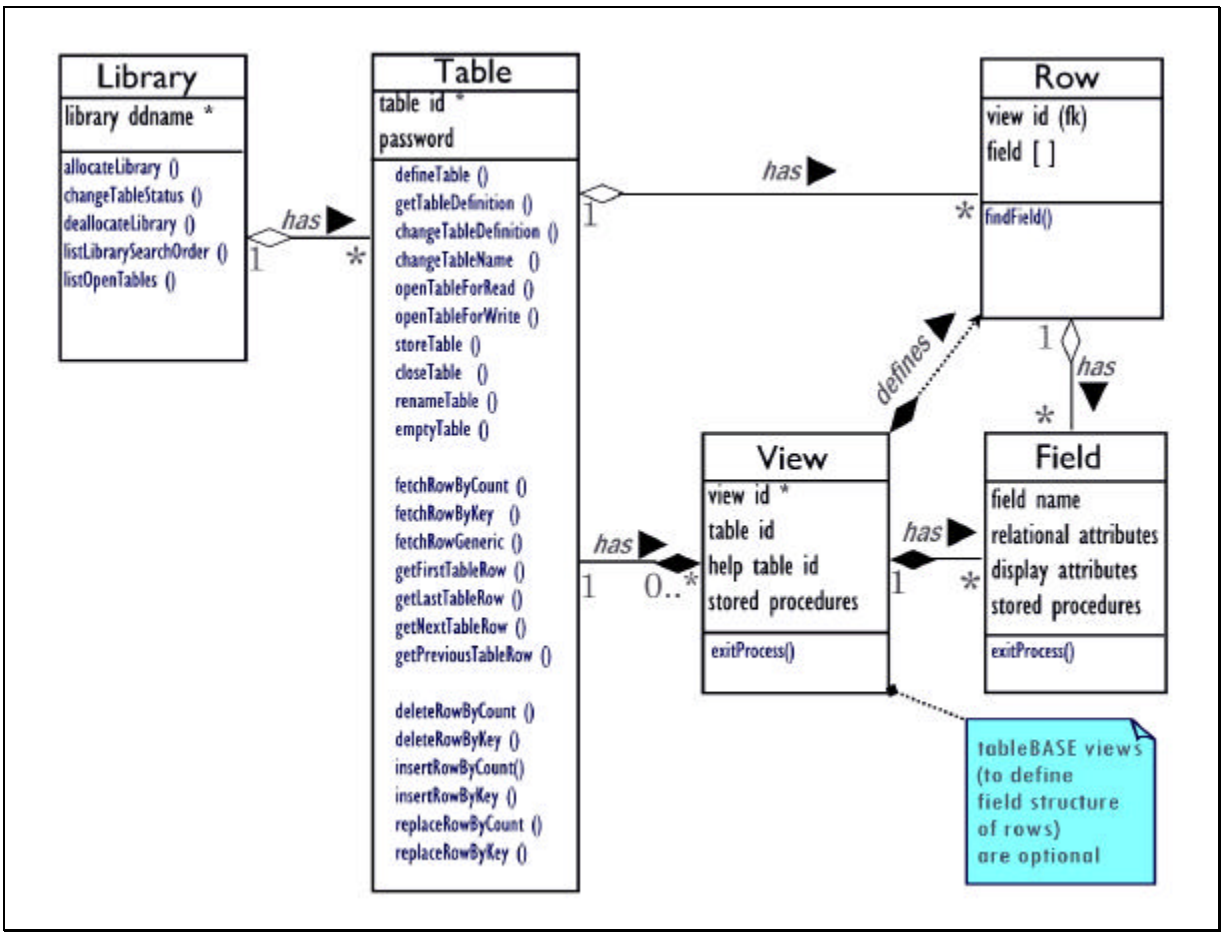


Figure 1 – tableBASE Class Model

© Data Kinetics Ltd. 2000

The Marketplace drives Technology

Modern enterprises have an increasingly large selection of solutions to their Information Technology requirements. The relationship between emerging technologies and the marketplace is highly interactive: as technologies are developed to fill particular gaps or niches, they can also provide new business opportunities for their customer base.

- The **Internet** continues to expand and evolve rapidly. In 1995, there were only 35,000 Web sites world wide. By mid-2000, the number of Internet Web sites exploded to over 16 million, serving more than 300 million users. Thanks to the original robust design, the addressing and network infrastructure has proven to be remarkably stable and reliable. Even so, experience has shown that deployment of "lightweight" or "thin client" solutions is preferable so that performance-related factors can be monitored and tuned on the server side. Server capacity and reliability can be scaled to respond to performance issues. As a spinoff benefit, wireless devices connected by Wireless Application Protocol (WAP) and Bluetooth will also benefit from enhanced server-side capacity to deliver Web content.

- Businesses are using **Enterprise Application Integration** (EAI) solutions to integrate their internal Information Systems (IS) technology for strategic advantage in the marketplace. This is also called **Enterprise Business Integration** (EBI). Corporate mergers and acquisitions, and the steady growth of electronic commerce, lead to current estimates of a world-wide integration market worth more than $1.2 billion by the end of the year 2000 alone.

- While EAI integrates processes within a single enterprise, the **Business to Business** (B2B) model brings in business partners and customers as well. **Supply Chain Integration** joins systems in two or more enterprises to support the flow of goods and services, determining speed and time-to-market, affecting the overall cost of goods sold and – most importantly – customer service and satisfaction.

- **Knowledge Management** facilities are increasingly in demand due to corporate downsizing and the increased mobility of skilled key employees, generating renewed interest in **rule-engine-based applications** to model business rules.

- **Enterprise Resource Planning** (ERP) products provide general-purpose and tightly-integrated application services. Leading ERP vendors include SAP, Siebel, PeopleSoft and Oracle. Each ERP application suite has its own particular architecture and inherent complexities. ERP promises platform independence and scalability as well as maintainability, but ERP vendors have had to struggle to provide tools for integration with other products and legacy applications.

- Emerging **XML middleware technology** is enabling the creation of low cost and flexible connectors between islands of information.

© Data Kinetics Ltd. 2000

The J2EE Application Model

Enterprise JavaBean (EJB) technology is a collection of non-proprietary interface standards that encapsulate the infrastructure complexities of distributed systems, enabling the component developer to focus on business logic. With Sun's latest release of the Java platform, Java 2 Enterprise Edition (J2EE), there is built-in support for Enterprise JavaBeans, Java Servlets, JavaServer Pages and XML.

The Java 2 Enterprise Edition (J2EE) simplifies development and deployment of highly scalable and highly available internet- or intranet- based applications. The J2EE platform provides a multi-tiered distributed application model, the ability to reuse components, a unified security model and flexible transaction control – all standards-based.

Enterprise application services such as transaction and resource management are built into the J2EE platform specification and are automatically provided by the environment. Designers and application developers are freed to focus on the capture and modeling of business logic and the creation of user interfaces.

The J2EE application model isolates functionality into specific components. With J2EE, you can design "thin-client" interfaces around HTML Web pages, interacting with server-side Java Server Pages (JSP) and Servlets. Enterprise Java Beans (EJBs) encapsulate the business logic and server-side access to data sources.

EJB technology models the full range of objects useful in the enterprise with two distinct types of EJB components: Session Beans and Entity Beans. **Session Beans** represent behaviors associated with client sessions -- for example, a user purchase transaction on an e-commerce site. All J2EE-compliant application servers must provide the full set of container services to Session Beans under the EJB 1.1 specification.

Entity Beans represent collections of data -- such as rows in a relational database -- and encapsulate operations on the data they represent. Entity Beans are intended to be persistent, surviving as long as the data they're associated with remains viable. Although the EJB 1.1 specification does not require application servers to provide full container support for Entity Beans, many vendors are working ahead of the specification to do so.

The **middle-tier server** plays the crucial role in a multi-tier application. By managing the business logic and the organization of the presentation view, it enables **a "thin client" architecture**. The middle tier shields the client tier from the complexity involved in dealing with back office systems and data sources. The middle-tier server supports a variety of client types -- Web browsers, Java applications or hand-held devices.

Using J2EE, the client tier has a rich user interface component set available that supports graphical controls using either the Java Abstract Windowing Toolkit (AWT) or the Java Swing API. The Java Swing API extends the AWT to provide a comprehensive set of GUI "widgets" (tables, panes, trees, buttons, etc.) that can be easily customized.

© Data Kinetics Ltd. 2000

The J2EE framework encourages reuse. Applications can be developed or re-engineered rapidly using existing and commercially available components together with new applications and components.

To maximize this reuse potential, design patterns are increasingly employed when designing components and application infrastructure. [1]

The J2EE framework uses the "Model-View-Controller" design pattern (MVC) to divide the application architecture into three kinds of objects. The Model is the application object itself, the View is its screen representation, and the Controller manages the user interaction. This approach is widely known as "multi-tier" architecture as shown in Figure 2, *Multi-tier Architecture and J2EE*.

The client browser-based application does not have to directly query databases, execute complex business rules or connect to legacy applications. The middle-tier server manages these jobs for it transparently.

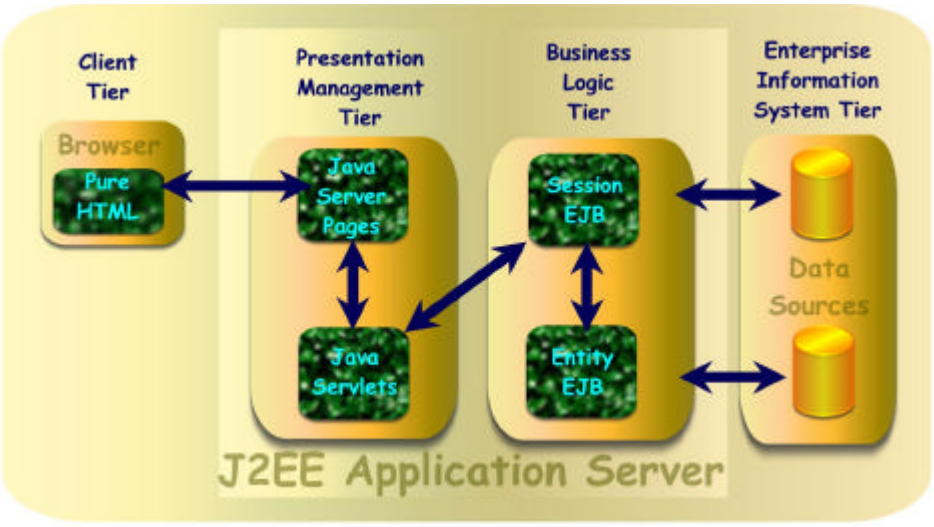


Figure 2 – Multi-tier Architecture and J2EE

[1] Design patterns are described in Design Patterns: Elements of Reusable Object-Oriented Software, by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides and Grady Booch (Addison-Wesley, 1995)

© Data Kinetics Ltd. 2000

Components, Containers and Connectors

The middle tier of an enterprise application server itself can be conceptually subdivided into three fundamental parts: **components**, **containers** and **connectors**.

Components are the packages created by application developers to deploy their business logic for interaction with the client tier.

Containers provide life cycle management, security, deployment, naming, and runtime services to components. Containers intercede between clients and components, providing services transparently to both, including transaction support and resource pooling.

Connectors are a class of database access APIs that plug into existing enterprise data sources, making new applications portable and flexible.

One reason for the distinction between these parts is that each type is provided by a different source. Formerly, the application designer had to deal with the entire complexity of all of these parts. In the J2EE framework, the containers and connectors can be obtained and deployed as "plug and play" products.

tableBASE: a Model Solution

The tableBASE Web adapter kit uses Enterprise JavaBean (EJB) architecture to provide new and existing Web applications high-speed access to tableBASE data. Any mainframe legacy file structure can be adapted via tableBASE.

Users equipped with typical Web browsers (such as Netscape, Mozilla or Internet Explorer) communicate with tableBASE via an application server. The application server uses EJB session beans and entity beans to manage the acquisition and presentation of tableBASE data structures from the "server-side". No custom software installation (other than the Web browser) is needed on the client workstation.

Running under the "server-side" application server, the tableBASE EJB entity beans manage the tableBASE data structures using the Java Native Interface (JNI). JNI uses the custom tableBASE connector library to provide interface "wrapper" classes to exactly match the interface definitions within the EJB business objects.

© Data Kinetics Ltd. 2000

A "Thin Client" Design

The application server uses a set of Java Servlets and Java Server Pages (JSPs) to process user requests contained on standard HTML Web pages. The JSPs enable separation of HTML and Java code.

The application server compiles and executes the JSPs as Servlets which activate EJB business objects and helper classes to manage the tableBASE data source and present it to the client.

The "metadata" that defines the properties of the tableBASE data to the EJB entity beans is stored in XML document format, enabling easy integration with other data sources.

The "Model-View-Controller" Design Pattern

"Design patterns", specially adapted for Java and J2EE, have been used in the design and development of the tableBASE Web adapter kit. Design patterns form a growing catalogue of reusable, object-oriented design architectures that have been customized to solve general design problems in particular contexts. They describe the design of objects and their interactions in an abstract way, allowing designers to devised and implement appropriate solutions to a given problem more rapidly.

The design architecture of the tableBASE Web Interface classes follows the Model–View–Controller (MVC) design pattern, the design architecture most commonly used to design GUI-based applications. According to this design pattern, applications access persistent data within a structured **model**, displaying it as a **view** to the GUI for user interaction. The **controller** coordinates the display of the view to the GUI and responds to the user interaction, making the appropriate changes to the model.

> **Model:** Business entities are represented by data objects used by the application. In this case, they have been built using Enterprise JavaBeans, and other Java classes.
>
> **View:** The HTML pages displayed and updated with a Web browser represent the views in the architecture.
>
> **Controller:** Java classes and methods, Java Servlets and JSPs all fulfill the role of controller to manage the workflow related to the business objects and events modeled by the application.

Among the other design patterns used in the interface design are the "**Aggregate Entity Pattern**", "**Bridge**" and "**Whole-Part**". These collectively provide a unified, coarse-grained remote interface architecture to tableBASE data.

© Data Kinetics Ltd. 2000

As illustrated in Figure 3, *tableBASE Web Access Architecture*, a workstation equipped with a typical Web browser can view and update tableBASE data sources. An application server provides the middle tier services (in effect generating the **view**, acting as the **controller** and providing access to the data source within the **model**).

Any J2EE-compatible application server for the OS/390 JVM environment can be used. This is one of the benefits of having a standard component model.

Some application servers intended for the IBM Unix System Services environment, such as WebSphere, will run as native OS/390 applications. Others, such as WebLogic or Sybase Jaguar, could be set up to run as "guest" applications under VM/ESA service or on another host server such as NT-Unix that can be linked with CORBA to the OS/390 JVM host.

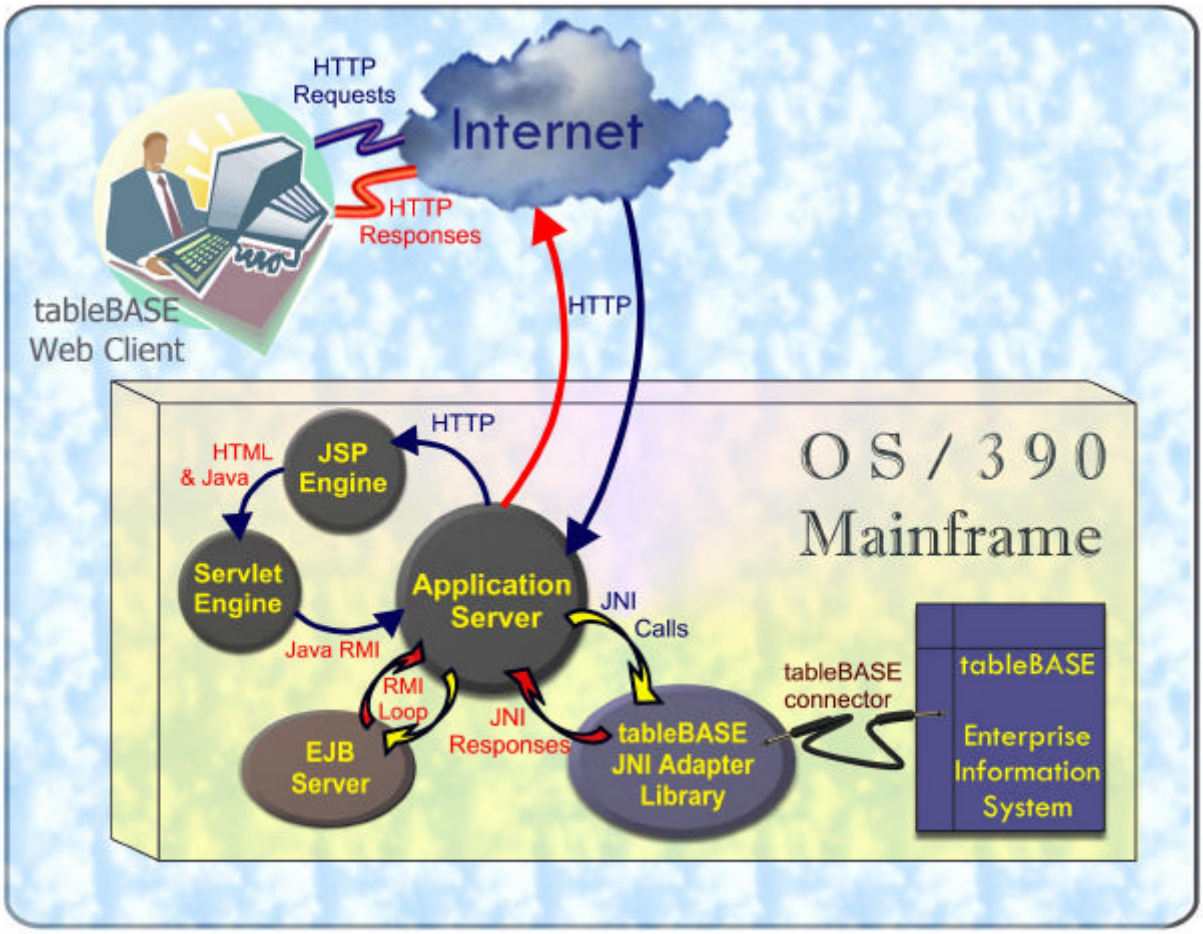


Figure 3 – tableBASE Web Access Architecture

© Data Kinetics Ltd. 2000

A tableBASE Explorer

In one of the class libraries supplied with the tableBASE Web adapter kit, a separately runnable "tableBASE Explorer" utility application is included to enable the user to explore, view and manipulate remote tableBASE data sources using a graphical user interface.

A sample tableBASE library is also provided, consisting of mainframe-resident tables that can be viewed and updated by the tableBASE Explorer application.

The user interface in the adapter kit has a rich set of interaction possibilities provided by the Java Swing API. Java Swing maximizes the possibilities of user interfaces based on windows, icons, menus and pointers to allow users to zoom in to a 2D graphical view of the data relationships in their tableBASE application.

Java Swing also allows simple customization of the "look and feel" features to operate in Motif, Open Edition or Windows modes without changing the basic operation or code.

The Swing user interface components in the adapter kit are all 100% Java components and are freely reusable for custom solutions.

Access and Asset Control

All user interactions with tableBASE are managed by Java servlets and Enterprise JavaBeans activated by the application server. This provides the full benefit of business rules and security policies that protect enterprise information.

Using the "tableBASE Explorer", users can browse tableBASE host information using their usual Web-browser, such as Netscape or Internet Explorer. They can view and manipulate the properties and data content of the remote tableBASE entities. If desired, system administrator and other role definitions can be established to manage access capabilities by *userid* and *password*.

Since the *userid* and *password* for the client browser interface are managed separately from the access control used between the middle tier and data source, customers, suppliers or other business partners can be given access to applications without compromising enterprise information system security.

Secure Socket Layer (SSL Version 3.0) encryption and authentication services are fully supported — especially important when your data leaves your intranet domain.

© Data Kinetics Ltd. 2000

tableBASE in an advanced Component Architecture

tableBASE data sources can be combined with Object Databases, Relational Databases and embedded database solutions by using the tableBASE classes as components in custom Java and CORBA applications. The Java RMI (Remote Method Invocation) objects can be invoked with JRMP (Java Remote Method Protocol) or IIOP (Internet Inter-Orb Protocol) interchangeably without source program modification.

The tableBASE connector classes and EJBs can be deployed so that data updates run as transactions under the Transaction Manager. This allows a "two-phase commit" framework for referential integrity when multiple data sources are used or when failover capability is required. Accordingly, Java Messaging Service, a "message-oriented middleware" framework, is also supported along with IBM MQ series, to enable tableBASE data source updates to be message- and event-driven.

Summary

- ✓ With the "abstraction layer" architecture in the tableBASE Web adapter kit, tableBASE functionality is encapsulated in reusable object-oriented terms and concepts.
- ✓ Together, tableBASE and J2EE provide Web applications with a high performance access connector to legacy mainframe data sources.
- ✓ J2EE contributes the benefits of Java technology: scalability, portability, and programming ease.
- ✓ tableBASE contributes a flexible, high speed data source capability with the computing power, size and speed of the enterprise mainframe.
- ✓ Data repositories of mainframe-based legacy applications can be Web-enabled using tableBASE and the Web adapter kit.
- ✓ The tableBASE Web adapter kit can be used as a component in larger design frameworks.

All put together, using tableBASE within a Java 2 Enterprise Edition (J2EE) component architecture helps the modern enterprise master change by adapting their legacy data sources instead of replacing them.

The clear benefits of **a tableBASE solution** are enhanced productivity and greater return on enterprise technology investment.

© Data Kinetics Ltd. 2000