

Enabling the competitive enterprise through data sharing and integration for z/OS-OS/390

In today's changing and information-driven business environment – mergers and acquisitions, expansion of business services and consolidation of business units are commonplace – the value of being able to integrate and share data is now a necessity to remain competitive. This paper reviews how database architects can enable data sharing within z/OS and OS/390 IMS, CICS, DB2 and batch applications.

The challenge of managing data from multiple sources

There are times when you need to integrate data from various sources (files/tables) and combine them for reports or the creation of summary tables. What would you do if you had data coming in from multiple sources that aren't easily shared? Data sharing allows an enterprise to share a view of data across communities of interest without replicating the data. If you're running 24 x 7 and need to pull together the information that is spread across the enterprise, how would you do it?

The challenge of integrating data can be dealt with quite easily by using temporary tables to collect the data for the final number crunching or reports. In these instances where different business units may each hold portions of the data, or the data resides in different tables, one or more applications can be written to integrate these sources of data for required results. Rules are often required to know which data needs to be integrated, and while this could be hard-coded, the process will be more flexible with the use of rules tables. The major challenge is to be able to integrate the data efficiently and speedily. If you're using rules tables, these can be put into memory for fast access.

The problem in many organizations is that there are too many different, and often, incompatible databases and files. These various data repositories don't live in a vacuum. It is often a business necessity to retrieve, analyze, and synthesize information from several of these sources at the same time. It is not unusual for business users to have to come up with ad-hoc solutions to inter-system requirements.

Organization wide business re-engineering to integrate all (or most) systems has not met with the success that many had hoped for. New requirements are a constant, both from within and without. Business development, reorganization, and new customer considerations are just some of the internal reasons that systems are subject to constant change. External reasons include regulatory changes and mergers and acquisitions.

The problem is not that individual data sources can't be shared. It is that different data sources can't be easily or properly integrated so that information requirements can be met in a meaningful way.

This issue can be addressed through two approaches. The long-term solution is to develop a whole new rules based system so that the data can be integrated and shared easily. The short-term solution is to use in-memory tables to work with existing applications. tableBASE, an in-memory table management system, offers both of these options by enabling rules-based processing that shares and integrates data across applications within the enterprise.

Using tables to share data

A temporary table can be easily defined to hold all relevant data from multiple data sources and can be viewed through multiple index keys by various applications. New index keys can be added dynamically as requirements evolve. By doing all of this in memory, you can add significant performance. The temporary tables can also be data space resident so that they are available to all address spaces (IMS, DB2, CICS, batch). tableBASE provides all of this functionality as well as common access subroutines that can be shared by all applications needing to interact with the table.

With tables that are managed entirely in memory, and with efficient row indexing, you can dynamically alter and massage the data as required by circumstances. Building a new index key as part of this process is feasible and efficient, given that a new index can be built entirely in memory with no access to secondary storage. With this facility designed and implemented, it can open new ways to think about the data on the fly.

Conceptually, it is just as easy to define a new table dynamically that would integrate and merge data from other tables, files, and data bases, as required. There's nothing new about this concept, programmers have been building arrays in working storage for many years. But, if you had a systemized approach, any program could build a new table in a data space and share it immediately in memory with other programs and processes. While you could build such a system yourself, a product like tableBASE offers it inherently.

What if you had a technology that accessed data rows in these in-memory tables with execution paths that are a mere fraction of the lengths of the ones that are used for data bases, and did it all in memory and in data spaces (that's 2 G!) without ever having to access records or rows on secondary storage while applications and queries are running?

And, what if these tables came with pre-built browsing and updating capabilities? If the only thing you had to do was define the rows and columns, and you would automatically have a secure system for authorized users to work with these tables?

What is it worth to you to be able to retrieve data in a tenth the time it now takes, times millions of times each day?

You might decide that it is efficient and effective to make in-memory tables your permanent solution to integrating data from multiple sources. The

examples cited below may help you understand how this approach can apply to your organization's needs.

Sharing and integrating data: Real life examples

Internal and external requirements mandate that organizations must merge data from non-integrated systems. This is often a very difficult and time consuming process.

Synthesizing data from multiple sources to make it available for high speed retrieval; summarizing information, maintaining up to date statistics – all in memory resident tables that are instantly available. If you didn't have to design a way to do this from scratch, you would be much more likely to utilize these approaches. Many Fortune 1000 organizations have chosen to use tableBASE, a product supported, maintained and continually improved by Data Kinetics.

Sharing data between multiple batch jobs

A securities clearing house has designed a Single Settlement Engine (SSE) for domestic and international securities transactions, covering bonds, equities and investment funds. While the SSE delivers some consolidation benefits, they have set out to minimize mandatory change to customer systems in the initial step of consolidation. They also seek to identify service improvements to customers they can deliver around the SSE implementation.

They are running a classic OS/390 mainframe environment, CICS, DB2 and MQ. Their intent for settlement is to run long running batch jobs initiated with MQ Series using RRS (Resource Recovery Services – a sync point manager). Their challenge is to share information between all these batch jobs.

Managing the data via in-memory read/write tables in data spaces will allow them to work with data in excess of 2 GB while ensuring a high level of performance that can handle ever increasing volumes.

Reducing CPU usage on high-read access data

A large retailer has a product pricing file that has been traditionally stored as a database. After successful experience accessing business rules from in-memory tables, the retailer allocates a data space to hold a 1.2 GB table that contains all the product retail pricing data. This table is now accessed by both long running batch jobs and 24x7 round the world queries.

The result has been a radical increase in the performance of these mainframe applications while extending the useful life of the mainframe system, which subsequently reduces maintenance and replacement costs. The re-architected system also requires fewer hardware resources, which translates into additional cost savings.

In-memory usage of tables ensures consistency in data

A credit card provider has a growing incidence of transactions involving currency conversion. These must be accurate and timely. A DB2 database is updated daily with the currency conversion rates. An in-memory table is loaded from the database daily; all transactions on one day must apply the same conversion rate. Remember that this table must be able to efficiently convert any currency in the table to every other currency in the table.

This table is always available to all CICS regions for retrieval.

Creating integrated statements from different data sources

A major bank is looking for a flexible solution to provide customized, integrated statements to its customers. Each customer receives a statement that shows just his specific checking, savings, loan, and credit information and transactions. Marketing messages are custom-tuned to each customer's account and demographic profile.

In-memory transient tables aggregate, format and summarize each customer's monthly account transactions. These tables sort and build indexes as required, in memory. Each new account holder process refreshes and formats a new set of tables.

Another set of shared tables hold all the business rules, marketing messages and formatting controls that direct the processing and display of the account data. These tables are updated as required and available to all programs in the application suite.

For white papers and case studies on tableBASE, the table-driven approach and more, please visit www.dkl.com.

Data Kinetics Ltd.
202-2460 Lancaster Road
Ottawa, Canada K1B 4S5
www.dkl.com