jBASE

MultiValue without Boundaries

BY jBASE INTERNATIONAL

Since its founding in 1989, jBASE has emerged as the platform of choice for migrations and integrations of MultiValue applications with mainstream technologies, offering customers a stable, robust platform now and for the future. In this interview, Dan Ell of jBASE Support — a 32 veteran of the MultiValue industry and one of the original jBASE Value Added Resellers — tells how nearly 25 years of jBASE's pioneering development has led to the evolution of a product perfectly placed for today's and tomorrow's application needs.

How is jBASE Different From Other MultiValue Databases?

The biggest difference between jBASE and all the other MultiValue implementations is that jBASE does not use a Virtual Machine. jBASE runs natively on an operating system so it does not incur the overhead or the inconvenience of running interpreted code in its own, enclosed, environment like other MultiValue databases. A jBASE program is compiled and catalogued as an operating system level executable that can be called from any other piece of executable code capable of calling external routines. Most importantly, once compiled, interoperability with third-party applications and drivers is seamless and native — there are no APIs in the way. If you have BASIC code that is responsible for the routing of your fleet of trucks or for determining the best way to place goods in a warehouse or for assessing risk on a stock portfolio then you can now call it from any other environment — Java, VB.NET, C or C++ or even your favorite IDE.

The biggest difference between jBASE and all the other Multi-Value implementations is that jBASE does not use a Virtual Machine.

Digital Version On-Line

Scan Me!

Running natively with the host operating system ensures that new technologies are easily supported which allows endless options relating to future direction.

What About Web Access?

We can easily present jBASE data via RESTFUL services. We have an interface that allows you to access and update the data via the conventional Account/File or MultiValue way with very little effort on the part of the system administrator. Setup only takes a few minutes if you want to expose your entire system, which will work for an internal or secure web. It takes a little longer to set up security if you want to limit what Accounts/Files are accessible and for what, but that is really just a matter of telling it what to do. The interface works with a simple Perl script that calls a jBASE program (since jBASE is a native program) and it returns the data to the web client in either HTML, JSON or CSV.

For those wondering what a RESTFUL service is; a RESTful service is an Application Programming Interface (API) that follows standard HTML rules and allows the jBASE database to be accessed and updated. It uses the HTTP defined verbs to access and update the database:

GET - Access a predefined set of data

PUT - Update the data

DELETE - Delete the data

POST - Possibly create or alter the actual database structure (add accounts/files/items)

Why do You Think 64-bit is so Important?

Today 64-bit is the way of the world in computing products. Yes you can run as 32-bit but what about all the internal restrictions? 64-bit removes many of the limitations that have caused problems in the past, for example, Shared Library size on AIX, File handles on Solaris and file sizes on just about everything! While some MultiValue databases might have added 64-bit addressing to overcome 2GB file sizing, jBASE 5 is a complete 64-bit implementation of the database. There are no hidden 32-bit addressing limitations, allowing complex and large applications to be handled with ease.

How Does jBASE Access Third Party Databases?

Core to the design of jBASE is flexibility. jBASE was originally designed to allow for MultiValue tools to be able to be used on any form of data. Flexibility was needed to allow an individual to be able to write BASIC code, reports via jQL (jBASE Query

Language) and stored procedures (Procs and Paragraphs) that could work without change against a range of databases such as ORACLE, MySQL and others. This flexibility is achieved by the jEDI (jBASE External Device Interface) which relieves all of the jBASE tools of the burden of how the data is accessed or stored. Any jBASE developed application can read and write to whichever database is required for data storage just as simply as the jBASE database. This means a jBASE solution can be billed as an Oracle, DB2 or SQL Server solution without a major rewrite and at an economic cost. Major end users can match a corporate requirement for RDBMS compliance while still benefiting from thousands of hours of development in tailored applications written in BASIC. VARs can deliver applications to markets demanding RDBMS databases to capitalize on their investment and expand their business. Even the tough task of merging two companies with diverse IT strategies is easily accomplished by jBASE and its jEDI architecture.

The jEDI is implemented with simple STUB files which allow the data to flow through pre-defined routines that handle the actual data retrieval and/or storage. What this does for the user is allow CUSTOMER to be a jBASE recognized file regardless of its final destined database. The closest concept in other MultiValue implementations would be Q-points which allow you to see files in another account or SUPER-Q-points which go as far as seeing them on another machine, the STUB files go a giant step further in that they let you see files in a foreign database.

How Secure is jBASE Data?

jBASE from its conception has been the best MultiValue implementation in ensuring data integrity. jBASE data files are each a separate operating system level file and therefore you do not have the overflow table corruption problems you have with other implementations. Data corruption is much less likely to occur and should it occur it will affect only the individual files. Other MultiValue implementations that depend on an overflow table can have data corruption that affects all files. jBASE also supports complete transaction processing which, if implemented, assures that data transactions are complete.

A simple example of transaction processing in practice can be seen when entering an order. The order updates the order file, the customer file and the inventory file; with transaction processing all the files are updated or none of them are. Transaction processing insures "complete transactions". jBASE includes jbackup utilities that back up the entire system, usually on a nightly schedule. Up to the minute transaction journaling can be implemented assuring that any data corruption or even accidental user error, deletion or update can be recovered. Transaction journaling on jBASE can literally be set up in less than 5 minutes and, when setup, will ensure you have record of transactions assuring data recovery. No other database that I know of, let alone MultiValue type databases, can setup transaction journaling as easily. jBASE 5 has added the capability of warmstart, which means in case of hardware power outage the system can restore transactions automatically to the minute of failure. Many customers take transaction journaling to another level in that they have disaster recovery systems which are updated in real time and should a server problem occur they can switch to the disaster recovery system immediately.

There are over 600 banks live or implementing jBASE in more than 120 countries, so I think it is safe to say that the critical mass for a proven, secure platform for business software has long been surpassed. Today jBASE technology is shipped on a daily basis around the world without concern regarding security or reliability!

What Debugging Tools are There?

jBASE debugging does so many more things than any other debugger I have worked with. Probably one of the coolest is remote debugging in that you can run a program on one terminal and debug it on another. This comes in handy if you have a problem with a screen where something prints and you don't want a bunch of debug commands all over the screen. With remote debugging, the screen that is running the program just shows the program and all the debug commands are on a separate screen. The jBASE debugger is fully interactive with the operating system it runs on which adds a lot of powerful features. With the jBASE debugger, you can redirect the output to a file or pipe it to any operating system command. And, since jBASE programs are native operating system commands, this means you can send the output of a debug to a jBASE program to filter the results or possibly go as far as emailing or texting the output somewhere. Those familiar with Linux understand the power you have with utilities like grep, awk and sed.

What Other Areas Epitomise MultiValue but Allow New Ways of Doing Things?

A great thing with jBASE is the program profiling. jBASE implemented it by setting an environment variable, which means you can profile any program, any time and not impact other users of the same program. There is no need to compile it with special options or run it with special options. With program profiling you can quickly see where the inefficiencies are in a program and it is all done by simply setting an environment variable. I ran it on a set of reports for a company and when we fixed the inefficiencies we literally went from taking longer than overnight to running in less than 30 minutes.

Along the same theme of the jEDI, jBASE spooler was created with that same flexibility in mind. Simply put, a jBASE "printer" can actually be a process that sent a PDF to email recipients, a process that archived reports to a report server or even something that posts information to a website. With jBASE, the printed output is directed through a jBASE filtering and translation process and then passed to whatever program or process the native operating system wants to use. The most typical jBASE printer will actually just simply send the print job to the operating system's lp command. The only limit of where a "printed"

Continues on page 24

JBASE: MULTIVALUE WITHOUT BOUNDARIES

Continued from page 23

job can be sent is your computers connectivity and your imagination.

Why Does jBASE Offer SQL Support?

One of the main benefits of providing a SQL engine for jBASE is that the database can be used with external tools and APIs. SQL has many benefits that can be applied to the jBASE database. In particular with jBASE, SQL allows users to query data where there might be tables within tables and no primary-key/foreign key relationship (these relationships are defined in the dictionary). This is an extreme advantage not available in most other MultiValue systems. As well as integration with external API's the jBASE SQL engine also allows SQL to be used to interact directly with jBASE files. For example SQL can be used wherever jQL is used currently while the rich set of SQL functions allows the creation and manipulation of data tables e.g. inserting updating and deleting records.

jBASE 5 allows you to access SQL directly from the jsh which means you can use jBASE SQL commands in jBC BASIC programs and in stored procedures. The SQL commands built directly into jBASE include everything from SQLCREATETABLE to SQLDELETETABLE and a lot of commands in between which are all familiar to those who use SQL. The SQLSELECT syntax allows for joins and selection of jBASE files or as those in SQL like to call them, tables. jBASE SQL is not an add-on product or third party utility, but is actually a part of the core design of jBASE 5. jBC BASIC programs can access the SQL capabilities by executing the jBASE SQL commands or by using the JQL functions. I don't believe any other MultiValue implementation has SQL commands built-in without either third party or add-on products.

MultiValue is Much More Than Programming in BASIC so Where has jBASE Evolved for the Developer?

jBASE 5 has introduced the jRemote API which provides an API to expose jBC functions like subroutines, execute commands and access files from JAVA. Similarly, jBASE provides a bi-directional interface to JEE by means of the JEE compliant jRemote Inbound and jRemote Outbound resource adapters. This is complemented by a full JDBC 2.0 compliant API for both read and write. The Java developer is certainly well at home when working with jBASE 5.

The Microsoft .NET developer has several options open from the straightforward jRCS API through to the all encompassing functionality provided by mv.NET Solution Objects. This Entity Model based development functionality means that any and all of the power of .NET application development can be exploited against all the great jBASE 5 features we have highlighted.

What About Migrating Applications Across to jBASE?

jBASE supplies lots of tools for migration from other MultiValue platforms. First of all, jBASE has utilities which allow for account restore from the other MultiValue implementations. Essentially, jBASE supports PQN/PQ proc and paragraphs with no change. jBC BASIC includes all but the most esoteric extended commands of other MultiValue BASIC languages. The main difference between jBC BASIC and other implementations' BASIC is the list of keywords that jBASE does not allow as variable names. jBASE includes a utility called portbas that does this conversion for the developer. Although there was a great effort to standardize how MultiValue implementations would work, there were slight differences between R83, REALITY, PRIME and others. Some implementations allow for setting the type you wish to use. jBASE went one step further in that we include preset types, but also allow for a-la-carte picking of over 150 different behaviors including the @(-n) functions. This means you could run as R83 with some of the added functionality from other implementations.

Our professional services team can even offer a free assessment of your current application with technical advice about how to get where you want to go.

How is jBASE Licensed?

As you might expect, a classic license to use with annual maintenance fees has always been available. However when there is more than the single user to license, the Server model is available. Today's applications users want more flexible access to 'open multiple windows' in to an application and that is fully supported with a Multi-session license. Access to an application can now be from a webserver or a terminal server and again a slightly different approach is required and this is supported via Websession licensing.

jBASE runs on most commercially viable hardware and operating system platforms, including virtual and cloud based systems. The classic licensing model described above still works for dedicated requirements but where applications are offered as software as a service, SAAS, model a different approach is required. To match this SAAS model jBASE is available with subscription licensing.

Whatever your application delivery model there is openness and flexibility with jBASE licensing.

How do I Get Started?

We offer a free software evaluation which is fully supported by a team that is often said to be the best in the industry. Go to the website (www.jBASE.com) to register for a trial, contact our sales team for a no obligation chat, or meet with our technical team for a free online demonstration with a Q&A session specific to your own company. At jBASE, we understand the challenges that customers face in maintaining the competitive edge in their industry and we know how to meet them. IS