

INTERNATIONAL Spectrum

THE MULTIVALUE  TECHNOLOGY MAGAZINE | MARCH/APRIL 2017

Structured
Database **Stop** Non Relational

Calling

it a

Storage

Retrieval

Database

Scaling

Performance

Big Data

Also in this Issue

- Practical Guide to JSON
- AI and DSS: Part II



GET CONNECTED.

KNOWLEDGE AND EDUCATION FOR THE MULTIVALUE PROFESSIONAL.

ABOUT OUR PROFESSIONAL MEMBERSHIP

We are all busy in our day-to-day work and staying up-to-date with the current MultiValue technologies can be difficult.

Professional Memberships provide you access to knowledge, solutions, information, and code that you won't find in other locations.

Professional Membership Includes:

- Magazine in Electronic and Print Formats
- Newsletter
- On-Demand Videos
- Live Webinars
- Discounted Conference Rates
- Research papers
- Case Studies
- Source Code



<http://www.intl-spectrum.com/membership/>



CONTENTS

COVER

6 MultiValue Framework vs. NoSQL/Relational Database We don't call a jacket a lapel just because a jacket has a lapel. Likewise, we need to stop calling MultiValue a database just because it has one. There's so much more. Underselling the power we bring to every project is a thing of the past. **BY NATHAN RECTOR**

FEATURES

15 Expanding Your Toolkit: What is JSON? Moving data in and out of MultiValue requires us to understand all of the different ways that we may need to transform the inbound and outbound information. Bennett offers us a practical guide to JSON (JavaScript Object Notation). **BY BENNETT BAROUCH**

20 Business Tech: AI and DSS: Part II SQL is designed for a specific type of optimization. MultiValue is designed for thingedness. Charles explains this odd term and gives some insight into the cultural differences between these two approaches. **BY CHARLES BAROUCH**

DEPARTMENTS

4 From the Inside

16 From the Press Room

22 Master Class In MultiValue

From the Inside

Innovate or your application will die a slow death. Harsh words, but we have been hearing it, and saying it, for years. Our employees and management are demanding more and more from the existing systems and business applications. Most companies have come to rely on their business software and systems more than they rely on their employees.

Despite this, companies have cut back on IT departments over the years. And they have not spent any time rebuilding them. Corporations still look at their IT budgets as 80% on IT maintenance of existing systems, and only 20% on new innovations or R&D. This kind of 80-20 rule that companies have been using for years is a trap. We've been forced to fall into it.

This fosters a culture where forsaking innovation is often a financial necessity. Shifting that already thin 20% to solve short term issues many times becomes the long term plan as well. Today's business systems are falling further and further behind the requirements of today's users. This is impacting business productivity, and many managers don't realize the depths of the problem.

Some of this is due to lack of

planning, which often stems from a lack of knowledge on what can or cannot be done within their existing business systems. Another reason is the, "I can do it better and cheaper" concept that is so prevalent in the MultiValue community. The problem is that is correct only part of the time.

Since the MultiValue environment is so efficient and effective for developing applications, ones that don't take much time or resources, we have a tendency to overlook all the tools that exist in the marketplace. Yes, DIY (Do It Yourself) works for us, over and over again, but it is not the only answer.

We can't afford to forget that the wheel has been invented already. We face problems which have already been solved. Sometimes, instead of home-brewing a tool or application, we need to do the research and spending required. Too often, we look at the tools, and say to ourselves, "I can do that better and cheaper," and spend the R&D budget developing what we could have purchased, a solution which actually is better and cheaper. Some of our DIY solutions are wonderful, but some eat up a part of that

INTERNATIONAL Spectrum

MARCH/APRIL 2017

NATHAN RECTOR
President

CHARLES BAROUCH
Editor

SYDNEY BAROUCH
Editor

TRACEY RECTOR
Layout



Learn more about the MultiValue Symbol and see what MultiValue Technologies and MultiValue Communities exist to help you support and manage your business and systems. To find out more visit

<http://www.intl-spectrum.com>

MISSION STATEMENT *International Spectrum* magazine's editorial mission is to be the premier independent source of useful information for users, developers, and resellers of MultiValue database management systems, open systems business database solutions, and related hardware, software, and peripherals. Published bimonthly, *International Spectrum* provides comprehensive coverage of the products, companies, and trends that shape the MultiValue marketplace as well as the computer industry at large — helping its readers get the most out of their business computer systems.

International Spectrum is published six (6) times per year at the subscription price of \$40.00 U.S. in the U.S.A.; \$45.00 U.S. in Canada and Mexico; \$50.00 U.S. for other countries. Single copy rates are \$7.00 U.S. in the U.S.A. and Canada, and \$9.00 U.S. in all other countries. *International Spectrum* is published by International Spectrum, Inc., 3691 E. 102nd Ct., Thornton, CO 80229; Tel: 720/259-1356; Fax: 603/250-0664 E-Mail: request@intl-spectrum.com. Copyright 2017 International Spectrum, Inc. All rights reserved. Reproduction in whole or in part, without written permission, is prohibited.

PRINTED IN USA

NEWS RELEASES/UNSOLICITED ARTICLES

International Spectrum is eager to print your submissions of up-to-the-minute news and feature stories complementary to the MultiValue marketplace. Black and white or color photographs and diagrams are welcome. Although there is no guarantee a submitted article will be published, every article will be considered. Please send your press releases, articles, and queries to: editor@intl-spectrum.com. *International Spectrum* retains all reprint rights.

International Spectrum is a registered trademark and MultiValue is a trademark of International Spectrum, Inc. All other registered trademarks and trademarks are the property of the respective trademark holders.

 twitter.com/intlspectrum

 [intl-spectrum.com/facebook](https://www.facebook.com/intl-spectrum.com)

Feedback

What came first, the letters or the letters-to-the-editor department?

International Spectrum Magazine has a Feedback Department, sometimes known as Letters to the Editor.

We want to hear your comments, your reactions, your agreement or disagreement with what you see. Also, do not hesitate to let us know about things happening in the MultiValue Community we may not have heard about yet.

Please send your comments by e-mail to:
editor@intl-spectrum.com

80% that should be spent on other maintenance.

Take a look at the ROI with a few back-of-the-envelope figures: sixty to two hundred hours of your time spent developing a tool is likely to cost you around \$4K to \$14K. In many cases, that \$9K — give or take — you have spent DIY-ing would be enough for the license costs, plus a few years of support fees.

Now, look at the maintenance costs if you write the tool yourself. Twenty to five hundred hours a year on maintenance and additional functionality, as needed. Again, this ends up costing around \$4K to \$35K a year. Most yearly maintenance costs for these tools are less than the \$4K. Allowing for third-party tools as part of your solution options is well worth it. We get some problems solved by others,

which allows us to spend more time on improving and innovating our core business systems.

We are charged with building the business software framework that can support the new technologies your CEO is demanding you put to use. IT isn't only on a monetary budget; our time is also a limited resource.

Now would be a good time to do a little research to see what tools exist that would make your job easier and cause your IT budget to be less strained.

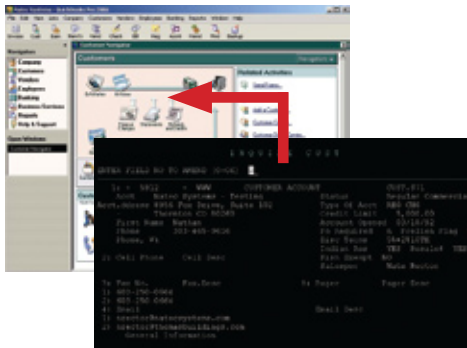


NATHAN RECTOR
President

International Spectrum
nathan@intl-spectrum.com

mv
QB

QuickBooks API for the MultiValue Database



- **Read/Write Directly to Quickbooks Databases**
Customer, Vendor, Invoices, Purchase Orders, Chart of Accounts
- **mvQB API is Designed for the MultiValue Program to Use**
All routines are simple BASIC calls designed for the developer. No special user interfaces required.
- **No Need to Learn the Internals of QuickBooks**
- **QuickBooks Pro/Premier/Enterprise**

NATEC
Systems



Providing Solutions to your MultiValue Questions

Phone: 303.465.9616

E-mail: mvqb@natecsystems.com

Website: www.natecsystems.com

MultiValue Framework vs. NoSQL/Relational Databases

BY NATHAN RECTOR



When we call an MV server a database we are identifying the core aspect but we are leaving a lot of the story untold. Our MV servers are full-fledged application development environments. They include a database, an application development language, a powerful scripting language, and a natural language reporting system. Telling the whole story communicates the value more clearly.

I'd like you to consider referring to it as The MultiValue Technology Framework. This better communicates that our "database" is really several important pieces that are seamlessly integrated together. I know I'm asking you to make quite a perspective shift, but our systems should get the credit they deserve.

There is a lot that the MV servers provide right out of the box. I've mentioned a few in the opening paragraph, here are some more:

- Flexible Data Modeling (Relational Data Model and NoSQL Data Model)
- Enterprise Class Features
- Business Application Layer

“Are you using an SQL database?” Well, no, MultiValue databases are not databases that store information in a format that is optimized for 1NF. At the same time, yes...

Let's untangle some terms that may help us to recast management's view of our tools.

Data Store vs. Database

Data store – a repository of persistent storage and management for collections of data. This can be anything that holds data electronically. CSV is a valid type of data store.

Database – includes the tools (and means) to organize, search, index, retrieve, combine, parse, and/or update data found in a data store.

These two terms are often thrown into the Relational vs. NoSQL debates because not all NoSQL data stores are databases, but all databases (NoSQL and Relational) can be used as a data store. The key difference: The data-

base's ability to provide tools in addition to storage.

SQL vs. Relational Data Store

SQL (Structured Query Language) is a query language which is the API (Application Program Interface) that a relational database uses to provide an interface to their data structure. Relational databases were designed specifically to optimize the use of SQL by forcing the data store to keep the data in 1NF (First Normal Form). This caused the terms *relational database* and *SQL database* to be treated as interchangeable.

Hence the confusion when someone asks, “Are you using an SQL database?” Well, no, MultiValue databases are not databases that store information in a format that is optimized for 1NF. At the same time, yes, because nearly every MV database does support the SQL query language.

MultiValue query (sometimes called Access, English, ReTrieve, etc.) is a query language and MultiValue BASIC is an API, SQL is a tool for CRUD (Create, Read, Update, and Delete). None of them define their data storage format. They just leverage it in an op-

timized fashion. Several MV flavors allow you to select a back-end data store without any change to your queries or programs. Dan McGrath did an excellent presentation at the last Spectrum show on how to program in mvBASIC *but* CRUD in Big Table, which is Google's NoSQL data store.

The key idea here is that while mvBASIC and MultiValue query both allow easy interaction with the MultiValue files, they are not what define the MultiValue data store.

Flexible Data Modeling

For a long time flexible data modeling was looked at as a cheap hack and was frowned on by the traditional database administrator. That is no longer the case due to the rising popularity of NoSQL databases. Spun one way: SQL is well structured to enforce uniformity of methods while NoSQL (including MultiValue) is undisciplined. Spun the other way: SQL is inflexible and does not model the real world while NoSQL (including MultiValue) offers the option to model different data differently, either as an aid to clarity or for optimization.

MultiValue is still quite unique in our data modeling compared to other NoSQL databases. We can do one or more (or all) of the following in the same database, even sometimes in the same database table:

- Non-structured data (schema-less/application-based schema)
- Structured data (1NF, MultiValue attributed)
- Virtual data structures (data structures imposed on non-structured data)

- Mixed data structure (mixing structured and non-structured data in the same database)

Evolution From SQL to NoSQL

Relational databases are designed to enforce data structure and consistency on developers. That really is its whole design: Impose data requirements, restrictions, and structure. The critical assumptions are that data types are the most important aspect of any field. It assumes that single values are the only approach to data architecture.

NoSQL, including MultiValue, is designed to store, distribute, and access data using the best method for each case. NoSQL technology was originally created and used by Internet leaders such as Facebook, Google, Amazon, and others because SQL has trouble scaling in complexity. These giants require database management systems that can write and read data anywhere in the world, while scaling and delivering performance across massive data sets and millions of users. In the world of big data, these are some of the biggest.

NoSQL has evolved to meet the needs of high-speed transactional input/output. It is predicated on the idea of constantly evolving data models. And it does that while still keeping the enterprise-class features associated with relational databases. In this way, MultiValue is *absolutely* a world class NoSQL database.

If you don't think we've been evolving with the times, go to your vendor of choice and ask for the last few years worth of version change notes. Just because your shop may not be using the features doesn't mean they aren't there. If your shop isn't using the newest fea-

tures, maybe they should be. There's a lot to gain in the newer releases.

MultiValue vs. NoSQL Data Store

Conceptually, most NoSQL data stores are not much different from what MultiValue developers have used for many years. Believe it or not, we have been ahead of the innovation curve for years. Sometimes we do a poor job of explaining (bragging) about MultiValue because we don't realize how hard other data stores and query engines have been working to keep up with us.

Let's look at a couple of examples. For reference, here's the SQL 1NF approach to data:

Key	Name	Dollar
129k	Nathan	\$1234.00
128p	Nathan	\$9881.00
515q	Nick	\$9881.00

NoSQL Key-Value Data Stores

Key-value data stores emphasize the read/write of non-transactional data. The data is stored as a non-structured data element with a single value, and are accessed via a unique primary key. This type of data structure is best for applications that need fast input/output of information and don't care about structure of the value stored in the key. The term *single value* is misleading because the value might be one document or one JSON string, not just one number or a single word.

Key-value example:

129k.dollar	\$1254.00
129k.name	Nathan

NoSQL Document Stores

Document stores emphasize the storage of JSON or XML. Similar to Key-value except that it provides query and indexing on popular fields found within the JSON and XML data elements. This allows developers to store JSON information natively, but still provide

a schema to create a stored data model that can be queried.

Document store example can be found in [Figure 1].

NoSQL Column Stores

Column stores are best stylized as an inverse of a relational row/column data store. While not originally a NoSQL data store, but more of a relational indexing feature, it has grown into a data store that is popular for specific types of data models.

Instead of each column being a field of information within a row, the column is the primary concept, and the row is secondary.

Column store example:

Name		Dollar
ID	Value	ID
129k,128p	Nathan	129k
515q	Nick	128p,515q

NoSQL Graph Stores

Graph stores are really one of the most innovative data storage methodologies in the NoSQL world, but it is relatively limited in its applications. Graph data stores use structures that link the relationships between the data of each record. This is a very specialized format that allows very fast BI, analytics, and data mining. The relationships need to be defined as they would be in a traditional schema, but the data is categorized, reduced, exploded, and processed so that the inter-relationships

```
[129k | {"dollar": "$1254.00", "name": "Nathan"}]
```

Fig. 1



Fig. 2

are easily queried, instead of querying the data itself.

Graph data example can be found in [Figure 2].

Enterprise-Class Features

Any LOB (Line-Of-Business) application will need some if not all the enterprise-class features. This has been a major argument against NoSQL. Fortunately, this is one of the places where MV is ahead of the other NoSQL players. MultiValue databases have had enterprise class features built-in for many years.

The features that are often looked at are:

Security

Access and update security provides the ability to control individuals, and/or applications, providing a sufficient granularity to the security of the data, processes, and endpoints. Most existing relational databases excel at table and field access security but lack strong endpoint security. Modern application developers have since moved beyond the need for security on individual fields within tables due to the ongoing evolution of application design. Since most databases provide access to the data through CLI (Command-Line Interface) endpoints *and* direct SQL query statements, only restricting access to fields and tables at the query level is no

longer a winning strategy. This is part of the problem

with SQL: It only supports one way to do things. And that way was defined in a previous age. Modern hardware, speed, programming languages, and user-driven demands are all moving away from that aging model.

I always loved this misguided quote from Ben Rossi, arguing for field security: “And even where security can be added to applications, this puts a huge burden on the application developers, not to mention the additional cost and time implications.” <<http://www.information-age.com/putting-enterprise-nosql-acid-ambiguity-out-123458126/>>

The reality is that most modern application developers are placing the security into the application regardless of the database security. The main reason for this is that most applications have to use security for the presentation model, rather than the rights to the data itself. It is about what the *users* can see, access, or update, not about what the *applications* can access or update.

If there is security on the database tables, then the application developer has to address the security violations *and* present UI elements to notify the user of issues. As an application developer, that makes database security a burden more often than it is an asset. Again, since most applications embed CLI scripts into a full blown GUI, the complexity of application development has evolved beyond that level of granular security.

Encryption

Database storage encryption is rising in importance because enterprise systems are routinely in danger of being hacked. This is a feature that you should implement as a must. Payroll and any other sensitive data that can

be used for user identity theft should be routinely encrypted. The same goes for anything which might be a target of industrial espionage.

Data theft is no longer limited to hacking users' accounts; mining the raw bytes of your hard drive for interesting information is becoming easier. In addition to that, our backups are often not encrypted, but are routinely being saved to hard drives on the network. Those trusty old backup tapes weren't network accessible.

ACID Compliance

This has always been a stickler for database comparisons, and a requirement for the enterprise checklist. It is also something that relational databases have always been good at due to the fact that they are designed to enforce data requirements on developers. When working with 1NF, it is very difficult to do something incorrect with your data from a compliance standpoint. This enforcement is often an issue with developers that have constantly evolving data models, but until the NoSQL database started being used, they didn't know there was any other way.

So naturally, if a database didn't enforce ACID, it was *wrong!* That was the traditional way of thinking about systems. The interesting thing about ACID in modern transactional applications is that the code takes care of that anyway. If a primary key doesn't exist in the database, the application has to respond to the error and correct it or notify the user. So to keep from having to address the data error after the fact, application developers are moving to control the front end of the application process, instead of the back, where ACID would normally be enforced. The requirement for ACID

is more about "we have always done it like that," than any specific application requirement.

For many developers, the primary reasons for ACID were dead before they started in the industry. It comes from a time when databases existed on a single machine, and issues with multi-nodes or servers never had to be considered. It was built to help with transactions boundaries, but when you get into applications development, transaction boundaries aren't always clear. ACID's attempt to manage consistency imposes delays on database updates when you have more than one node or machine in the mix. Because it isn't optimized for the realities of modern networks, ACID tests must check all the nodes, or servers, to verify that the information exists. If it doesn't, ACID issues a roll-back to the application because of a perceived sync error.

Since ACID across clusters, nodes, and servers is very hard and can cause bigger issues, it is often turned off or ignored except for the simplest cases; for example: Does a customer record id exist? Many ACID shops are ACID in name only. Full implementation can put you out of business.

Replication

Even though ACID interfaces are supposed to include support duplication and replication, replication is often looked at separately from the ACID requirement. And it should be. Replication is often more important than ACID in modern applications. Business nowadays require 24x7x365 on their servers. Replication makes that possible.

It isn't always about backups though. Replication is also a way to support reporting servers and data warehous-

es. That means that replication is not always about identical data, and that's another reason ACID's day has passed.

Get Close to the Data

Most NoSQL databases and relational databases have had a problem that MultiValue does not impose on its application developers. Non-MV systems make it a challenge to get as close to the data as possible. There are a lot of reasons we want this closeness, but the main reason is performance. That may be hard to fathom if you grew up in MultiValue. Think about it. Why would you design a database that doesn't prioritize ease of access for developers?

Any large or complex application requires interaction with large volumes of data and the relationships between that data and other data within the database. Most of these relationships are not schema defined, but instead are dynamic. They have to be calculated on-the-fly.

Contrary to what database administrators often believe, the traditional database is just a place to persist data for use by an application at a later date. The database by itself will not keep the data fresh and up-to-date without an application and/or user using it. If it doesn't get used, it gets old and stale.

As data volumes grow in the database, the amount of processing that has to be done by the application grows as well. If you have to work with large datasets, then processing the data as close to the database as possible decreases the amount of time it takes to get an answer to a specific business process.

Relational databases tried to address this by creating transaction scripting (T-SQL, PL-SQL, etc.) to facilitate

data processing as close to the data as possible, but transaction scripting is still limited by the fact that relational databases are primarily focused on storing and retrieving data, not processing and using the data. As I keep saying, SQL was designed based on outmoded assumptions. Unfortunately, if you look at many of the NoSQL databases currently being used, their transaction scripting is still just as primitive, if not more so, as their relational database counterparts.

They both depend on other application development languages to do the processing of the data. This means that the data has to be transmitted from the database to the application, processed, and then returned to the database with changes, updates, or additional queries based on the processing. Literally speaking, they are required to handle data in a highly inefficient way. This is not a failure of the SQL model or the various NoSQL models. They could implement more robust close-to-the-

data tools. We've had them since the inception of Proc and mvBASIC.

The performance of accumulating the datasets and transmitting the datasets to an application becomes a bottleneck. Even with tricks like view, cursors, and transaction scripting, the database is still relegated to just retrieving the data and pushing it (in bulk) to somewhere else to be processed, and then pulling it back to be stored.

This processing bottleneck was one of the problems that NoSQL databases were trying to address, by changing the way the data was stored. They are predicated on the belief that doing the wrong thing (shoveling data) *faster* was the solution. The retrieve time is decreased, compared to SQL, so the application doesn't have to wait as long to process the remaining data. But, as you can see, NoSQL databases only address one aspect of that performance bottleneck, the retrieving and storing of the data itself. That still requires the data to be offloaded to the application in order to do additional processing, parsing, or combining that wasn't built into the database schema.

MultiValue does not suffer from these performance problems. It never has.

Tiered Application Architecture

We can see some of the value provided on the database storage side of things. Let's now look at the application architecture that the MultiValue framework offers.

We have all heard of the *N-Tier* application architecture. This is how most legacy applications and all modern applications are developed. Most developers usually choose between a 2-Tier or a 3-Tier architecture.

The two tiers in a 2-Tier architecture are the database and a client application, which work together [Figure 3]. The client application generally includes all the business rules, workflow, and the data processing code. It exchanges information with the database. This is the simplest architecture, but it creates a lot of traffic between the database and client.

The three tiers in a 3-Tier architecture are: database, application processing (containing the business rules and workflow), and a client application (for presentation). This is the preferred architecture for most LOB applications that use relational databases [Figure 4].

The 3-Tier architecture allows developers to put the business processing applications on the same machine as the database. By doing this, they get the majority of the application as close to the database as possible, so the processing time of pulling and pushing the data in and out of the database is decreased. The end result of the application processing layer is then transmitted to the client to be presented or interacted with.

When building modern MultiValue applications (GUI, Web, Mobile, IoT, etc.), we have a tendency to choose what we think of as a 2-Tier architecture. You really aren't creating a 2-Tier application, though. The third tier is inherent in the MultiValue model. Instead of the traditional independent layer pulling and pushing the data in and out of a database, our third tier is on top of the data, and therefore closer to it.

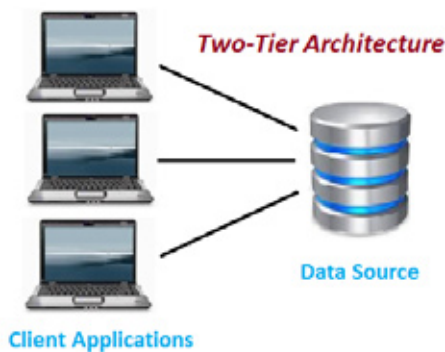


Fig. 3

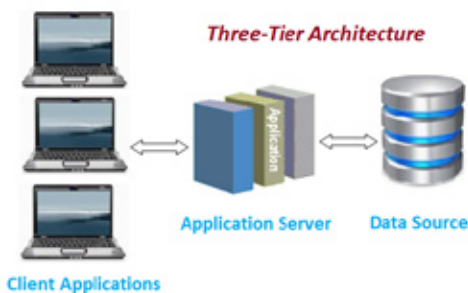


Fig. 4

The MultiValue Database Becomes an Application Framework:

This is what sets MultiValue above being a database and qualifies it as an application framework. MultiValue acts as a tool to supply the structure and templates for constructing an application. And that's the very definition of a business-centric framework.

Any application framework generally allows all of the following:

- Skeletal support for applications
- Ability to create reusable components for an application
- Abilities to retrieve, process, and update data
- Interaction with input and output for users, applications, and processes

The MultiValue framework is held together by mvBASIC, but also supports query, reporting, PROC (Procedural Control Language), and Paragraphs (Batch Scripts and macros). Many of the database management tools provided with the MultiValue system are written using mvBASIC.

What is included in the MultiValue framework:

- Database data I/O
- Database schema management
- CLI (Command-Line Interface) I/O
- Math and string functions
- Background scheduling
- Printer management
- Reporting Tools
- Multi-user/multi-threaded support
- Encryption and hashing

- Data triggers and indexing

The mvBASIC environment's features aren't news to MultiValue developers. We use them daily to create LOB (Line-of-Business) applications. But its real power is the fact that you can't get a full blown programming language any closer to the database than this.

This means any calls done to endpoints into the MultiValue framework will process, retrieve, and update the data without having to transfer it in and out of the database. Since the MultiValue framework also includes a process scheduler, a developer could run processes in a multi-threaded/parallel processing model. This is why MultiValue developers can write 2-Tier applications which provide the value of optimized 3-Tier applications.

Framework Endpoint

The MultiValue framework has built-in support for several different endpoints in order to provide applications developers different interfaces and protocols.

CLI (Command Line Interface)

Most databases have a command-line interface for database management, but that is all it's designed for. The MultiValue framework supports full user interfaces at the CLI level. Input, output, data validation, and many other features are supported through Telnet, SSH, and O/S shell scripts.

Web (REST, SOAP, HTTP)

The MultiValue framework has additional support for interfacing with web software frameworks, as well as the ability to directly present data through web services. MultiValue frameworks are not limited to just presenting data for other web frameworks to use. It can also be used to generate HTML, CSS,

and JavaScript directly, using CGI as an alternative for the CLI user interface.

RPC (.NET, JAVA, PYTHON)

RPC calls are supported, allowing the MultiValue framework components to be accessed through .NET, JAVA, and Python. These interfaces are not limited to just interacting with the data, but are best suited to interact with existing MultiValue BASIC routines, which can act as efficient pre-processors.

SQL Mirror (Access/update SQL databases)

Even though the MultiValue framework includes its own database, and is optimized to work with that storage structure, it is not limited to the MultiValue database files. It can also interact with SQL data sources as natively as its own internal storage structure.

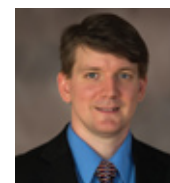
ODBC/SQL

As stated before, SQL is a query language, which is also supported through the MultiValue framework.

Conclusion

Stop trying to explain MultiValue as a database. It is so much more than that. The real power resides in the application framework it provides to developers.

MultiValue, in one line: A MultiValue system is an enterprise class business application framework with a built-in application development layer for 3-Tier programming that combines the power and stability of a relational database with the flexibility of a NoSQL database. **IS**



NATHAN RECTOR
President
International Spectrum
nathan@intl-spectrum.com

FROM THE PRESS ROOM



Carnation Software Release 15.4 of MacWise

MacWise version 15.4 added support for MacBook Pro Touch Bar function keys.

Some MacBook Pro models have a Touch Bar instead of real function keys. By default, the function keys will not appear in the Touch Bar. To activate them, select "Activate Touch Bar on Startup" from the Connection Menu. Then follow the instructions.

Note: The "Activate Touch Bar on Startup" menu item will only be available on MacBooks with a Touch Bar. ■

upcoming Gartner's Data & Analytics Summit at the Gaylord Texan Hotel & Convention Center in Grapevine, Texas.

Entrinsik will be demonstrating how Informer's extensible architecture and modern intuitive user interface combine to create the most customizable, enterprise-ready data discovery platform on the market. Informer 5 is scalable and flexible for large deployments, while ensuring fast performance for governed user analysis and interactive exploration.

Throughout the Summit, Entrinsik welcomes the opportunity to meet you in person at booth #805 to discuss your business needs and how Informer can help your organization address data analysis requirements. As a sponsor of the event we are bringing some of our best data experts and leaders to meet with you to discuss how data & analytics can be your most powerful catalysts for change.

This year's agenda features eight tracks covering your biggest data and analytics challenges and priorities.

- Leadership and Organization: Build the Data-Centric Team
- Business Outcomes and Strategy: Realize the Value
- Master Data Management: Curate Your Most Critical Data Assets
- Analytics For All: Reshape

Entrinsik announced it will showcase Informer 5 at the

the Entire Enterprise

- Governance: Maximize Leverage and Control Chaos
- Architecture and Technology: Modernize the Foundation
- Advanced Capabilities: Be Sophisticated and Precise
- Innovation: Explore New Frontiers

"Entrinsik is pleased to be supporting Gartner's Data & Analytics Summit," said Entrinsik President and CEO, Doug Leupen. "The Summit will offer an excellent educational opportunity, with seminars from BI industry thought leaders and Gartner analysts. We are especially excited about the Gartner BI Bakeoff where we'll be able to show exactly how Informer's agility, power and extensibility can help organizations use data to streamline operations and identify opportunities to grow." ■

applications in character mode (a.k.a. "green screen"), Netbuilder now also runs the applications in a BUI, a browser user interface.

"This has been a very complex project, taking existing applications and running them in a browser. Our architects and developers have pulled off a hat trick with MVON# tools for running MultiValue in .NET with SQL Server, MVON# Netbuilder for running former SB+ applications in .NET with SQL Server, and now adding a Netbuilder BUI for running those same applications with a browser user interface," said Dawn Wolthuis, Executive Vice President of ONgroup.

While this browser user interface is written for those with existing SB+ or Netbuilder applications, organizations that have written their own application development tools in MV BASIC can use MVON# to turn their tools into .NET with SQL Server development tools. Grant Hart, the Chief Software Architect for ONgroup, indicated that, "in some cases, the same techniques we used to create a browser user interface for Netbuilder can be used to web-enable applications written with other MultiValue development tools."

Once a MultiValue application is running with .NET as the run machine and SQL Server as the MV DBMS, the full Microsoft developer toolset is also available for ex-

MVON# Netbuilder from ONgroup Intl runs SB+ (System Builder) applications in .NET. In addition to running the ap-



Entrinsik Showcases Upcoming Release of Informer 5 at Gartner's Data & Analytics Summit 2017



MVON# Netbuilder runs Former SB+ Applications with a Browser

FROM THE PRESS ROOM

tending and integrating the application. MVON# turns MultiValue into a toolset used in conjunction with the Microsoft platform. MVON# Netbuilder facilitates moving SB+ applications to .NET with SQL Server at the back-end and now with a browser at the front-end. ■



Paradigm Systems, Inc. Appoints Meier Business Systems as Asia Pacific Distributor

Paradigm Systems announced that it has appointed Meier Business System (MBS) as its distributor in the Asia Pacific region. Under the agreement MBS will become the primary provider of service and application support in the region.

"With MBS as our distributor in the Asia Pacific region, we bring our state-of-the-art solution for U2 database management to the area" said Jay LaBonte, President and CEO of Paradigm Systems, Inc. "Our goal is to continuously improve our customers' experience in every aspect. Expanding our network of sales and support personnel and being in close proximity to where our customers are is

a key towards achieving that initiative."

This agreement covers Australia, New Zealand, China, Hong Kong, Japan, India and the ASEAN countries. "We are particularly pleased and excited by this appointment as it provides an excellent opportunity for our existing and potential partners to work much closer with their customer base," said Brian Egan (MBS Channel Manager, Asia Pacific). "We were initially attracted to Paradigm Systems because Mercury Console is a solution for all Rocket U2 environments and is designed to lessen the load on System Administrators and Managed Services Providers alike." ■



Pavuk Systems Announces Redaktor IDE - In-Browser Code Editing

Pavuk Systems announces the release of Redaktor IDE. It provides support for editing of server-side code in a web browser.

Redaktor's UI is built upon CodeMirror and is fully integrated into the Pavuk Web Server. Source items are transferred to and from the

Redaktor window. Developers can even execute a compile on the server!

Redaktor supports BASIC and other languages with code-complete options. You may edit and remotely compile code from within the browser as well.

Colors are fully themed and may be changed to suit the users' preferences. ■



Pick Cloud, Inc Announces New Lower Minimum Price For OpenQM DBAAS

Pick Cloud, Inc. announced a new, lower minimum price for its OpenQM MultiValue DBaaS (database-as-a-service) product.

"It is now easier than ever for MultiValue customers to enter the cloud. We have reduced our minimum for OpenQM DBaaS from \$269 per month to \$199 per month. This includes an OpenQM license, AccuTerm, and the entire infrastructure necessary for a fully managed hosted environment. All the client needs is an Internet connection. No more purchasing hardware. Now their capital expenses (CapEx) become operat-

ing expenses (OpEx) which is another one of the many benefits of cloud computing," says Mark Pick, CEO of Pick Cloud, Inc.

Partnering with Google has allowed us to lower our prices and pass the savings onto MultiValue clients. So in addition to a lower minimum, clients also get Google's world-class security and infrastructure. ■



iTMS Uses Revelation Software to Stay Ahead of the Curve

OPTO Software, part of iTMS Software Pty Ltd, provides manufacturing inventory software, including ERP solutions, to a wide variety of industries. Customers span the fields of manufacturing, mining, civil, fabrication, and engineering as well as distribution, retail and wholesale, construction, and importing and exporting.

Based in Brisbane, Australia, OPTO has been providing ERP, material requirements planning, manufacturing and inventory software solutions to Australian businesses since the early 1990s, and in turn, these software solutions have enabled smaller manufacturers to close the techno-

FROM THE PRESS ROOM

logical and competitive gap between their business and larger-scale market leaders.

Used by hundreds of clients across Australia, OPTO prides itself on being small enough to be highly flexible but large enough to deliver the latest technology trends to its customers while shielding them from the underlying complexity. Providing steadfast support, OPTO seeks to guide customers through every phase of their installation, from evaluation and feature selection to technical support and helpdesk inquiries.

OpenInsight at the Core

"Our focus is on the manufacturing technologies you use to build things. Surrounding that, we also plug into accounting systems," said Jeremy Bolton, managing director. "Revelation Software's OpenInsight is the heart of our business; our whole product is based on it."

According to Bolton, a key differentiator for ITMS/OPTO is its deep understanding of the nuances of manufacturing. Revelation supports that agility, ensuring that OPTO platform is easily configured.

"We're very good at making software fit into many applications with the ability to deliver it quickly and cost effectively, while also making it easy to understand," Bolton said. "What sets us apart is the ability for the OPTO plat-

form to be easily configured to precise customer requirements."

The company's partnership with Revelation was forged after OPTO's CEO sought to build an effective and simplified system to solve manufacturing issues.

Ability to Adapt

Revelation has made OPTO's platform very adaptable. It's simple to build on and it supports both emerging and existing applications, according to Bolton.

Because it runs on a MultiValue database, OPTO can add new capabilities to its platform without affecting the data model.

"If someone needs something with Revelation, the architecture of our product enables us to add the new capability without changing the underlying data model of every other customer," Bolton said.

"Our software can fit in many applications."

MultiValue database technology can be quickly tailored to unusual requirements and it can talk directly to the CRM copy machine, Bolton explained.

"When we are talking to people about layout, normally we have to get spreadsheet, but this technology allows us to do a layout quickly and effectively," Bolton said. "Revelation gives us the flexibility we need when responding to customer requirements."

Built for the Future

Reliability and compatibility are also why OPTO has stayed with Revelation for more than 20 years. The technology helps OPTO support its customers by providing them with a platform that delivers better customer outcomes, quickly, and cost-effectively.

The MultiValue aspect of the platform makes it possible to

process requests from customers quickly and with a speed that relational database management systems can't provide, according to Bolton.

"Running Microsoft SQL Server or Oracle didn't make any sense," Bolton said. "There is nothing else you can find to handle so much data."

Additionally, the cost to customize the platform for a customer was "astronomical," whereas using Revelation makes the platform cost-effective for the client, Bolton explained.

However, if customers do need to utilize relational database management software, the OPTO platform can integrate to any part of the database that's necessary, he noted. ■



e-Xtra Newsletter

Stay on top of Industry News

- ◆ Tech Tips
- ◆ Job Postings
- ◆ New Products
- ◆ Corporate Updates



www.intl-spectrum.com/newsletter

What is JSON?

BY BENNETT BAROUCH

Expanding Your Toolkit is an occasional column explaining technology which can extend the reach of your system.

JSON (JavaScript Object Notation) — The name tells us that this is a way of notating the contents of a JavaScript object. As we will see, JSON is neither JavaScript-specific nor a full-featured object notation. So what's with the name, and more importantly, what is it really?

JSON was invented for use by a small group of JavaScript programmers who were much smarter about technology than they were about naming things.

By design, JSON is literally *XML Light*. Among other things, XML is the X in AJAX (Asynchronous JavaScript And XML). Douglas Crockford et al. wanted to use something simpler and smaller in the AJAX context. JSON is the result. You might say it's an application of the 80-20 Rule (or maybe the 99-1 Rule). JSON is not as feature-rich as XML, but it adequately supports a huge range of common use-cases more elegantly. Indeed, it may even be a good idea to force more sophisticated datasets into JSON's world, simplifying interfaces while lessening training and support costs for a service that then only requires its clients to handle JSON instead of XML. At the least, it cuts down on the complexity and size of the payload being produced, sent

...there are JSON generator-parser pairs available for most web-oriented languages...

over the internet, and consumed — Crockford's goal.

It Supports Less Than JavaScript

If a JavaScript object has methods (which is very common in practice), they silently just do not show up when that object is expressed in JSON (via the definitive generator `JSON.stringify()`). This seems to reflect the fact that the use-cases Crockford et al. had in mind did not include transferring code between browsers and back ends, so it was a *feature* to ignore methods, automatically restricting payloads to items of value. Minimizing payload was a key objective.

There are other data types (maps, sets, and more) and special values (`Infinity`, `NaN`, and `undefined`) in JavaScript that also silently fail to be represented in JSON. You can, of course, create your own non-standardized, non-automatic, application-specific (or library-specific) encoding and decoding for *anything*, but that is something you may do *with* JSON, not something that JSON represents or that its standard generators and parsers do for you.

Perhaps this is because the backends, with which browsers communicate, are rarely implemented in JavaScript, and universal, automatic cross-language translations of uncommon items is a creature best left unprovoked. One can argue that if you need these things, then it *should* be application level code that deals with them in each language's own idioms.

It Supports More Than JavaScript

Oddly, JSON allows some characters in strings that are not allowed in JavaScript.

More importantly, JSON is not in any functional way linked to JavaScript — it's just badly named. JSON has proved so useful and widely used that there are JSON generator-parser pairs available for most web-oriented languages, including Perl, PHP, Python, Ruby, Java, ASP, C#, etc. Since the syntax is simple and the number of supported data types is small, creating a generator-parser pair is a modest programming task in any language for which a set does not already exist. (See this article on the *International Spectrum* website, for example.)

In practice, JSON is a language-independent data container used for more browser back-end and web service traffic than any other and is also used for configuration files and other storage purposes.

**Celebrating 20 Years
as a leader in the
MultiValue Industry**



D3 UniVerse UniData PICK jBase mvBase Caché

IT Solutions. Proven Results.

One Project. One Decision. One Keystroke at a time.

- Custom developed solutions based on your needs
- Senior level developers and business analysts to guide you
- Developing long term partnerships

**www.pickprogram.com
contact@pickprogram.com
(614) 921-9840**

The Data

JSON allows for six kinds of data to be expressed. *Numbers* are integers and floats (with no distinction between them). *Strings* are runs of zero or more Unicode characters inside quotation marks. Booleans are `true` or `false`. *Null* is a data type with a single possible value of `null`. *Arrays* are collections of arbitrary elements, such as `["two", 1, { "pi" : 3.14159 }]`. *Objects* are associative arrays, each element being an NVP (Name-Value Pair). For example `{ "dbType" : "Pick multi-value", "born" : 1965 }`. As shown in these examples, array and object elements do not need to be of like type.

The Syntax

Strings are delimited with quotation marks (and not by apostrophes or so-called single quotes). Strings can contain problematic characters via escape sequences begun with a backslash (`\`). An array is indicated by square



**taking multivalue ...
where it has never been before**

- Close compatibility with most other multivalue environments
- Easy migration process
- Maintenance-free file system for ease of use
- High quality documentation
- QMClient API for development of GUI and web applications
- Low licensing cost
- AccuTerm bundled at no additional cost
- Many unique features

Ladybridge Systems Ltd

17b Coldstream Lane, Hardingstone, Northampton, NN4 6DB, England

Worldwide distributor: Zumasys, 9245 Reasearch Drive, Irvine CA 92618, USA

www.zumasys.com

www.openqm.com

brackets ([]). An object is indicated by curly braces ({ }). A colon (:) is placed between a name and the its value. Names themselves are strings, not unquoted tokens. A comma (,) is used to separate adjacent array elements or object properties. Empty arrays and objects are allowed, as is nesting to any depth. Whitespace outside of quotation marks can and should be used to maximize human readability.

There is something to be said for a structured data format that can be completely defined in two paragraphs and serves so many actual use-cases that it dominates data exchange activity on the web!

Example

Let's look at a simple example [Figure 1]. This object holds information about a well-known duck.

Comments

When used for configuration files, the fact that JSON does not support comments is a drawback. One typically wants to annotate a config file with the reason each value there was chosen, and what factors should govern setting a different value. You can easily implement a file reader that strips comments and then feeds what is left into a JSON parser. Consider the following illustration of an obvious possibility that is trivial to implement [Figure 2].

Help is on Hand

As simple as JSON is, when you are new to it, using an online resource such as jsonlint.com will tell you when you have it right, or help you figure out where you went wrong. **IS**

BENNETT BAROUCH has over 30 years of industry experience spanning design auto-

mation for integrated circuits deployed in satellites, financial portfolio software, high transaction volume and big data systems, information management, secure on-site and mobile networking, and IT operations software.

Groundbreaking work under Bennett's leadership produced a virtual assistant that could understand 20 million English phrases and respond with a wide array of information and with complete computer-telephony integration. This work was made part of the permanent collection of the Smithsonian Institution, for Outstanding Achievement in Information Technology, 14 years before Apple released Siri and became the basis of the OnStar virtual assistant found in GM automobiles. Bennett has been certified in ITIL, and as a Scrum Master and as a Scrum Product Manager.

```
{
  "first name" : "Donald",
  "last name"  : "Duck",
  "address"   : {
    "home"    : {
      "line1"  : "123 Maple Street",
      "line2"  : null,
      "state"  : "CA",
      "city"   : "Los Angeles"
    },
    "work"    : {
      "line1"  : "Disneyland Resort",
      "line2"  : "1313 S Harbor Blvd",
      "state"  : "CA",
      "city"   : "Anaheim"
    }
  },
  "anthropomorphic" : true,
  "birthdate"       : { "month" : "June", "day" : 9, "year" : 1934 },
  "military status" : [ { "army" : "retired" }, { "navy" : "reservist" } ],
  "family"          : { "nephew" : [ "Huey", "Dewey", "Louie" ] }
  ...
}
```

Fig. 1

```
{
  "version" : 1.7,
  "timezone" : "America/Los_Angeles", // used in error logging --
                                           // set to match server's time zone
  "resource" : "http://coolplace.com/resource", // the only tricky bit to parse
  ...
}
```

Fig. 2

IS.HASH.MD5

Generating with UniBASIC DIGEST Command

The MD5 message-digest algorithm is a widely used cryptographic hash function producing a 128-bit (16-byte) hash value, typically expressed in text format as a 32-digit hexadecimal number. MD5 has been utilized in a wide variety of cryptographic applications and is also commonly used to verify data integrity.

While not as secure as SHA1, it is still used in many places for data integrity, version control, and other features that need unique one-way signatures.

MD5 Function

MD5 processes a variable-length message into a fixed-length output of 128 bits. The input message is broken up into chunks of 512-bit blocks (sixteen 32-bit words); the message is padded so that its length is divisible by 512. The padding works as follows: first a single bit, 1, is appended to the end of the message. This is followed by as many zeros as

are required to bring the length of the message up to 64 bits fewer than a multiple of 512. The remaining bits are filled up with 64 bits representing the length of the original message, modulo 264.

```
MD5("The quick brown fox jumps over the lazy dog")
```

```
9e107d9d372bb6826bd81d3542a419d6
```

MD5 will detect even small changes in the string and cause the returned hash value to change. You can always read more about MD5 at:

<https://en.wikipedia.org/wiki/MD5>

UniBASIC Hashing

UniBASIC comes with hashing extensions in the form of DIGEST(). The DIGEST has SHA1 built-in. This allows you to use code like [Figure 1].

```
MD5.MESSAGE = "The quick brown fox jumps over the lazy dog"
ret = DIGEST('MD5',MD5.MESSAGE, 1, SHA1.HASH)
CRT SHA1.HASH : ` = 9e107d9d372bb6826bd81d3542a419d6 `
```

Fig. 1

```
MD5.MESSAGE = "The quick brown fox jumps over the lazy dog"
CALL IS.HASH.MD5(MD5.MESSAGE,HASH.VALUE)
*
TEST.VALUE = "9e107d9d372bb6826bd81d3542a419d6"
CRT HASH.VALUE : " =" : TEST.VALUE : " - "
IF (HASH.VALUE EQ TEST.VALUE) THEN
  CRT "Ok"
END ELSE
  CRT "Failed"
END
```

Fig. 2

```
9e107d9d372bb6826bd81d3542a419d6 = 9e107d9d372bb6826bd81d3542a419d6 - Ok
```

Fig. 3

The subroutine found with this article is based on the code above, but has been structured to be interchangeable with non-UniBASIC versions of the MD5 subroutine also found on the International Spectrum Website.

Example

See [Figure 2].

Output


See [Figure 3]. **IS**

MARKETPLACE

ACCOUNTING

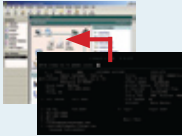
Natec Systems

www.natecsystems.com | nrector@natecsystems.com




QuickBooks API for the MultiValue Database

- Read/Write Directly to Quickbooks Databases
- mvQB API is Designed for the MultiValue Program to Use
- No Need to Learn the Internals of QuickBooks
- QuickBooks Pro/Premier/Enterprise



NATEC Systems
Providing Solutions to your MultiValue Questions



Phone: 303.465.9616
E-mail: mvqb@natecsystems.com
Website: www.natecsystems.com

COMPLIANCE

SJ+ Systems Associates

www.sjplus.com | sjoslyn@sjplus.com

CONSULTING

Drexel Management Service

www.drexelmgmt.com | dconboy@drexelmgmt.com

Execu-Sys, LTD

www.eslny.com | mh@eslny.com

HDWP

www.HDWP.com | results@HDWP.com

Modern MultiValue, LLC

www.ModernMultiValue.com | info@ModernMultiValue.com

PICK Programmers Shop

www.pickprogram.com | brian@pickprogram.com

Precision Solutions

www.precisiononline.com | Kevin@PrecisOnline.com

DATABASE

Ladybridge Systems Ltd

www.ladybridge.com | sales@Ladybridge.com

REPORTING

Brian Leach Consulting, LTD

www.brianleach.co.uk | brian@brianleach.co.uk

TERMINAL EMULATOR

Zumasys

<http://www.zumasys.com/products/accuterm/>



AccuTerm® software, the leader in terminal emulation, allows you to access your MultiValue application—whether it is on-premises or in the cloud—from any Windows device.

www.zumasys.com/accuterm
818-951-1891



LETTERS TO THE EDITOR

Have an opinion on an article: Agree, disagree, or enhancement to an article from a previous issue? International Spectrum and our authors are interested in hearing from you!

E-mail: editor@intl-spectrum.com

WANT TO SEE A SPECIFIC TOPIC?

International Spectrum is looking for writers, feedback, and topic ideas. We all have specific topics and issues that we need answers to find solutions for. Send us an E-mail with topics you would like to have covered in the magazine or on the website.

E-mail: nathan@intl-spectrum.com

WANT TO WRITE?

Expand your professional credentials, and provide us with an article.

Give us a rough and ugly outline, and we will help you refine it, proof it, and make it press ready. Or you can give us something polished, proofed, and press ready to publish.

Share your thoughts and expertise with over 10,000 fellow MultiValue developers and users.

E-mail: editor@intl-spectrum.com

NEED A MENTOR?

Mentors give developers the ability to ask industry experts for direction, code examples, and/or just ask them to see if something makes sense. Sometimes, all you need is a resource or example to start or complete a project.

Check with us to see who is available for mentoring, and how you can take advantage of it to save your business or company money.

E-mail: nathan@intl-spectrum.com

WANT TO BE A MENTOR?

We have many retired or semi-retired professionals out there that would love to share their knowledge of MultiValue development. If you are one of them, please contact us to see what mentoring is all about.

E-mail: nathan@intl-spectrum.com

AI and DSS

Part 2

BY CHARLES BAROUCH

Dystopia is the term used for any number of bleak, failed societies. In Science Fiction, computers, robots, and evil algorithms – AI (Artificial Intelligence) and DSS (Decision Support Systems) being two examples – are often to blame. The emphasis is generally on dehumanizing us by reducing our society to the data which describes it. Personally, I blame SQL.

To be serious, SQL is designed around the idea that optimizing data means optimizing the digital use of data. It is not designed for you, my organic friend.

What if we take a more human approach to data? Well, if you want data that has a human touch, we have plenty of options. NVP (Name-Value Pairs) offer a readable – human readable – label with each jot of data. Formalize that a bit and you are in the realm of XML and JSON. These three are certainly not machine optimized. They are people-centric, focusing on

For some jobs, the rules of SQL are the most reasonable ones.

clarity to the reader over mathematical minimalism. If the AI uprising is your fear, your best defense is to skew the rules toward... well... us.

What About MultiValue?

MultiValue sits in the middle. I once heard, and often quote, Mike Ruane as saying that MultiValue is compressed XML. We use positions instead of labels, but we bring a structure that is more eyeball friendly than SQL. Think of it this way: I have to transform SQL data to share it. It has to become tab-separated, or XML, or some other decidedly non-SQL thing before it can move. Generally, this isn't just swapping columns for commas. SQL data is spread out and has to be unified and essentially re-architected before it can be transportable.

Given that moving data, dissecting data, and assembling data is a big part of what we do, having a database that can't do any of that easily is an odd choice. Unless, of course, you are in the thrall of the metal ones. MultiValue pays attention to speed, but it also has its bags packed at all times. Any modern developer who can tease data out of a comma-separated file can handle a string with @AM delimiters. Tell them the @VMs are embedded sub-strings and they'll probably be just fine with those delimiters as well. If we must dress it up for travel, subbing @AM to comma and @VM to pipe is often enough. When it comes to speed, the less we handle the data, the faster we can ship it.

Thingedness

XML, JSON and MultiValue also have another critical edge when it comes to readability: *Thingedness*. This is the term I coined to describe the ideal relationship between data and the user of the data. Here's where columnar data-

PICK PROFESSIONAL

Don't make the mistake of placing your career in the hands of just ANY search firm. We are professionals who understand the importance of CONFIDENTIALITY and RESPECT of a job search, and our database of clients is one of the largest in the country. Unlike the rest, we will work in YOUR best interests' to help you further your career. Because of our dedication and professionalism, we are recognized as the leaders in the PICK/UniVerse/Unidata placement industry in the Tri-State area and throughout the U.S. So if you are tired of putting yourself at the mercy of the rest.

CALL THE BEST! Contact...

Matt Hart

EXECU-SYS, LTD

1411 Broadway, Suite 1220

New York, NY 10018

(212) 967-0505

(800) 423-1964 x 302

Fax: (212)947-8593

Email: mh@eslly.com

Consultants Needed Nationwide

bases and SQL databases fail the thingedness test: Can you point to a single record and associate it with a common, real world, thing? My XML, JSON, NVP, or MultiValue INVOICE file can have the entirety of an invoice in each record. One read equals one invoice. That's something a non-database person can grasp: one hundred invoices equals one hundred records.

While there are reasons to not do this — many excellent reasons — the closer your data gets to this model, the easier it is for the programmer, the user, and the architect to keep the entire data model in their head. As you approach thingedness, you approach clarity of concept. The data world has more in common with the human one.

With XML, JSON, and MultiValue, thingedness is achievable. The big difference between the three is that the

first two have to be transformed to be used. MultiValue can chose to unpack its bags, but a MultiValue string is always ready to work.

Some of the Excellent Reasons

SQL is the extreme counter-argument to thingedness. It is based on the premise that the more you break something down, the better you can control it and account for it. There is merit to this approach if you are concerned with scaling up the size of your data. However, the more the complexity of your data scales up, the worse this idea becomes. There is a reason Google uses NoSQL to manage search. There is a reason that Facebook uses NoSQL

Still, SQL's popularity isn't random. For some jobs, the rules of SQL are the most rational ones. A good example is tool building. It is easier to generalize a tool, for reporting or analytics, when

THEME-THOLOGY: INVASION

Voices by Lisa A. Kramer

I Was a Teenage Alien by LJ Cohen

Singularity by Jeremiah Lewis

Not Like Us by Mike Reeves-McMillan

That Kind by Charles Barouch

Yellow by Bill Ries-Knight

An Invasion of Ideas by Jeremy Lichtman

Famine, with Fries by Jefferson Smith

The Several Monsters of Sainte-Sara-la-Noire by Michael Williams

Going Viral by Rachel Desilets

Dead Planet Scrolls by Timothy Hurley

Red Vapor by Michaela Susanne

The Worms Crawl In by Michelle Mogil

Nano Nation by CM Stewart

The Woods, The Cellar, and Cover art by Aaron Wood

All other interior art by Juan Ochoa

WWW.THEME-THOLOGY.COM



all data has a rigid uniformity of storage. The less creative the structures are, the easier it is to make new tools.

Moreover, forcing the table designer to specify field types and lengths helps keep the design focused on the use and intent of the data. Free-form data can often result in sloppy design. Working in SQL makes me a better NoSQL architect.

So, please don't damn the methodology out of hand. It has its place. Not every place, but I wouldn't want a pure thingedness database, either.

Where the Pendulum Stops

What we are looking for here is an acceptable level of atomicity. Simply put, we want to break things down just enough.

The middle is where the winners want to be. Reasonable control, but not the OCD of SQL. Reasonable thingedness, but not a rigid mandate to mirror the structures of the world. SQL doesn't do middle. Columnar doesn't do middle. XML and JSON can do middle, but they can't be operated upon directly for complex tasks.

MultiValue can do middle. We can create an invoice header record, with unified data, and split the details, each

...the closer you put the data interaction to the data, the better your speed.

to their own record. We can keep multiple values in the header efficiently: Three contact names? No problem. Only one on the next one? No wasted space.

This is the balance between the AI/DSS view of data and the human view. We can scale in complexity because we can make decisions in our architecture and applications to treat elements of our data in sane ways.

How Does This Relate to AI and DSS?

As you saw last issue in the Animals program, we needed to construct the growing AI data in a way which favors decision trees. The less efficiently we implement, the slower our program will get as it matures. The infant version, the one with just a few started animals, will always be faster than the adult, with its extensive zoo, but here we aren't worried about relative speed, we are worried about being fast enough to keep the user feeding the program.

The game Animals doesn't grow if no one plays.

As Nathan discusses in another article in this issue, the closer you put the data interaction to the data, the better your speed. Additionally, as my dad would point out, the more parts, the more that can break. Keeping the programming close to the data requires fewer transformations, less network bandwidth, and fewer steps. That makes it faster and less fragile.

When we implement DSS or AI, we are talking about extensive data. If we are being really smart about it, we are also expecting that data to keep growing. Real AI and DSS should eventually perform successfully outside of the original parameters. If you planned everything it does, it is more of a performing bear than a critical thinker.

Your choice of data storage matters. Your choice of programming language matters. With enough time, trouble, effort, and money, you might be able to make a pig sing, but starting with a singer is probably a wiser move. Understanding the underlying effects of your choices raises you above decisions like, "Well, it was the only language I knew, so I wrote everything in Whitespace." Picking tools responsibly? That's real intelligence. **IS**

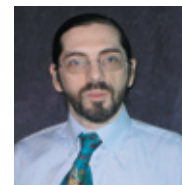
IT audits have you jumping through hoops?



PRC can help you meet your compliance requirements and make IT more agile and productive. No extra work, nothing to remember, nothing to fall through the cracks. Our software development lifecycle tool automatically prevents or detects change according to your criteria. You can deploy, rollback, test and report quickly, automatically and with confidence. Let PRC protect your company's valuable U2 data and software assets.



Sj+ Systems Associates • info@sjplus.com • <http://sjplus.com>



CHARLES BAROUCH is the CTO of HDWP, Inc. and the Publisher at HDWPbooks. You can read his writing in

International Spectrum, Theme-Thology, Novo Pulp, Pax Solaria, PerhelionSF, and the Interrogative series, which begins with Tiago and the Masterless.