Out in Left Field
strategic tools for strategic teams
Volume One, Issue Seven

SOA, Speed and DB2

Something old, something new, something borrowed, something blue.

What does this "wedding saying" have to do with Service Oriented Architecture (SOA)? Before we answer that question, let's explore what SOA is all about and why it seems to be the technology du jour.

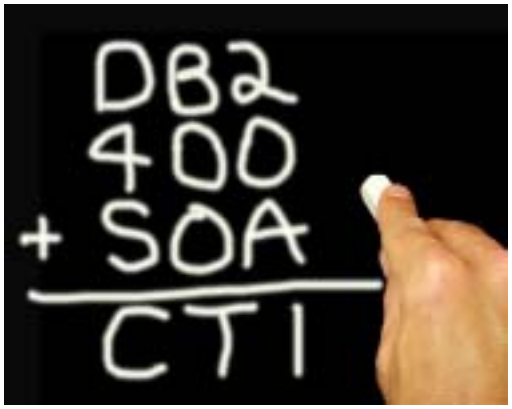
At its core, SOA is an approach to building applications from "services." In the context of SOA, these services are software components with well defined interfaces that are platform and language neutral. In theory, this architecture means applications can be assembled using a bundle of loosely coupled services so they can be quickly developed and quickly adapted to changes in business processes and the external environment.

Something old

All "new" technology concepts borrow heavily from preexisting ideas. SOA is no different.

The objectives of an SOA are similar to those of object oriented (OO) programming and Java – hidden implementations and platform independence. SOA takes these existing concepts to a higher level than OO design and Java by defining a component-level interface and leaving the implementation of the service up to the service provider. The service itself could have any number of languages, platform, or design paradigm. In fact, a service could be completely rewritten with a new technology base and the consumer of the service should not know the difference.

Secondly, many services will originate from existing software. With many billions of dollars invested in current technology, the first services will be built utilizing software that has already proven its worth.



Finally, a database is the foundation of all application architectures. Databases like DB2 will continue to play a key role in retrieving data, storing transactions, and providing the storage backbone required by robust software.

Something new

The new aspect of the SOA will be less theoretical and more practical. While an application does not have to be implemented with a particular technology to fit the SOA mold, most services will be built using XML and Service Oriented Architecture Protocols technologies to define the interface and flow between the consumer and producer of a service.

XML is simply a text-based way to define data. The primary use of XML has been to add meaning to raw data elements. SOAP is a messaging protocol that uses XML to provide communications between different pieces of software. These relatively new technologies provide the mechanism for a service consumer and a service provider to communicate in a platform and language independent manner.

Something borrowed

While SOA proposes a new approach to componentizing software and abstracting its implementation, IT departments will have responsibilities and challenges similar to those of today. In general, many of the tools and techniques used to monitor, manage, and performance tune applications will apply to an application built with an SOA.

The good news is that the skills and software tools in use today can be transferred to these new applications. Of particular interest is the need to ensure that database performance is good. If they get it right, companies should be able to consolidate multiple function providers and 'single source' services instead of having duplicate software across the company. This will mean that the remaining software stacks will need to scale up to more users. If the service is made available to external customers or business partners, there will naturally be an increased focus on the responsiveness of the software. Since database access is often the critical part of response time it will be increasingly necessary to ensure fast SQL performance.

Something Blue

In a recent article about SOA, a number of companies were listed as endorsing this style of application development. The company most identified with SOA however, was Big Blue. IBM has put a significant amount of technical and marketing effort behind SOA concepts and implementations. We are likely to see this continue throughout 2006.

(Continued on page 2)

Inside This Issue:

A Beautiful View
Part 3 of 4 - Page Two

Customized QAQQINI file for specific ODBC/JDBC users
Page Three

Reader Input
Page Four

Part 3 in a Series of 4

A Beautiful View

In our last newsletter we talked about how to use SQL views to hide table relationships with joins and how to turn cryptic data into human readable values. Both of these techniques allow end-users to focus on the answers they are trying to obtain instead of the mechanics of getting those answers.

This edition will discuss the use of views to do some basic formatting and how to use views to provide another level of data security.

Format away

Much of the effort to produce any report is getting the formatting right. One of the challenges with good layout is introduced because the data is in a very granular form in the table. For example, a phone number might be broken up into country code, area code, the first three digits, and the final four digits. This is done to make data validation easier and to enable very granular data analysis. When a report is written however, we want to recombine this data so that it has a compact and readable layout.

Let's say for example, you have a table that contains a person's first name, middle name, and last name. When these names are put into the table, they are put into fixed length fields and blank padded. In your report you want to have the following format:

Last_Name, First_Name Middle_Name

To accomplish this type of formatting, you could do the work in the report writer itself, the SQL used to retrieve the data into the report writer, or in an SQL view. If this is the format you want for a variety of different reports, then it makes the most sense to put the logic in a view since it could be reused by all the reports that want this format rather than having to copy the same logic to multiple reports. Essentially, the idea is to push the most common formatting (or calculation) logic into the view to make it widely available and avoid the time and effort to replicate it. This also makes it easier to change if there is a new standard for the formatting in the future.

To do the formatting as above, you will need to use the SQL functions of TRIM and CONCAT.

```
SELECT
CONCAT(
CONCAT(
CONCAT(TRIM(Last_Name) ,', '),
CONCAT(TRIM(First_Name),' ')),
TRIM(Middle_Name))
FROM customer
```

In this example, the TRIM function gets rid of the trailing blanks on the name columns and the CONCAT function puts the shorter strings together with the necessary punctuation and the needed blanks.

Security hiding

One of the least used aspects of SQL views is the ability to secure data. As discussed in previous articles, you can simplify the database for end-users by simply leaving out unnecessary columns in the definition of the view. You can also use this same technique to restrict columns that some users should not have access to. Lets say, for example, you do not want most users to have access to a person's social security number. By excluding all users from the base table, you have blocked access to the data except by those users who are explicitly authorized by private authorities. To grant specialized access to the table, you can then create a view with a subset of columns and then give explicit authority to the users that need to query or generate reports on the data.

In addition to column security, you can also restrict access to specific parts of the data (i.e. row security). For example, you may want your western region managers to only access western region sales information. To accomplish this, you simply define a view that filters out all data but the western region's information. For example:

```
CREATE VIEW WestRegion
      (CUST_ID,
       CUST_BALANCE,
       CUST_NAME,
       CUST_CREDIT,
       SALES_REP,
       SALES_COMMISION)
AS
SELECT CID, CBAL, CNAME, CCREDIT,
CCREDIT, SREP, SCOMM
FROM CUSTMAST
WHERE CREGION = "West"
```

Once the view is created, then the view's authority can be granted to the western region managers (or the group they belong to) and you have made sure they can only see the columns and the rows that they are authorized to view.

(SOA, Speed and DB2 - Continued from page 1)

Summary

SOA is more of a concept than a set of concrete technologies. These fundamental concepts of technology independence and loosely coupled software have a lot of promise, but will be based on a lot of existing software for quite some time. Furthermore, from an IT perspective, the skills required to manage these new applications will remain very similar to those required for applications that are already deployed. While SOA introduces some new alphabet soup into the IT world, the need to monitor and tune the backend database will not only continue but increase in importance.

Customized QAQQINI file for specific ODBC/JDBC users

Based on positive feedback from an article dealing with configuring specific ODBC/JDBC connections to run under a different subsystem, I decided to write another article aimed at helping iSeries system administrators whose business or technical circumstances require even *more* customization.

Not too long ago, one of our customers called in explaining that his SQL users hitting the iSeries via the ODBC/JDBC interface were experiencing problems. IBM diagnosed the problems and recommended he change the QAQQINI IGNORE_DERIVED_INDEX to *YES (in base V5R3, but on V5R2 only available via MF31320, SI10287, SI10293 PTFs)

I don't know the details behind IBM's recommendation and can only guess that they wanted to force more queries to use the new SQE optimizer rather than the old CQE optimizer. By default, the query dispatcher will send the query to SQE but once it diagnoses that there are keyed select/omit logical files on the physical file (a.k.a. derived indexes), SQE will send it back to CQE. This wastes optimization time and narrows down the amount of SQL that can take advantage of the new, enhanced SQE optimizer.

It's simple enough to change the QAQQINI file. When it's residing in QUSRSYS it behaves as a SQL query optimizer system value repository so all he'd need to do is:

1) **STRSQL**
2) **INSERT INTO QUSRSYS/QAQQINI VALUES ('IGNORE_DERIVED_INDEX', '*YES', 'Force more queries to SQE by ignoring S/O logicals')**

Now, **ALL** jobs running SQL will enforce this new setting.

This is exactly what the customer didn't want (all jobs). His problems were only being experienced by interactive users running SQL via the ODBC/JDBC interface, but not by all jobs on the system. Furthermore, problems were being experienced only by specific power users, not all ODBC users. Accordingly, his objective was to only enforce this change for jobs running on behalf of these specific users. I agreed with him 100%. By narrowing down this change, he reduces the risk of breaking functionality or otherwise negatively impacting other users and business processes on his system.

I was able to help this customer narrow down this change for ODBC/JDBC jobs alone AND specific users and am going to share exactly how it was done.

First of all, we'll need to create a new copy of the QAQQINI file, so we can point the ODBC/JDBC jobs to use this customized file and not the one in QUSRSYS library. Creation of a new QAQQINI file has to be done via the CRTDUPOBJ command, as there are some triggers on the original QSYS file that have to migrate along with the QAQQINI file. It's your choice whether you want to duplicate the master version in QSYS library or the system administrator created copy in QUSRSYS (it may have been created years ago).

For this exercise I'll use master version:

- **CRTDUPOBJ OBJ(QAQQINI) FROMLIB(QSYS) OBJTYPE(*FILE) TOLIB(QGPL) DATA(*YES)**

Next, we need to insert the IGNORE_DERIVED_INDEX setting as demonstrated earlier in the article:

- **INSERT INTO QGPL/QAQQINI VALUES('IGNORE_DERIVED_INDEX', '*YES', 'Force more queries to SQE by ignoring S/O logicals')**

A modified QAQQINI file now exists, but who's using it? No one. Not until we use the CHGQRYA command and force a job to use it. Unfortunately, CHGQRYA only works for a single, running instance of one job. We want all ODBC/JDBC jobs to automatically use this customized version of the QAQQINI file.

Before I address this challenge, I need to create the configuration file that will contain a list of users for which we want to implement the QAQQINI change. All this involves is:

1) STRSQL
2) CREATE TABLE QGPL/USERLIST (USERNAME CHAR (10) NOT NULL WITH DEFAULT)
3) INSERT INTO QGPL/USERLIST VALUES('ELVIS')
4) …… X number of inserts as in step 3, for each user you want to implement the change for ….

NOTE: I have used ELVIS just for illustration; you'll obviously need to enter all users for which you intend to implement customized version of QAQQINI.

To solve the limitation of CHGQRYA affecting a single job only, I am going to resort to one of the database exit points provided by the iSeries. For those unfamiliar with exit points, they're simply hooks provided by the OS and once the program is registered within it, the system honors the registration by invoking that exit point program every time a certain event happens. In case of SQL database server exit points our exit point program will be invoked each time a new connection starts.
I mentioned that we'll need an exit point program so let's create one using simple CL.

```
PGM (&alwAccess &input)
DCL &alwAccess *CHAR  1
DCL &input     *CHAR 32
DCL &curuser   *CHAR 10

DCLF       FILE(QGPL/USERLIST)
```

(Continued on page 4)

Reader Input

A couple of issues back in Centerfield's "Out in Left Field" newsletter, you wrote an article about separating ODBC/JDBC Connections by IP address. I wanted to let you know that we have implemented this change and it works PERFECTLY!!!!!! We have been looking for a way to have a web applications come in at a different run priority and memory pool.

I was in Atlanta two weeks ago when I finally got tome to really read the small article and understand it. I implemented it on a couple of our systems and we now have a view over the page faulting of web applications versus non web applications. Keep up the good work on the articles.

Ken K., Systems Programmer

Correction: October 2005:

You might want to double check page 6...the JLF3 example should be in the "Right Outer Join" section of the article (and the JLF2 example should be where the JLF3 example is).

Mike K., Senior System Analyst

(We did Mike, thanks for bringing it to our attention!)

Very helpful... Very useful information.... looking forward to the next issue!!!!!

Bob N., DBA

(Customized QAQQINI file for specific ODBC/JDBC users - Continued from page 3)

```
RTVJOBA    CURUSER
(&CURUSER)

READLOOP:    RCVF
             MONMSG MSGID(CPF0864) EXEC(GOTO CMDLBL(EOF))
              IF (&curuser = &username) THEN(CHGQRYA QRYOPTLIB(QGPL))
             GOTO readloop
EOF:
ENDPGM
```

Create OURQAQQINI program out of the source above:

CRTBNDCL PGM(QGPL/OURQAQQINI) SRCFILE(QGPL/QCLSRC)

As you can see, this CL program is very straightforward. All it does is retrieve a current profile name for the job, compares it against the profiles in the USERLIST file and if a match is found, points the job to the customized QAQQINI file in QGPL library. The final step is registering our newly created program with the iSeries registration facility:

ADDEXITPGM EXITPNT(QIBM_QZDA_INIT) FORMAT (ZDAI0100) PGMNBR(1) PGM(QGPL/OURQAQQINI)

One thing to note is that presently active ODBC/JDBC jobs (i.e. QZDASOINIT) will not honor new registration entry until they're recycled (ended/restarted). You can let natural prestart job progression take care of that (default reuse count of a PJ is 200), or force all jobs to restart by ending the *DATABASE host server, ending active prestart jobs (ENDPJ command) and restarting *DATABASE host server. Other processes like ending the subsystem, restricted state or IPL will guarantee all jobs are invoking the newly registered exit point program as well.

So what have we accomplished with all this?

We now have **dynamically configurable control** via the USERLIST file over which version of the QAQQINI file should ODBC/JDBC connections use. If an occasion rises in the future to do something similar, you can simply insert another entry into the customized QAQQINI file and all configured users will pick up the change automatically.

For example, I've seen an issue where a shop was using journaling and reading the journal to implement some other changes (i.e. replication). With V5R3 IBM has enhanced DELETE FROM processing where they'll resort to CLRPFM when appropriate and possible. As you may already know, CLRPFM is very fast as the system doesn't have to worry about rolling back the changes. If it does resort to fast delete, then instead of a delete entry in the journal, your program will only see a clear physical file member entry in the journal. How do you roll that back when you don't have the before image in the journal? One way to revert to old behavior is to insert SQL_FAST_DELETE_ROW_COUNT in the modified or system wide QAQQINI file and set it to *NONE. This will guarantee that OS never attempts fast delete (CLRPFM).

Don't you love to dig under the hood and customize your iSeries? I do!! Yeah, yeah, I know I'm on a power trip!

y IBM to register for an online conference:

MMARY

==========

d Speed? Fast database access has it role in DB2.

Date: 12/06/2005
Start Time: 10:00 AM US/Central
Duration: 60 minutes
Presenter: Mark Holm

CONFERENCE DESCRIPTION

==========

Service Oriented Architecture: The role of DB2

NEED SPEED? Fast database access has its role in SOA. This presentation will discuss the key ideas behind Service Oriented Architecture and the key role for DB2 in delivering on the promise of modularized and flexible applications. Of particular interest is the need for scalable, stable, and fast database access as the backbone of a service oriented architecture.

ABOUT MARK HOLM

==========

Mark Holm is founder of Centerfield Technology. He has held various management and technical positions in his 20-year career in the computer industry with IBM, Showcase Corporation (now a division of SPSS), and Centerfield Technology.

At IBM, Mark has been member of the AS/400 system design group, helped formulate strategy for DB2, developed key parts of the query engine, was a key designer for many DB2 features including referential integrity, and has been involved in many performance improvements to DB2 and SQL. Mark has received two Outstanding Technical Achievement Awards, six informal awards, two software patents, and published numerous technical disclosures. Mark is a graduate of the University of Idaho in computer science with a minor in mathematics.

REGISTRATION

==========

To register for this event, click the link below:

https://whiteglove.on.raindance.com/confmgr/event_register.jsp?eventId=4830&invitationId=27037

Once your registration has been accepted, IBM Event Host will send an email containing details to join the conference.