132

DB2 update

October 2003

In this issue

DB2 Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1999 issue, are available separately to subscribers for £22.50 ($33.75) each including postage.

DB2 Update on-line

Code from *DB2 Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/db2; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £100 ($160) per 1000 words and £50 ($80) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £20 ($32) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

DB2 Version 8 – don’t be held back by earlier limitations

IBM has made many fundamental changes in DB2 Version 8 on z/OS to enhance performance, availability, scalability, and security. It radically changes – and, in most cases, nullifies – the old rules based on past experience.

With previous versions you would have created backward indexes, modified column attributes to match the table your join uses, changed VARCHAR to fixed length strings, and spent hours matching your copy books with DB2 column attributes, to meet your performance needs. While gaining on the parallelism offered by partitioned tablespaces, you would have been constrained by its inflexibility. You would have been forced to code multiple SELECT statements accessing the same tables, because DB2 didn’t allow you to code the SQL the way you wanted.

DB2 V8 offers you a chance to relieve your applications from these constraints and overheads and focus on your business needs. This article highlights the major breakthroughs in eliminating many of the earlier limitations in DB2.

ABILITY TO INDEX PREDICATES WITH MISMATCHING DATA TYPES

As a good designer, you were expected to ensure that the data types of the table columns matched your program’s host variables so that you benefited from performance gains.

Though easily enforced in a COBOL-based development environment, this becomes difficult with more and more application development happening in C or Java. C doesn’t support the DECIMAL data type, and in Java every string is of variable length, whereas in DB2 fixed-length character columns are used.

Before DB2 V8, if the data types of the predicate operand did not match, it became a ‘stage 2’ predicate and had a performance overhead. With DB2 V8, the optimizer allows predicates with mismatched data types and lengths to be processed as ‘stage 1’ predicates.

This also enables you to join tables on columns that have different data types and lengths without facing a performance overhead.

AVOID SORTS USING BACKWARD INDEX SCANS

Prior to DB2 V8, you would have created two indices, ascending and descending, on the same columns if your application needed to traverse the data, based on the column, in both forward and backward directions.

Typically, backward index is used to get the MAX value for generating the next sequence number, and forward index is used for processing all the records. If you didn’t have a backward index, DB2 had to do a sort in order to return the MAX value or to handle an ORDER BY DESC clause.

DB2 V8 is capable of traversing an existing index in a backward direction, and doesn’t need an additional backward index for this. Now, you can delete the additional descending indexes and get better performance in terms of reduced index maintenance, as well as making storage gains.

SUPPORT FOR STORED VARIABLE-LENGTH INDEX KEYS

In DB2 V7, if you create an index on columns of varying lengths, the index keys are padded to their maximum length – occupying more storage. Also an index-only access path cannot be used for such VARCHAR variables.

DB2 V8 truly supports varying-length index keys in the sense that the index keys are no longer padded and the data is stored as is – making index-only access on such columns possible. You also have the flexibility of controlling whether padding is done or not.

INCREASED LIMITS FOR PARTITIONED TABLE SPACE

In DB2 V8, the maximum number of partitions in a partitioned table space is increased from 254 to 4096. Hence the maximum size of a partitioned table space is increased from 16TB to 128TB.

The large number of partitions helps in increasing the granularity of the data and decreasing the size of individual datasets.

If you wanted to use one partition per day, the earlier restriction of 254 partitions would allow only up to 8 months' worth of data. As a workaround, you would have increased the granularity per partition to a week or a month – impacting the extent of parallelism achieved, say, in a weekend job. With the current limit of 4096, you can use one partition per day and still store up to 11 years' worth of data.

MORE FLEXIBILITY ON PARTITIONED TABLE SPACES

With DB2 V8 you can start with a minimum number of partitions and add partitions at the end of the partitioned table space and bring it into effect immediately.

Similar to GDGs, partitions can be reused for newer data as old data becomes obsolete. You can reuse existing partitions for new data as old data is rolled away.

The REBALANCE utility can be used to rebalance partitions without impacting the availability.

You are no longer forced to use the partitioning key as the clustering order. Now partitions can be created without an index. You can define clustering order differently from partitioning order. For example, if the tablespace is partitioned on branch id, within the partition you can cluster based on account number. You can even remove your partitioning indexes if your application doesn't need them.

MULTIPLE DISTINCTS ALLOWED IN A SINGLE QUERY

In DB2 V7 multiple DISTINCTs on a single query were allowed only if they were on the same column. For example, the following statement is valid:

```
SELECT COUNT(DISTINCT(N1)), AVG(DISTINCT(N1)) FROM T1
```

Whereas if you specify multiple DISTINCTS on different columns, as shown below, you will get SQLCODE -127:

```
SELECT COUNT(DISTINCT(N1)), AVG(DISTINCT(N2)) FROM T1
```

To retrieve multiple distinct column values, you have to resort to writing multiple queries. With DB2 V8, this constraint is removed and now you can specify multiple DISTINCTs on different columns in a single query.

For example to get the distinct number of cardholders and merchants present in a transaction table, you can now use a single query as follows:

```
SELECT COUNT(DISTINCT(CARD_NUMBER)), COUNT(DISTINCT(MERCHANT_ID))
FROM CARD_TRANSACTIONS
```

Do exercise caution before using queries with multiple distinct columns because it could be expensive, depending on the number of sorts performed and work files created.

SUPPORT FOR ALTERING IDENTITY COLUMNS

With DB2 V8 you can dynamically alter the attributes of an existing identity column including the GENERATED clause, without dropping and re-creating a table.

For new applications, you should consider using the new Sequence feature provided by DB2 V8. Sequences are stand-alone objects that provide recoverable, unique, sequential numbers for applications and avoid the concurrency and performance problems faced by traditional methods of sequence number generation.

SUPPORT FOR MULTILEVEL SECURITY WITH ROW-LEVEL GRANULARITY

Restricting access to specific rows in a table has been a requirement of customers for quite some time – for example restricting customers' online Web access to their accounts only. Until now, application developers have handled this requirement by creating views and/or including additional predicates in the application program.

DB2 V8 now provides security to a granular level of a row, implying that the user's access can be restricted to a specific set of row without having to use additional views or predicates.

DB2 V8 supports multilevel security, allowing you to use a system of hierarchical security levels combined with non-hierarchical security categories. This approach brings DB2 security closer to the directory services like LDAP. Now you can map the organization's hierarchy directly to DB2's security hierarchical scheme – for example each manager's access will be limited to employees reporting to them.

DB2 V8 brings a whole new concept of security to your RDBMS. To understand this, *DB2 UDB for z/OS Version 8, Major Security Improvements* by Roger Miller of IBM is a good place to start.

NEWER LIMITS

The following are some of the new limits as applicable in DB2 V8:

- The size of a column name is increased from 20 to 30 bytes.
- Maximum length of a VARCHAR column is increased from 255 to 32,704 bytes.
- Maximum length of an index key is increased from 255 bytes to 2000 bytes.
- A single FROM clause can be used to join up to 225 tables as against the earlier limit of 15 tables.
- You don't need hiperspaces or dataspaces to overcome the virtual storage limit for a single address space. Exploiting the zSeries architecture, DB2 supports 64-bit virtual storage. This raises the real memory limit from 2GB to 16 exabytes – 8 billion times larger.

OTHER RESTRICTIONS ELIMINATED

The following are the newly-eliminated restrictions offered by DB2 V8:

- Columns can be appended to the end of an existing index.
- Column attributes can be modified using the ALTER COLUMN clause.
- INSERT and UPDATE statements can have qualified column names.
- Clustering indexes can be altered.

CONCLUSION

IBM has continued DB2's renaissance, and Version 8 has dramatic and fundamental changes that break through earlier limitations, making significant progress. When upgrading to V8, you can exploit these factors for improving your existing applications and gain in terms of improved performance, decreased storage, role-based security, and continuous availability.

C Sasirekha
System Software Group
Tata Consultancy Services (India)

A quick look at the DB2 Universal Database for Unix, Windows, and Linux Version 8.1.2 update

In May 2003, IBM released the much-anticipated update to the IBM DB2 Universal Database (DB2 UDB) V8.1 release with the V8.1.2 update. This article briefly introduces you to the major new features that are bound to appeal to a broad spectrum of users, from DBAs to developers, and those users in between.

This article divides the new features in the V8.1.2 update into four categories – enhancements for developers, enhancements for DBAs, miscellaneous enhancements, and new packages. This article isn't intended to represent a comprehensive list of the new features, but rather to identify those features that I find most customers take an interest in.

FIX PACKS AND UPDATES – WHAT HAPPENED

In order to help identify feature-oriented ship vehicles from fix-oriented vehicles, vehicles that deliver mainly maintenance and fixes to a DB2 UDB installation are called Fix Packs (FPs). Those that are feature-based are called updates. The path that most maintenance deliveries will follow throughout the release of DB2 UDB will be fix packs, followed by updates. The proposed schedule could be as follows: FP1, then V8.1.2 (which includes FP1), then FP2, then V8.1.4 (which includes FP1, V8.1.2, FP3), and so on.

Many resources, people, and parts of the DB2 UDB information library may still refer to the V8.1.2 update as FP2, so this new naming convention is something that you should be aware of.

ENHANCEMENTS FOR DEVELOPERS

The most significant enhancements delivered to the DB2 UDB platform in the V8.1.2 update are for application developers, specifically those that develop to the Microsoft .Net API.

When DB2 UDB V8.1 became generally available, it came with integration points into Microsoft Visual Studio (its supported languages: Visual Basic, InterDev, and Visual C++) with Java-based add-ins to these respective Integrated Development Environments (IDEs).

The V8.1.2 update embraces the next generation of Microsoft developers, namely .Net developers, with tight native integration into the Visual Studio.Net IDE and a native DB2 UDB managed provider for .Net.

The native integration into Visual Studio.Net includes a specialized

IBM Explorer window, which provides a focused DB2 UDB view into application development. The IBM Explorer is designed to use the DB2 UDB .Net managed provider, is optimized for speed, supports all the DB2 UDB data types and platforms, supports filtering, and more.

The integration allows for ‘drag-and-drop’ ADO.Net client-side code generation for connection objects, data objects, etc.

The integration into Visual Studio.Net is so tight that all windows are displayed in the native Microsoft format, Intellisense help is fully supported, and the help is delivered in the native Microsoft Windows Help format.

The V8.1.2 update also includes a native DB2 UDB .Net managed provider that has been written in fully-managed C# code. Like its Microsoft counterpart for SQL Server, the DB2 UDB .Net managed provider (called IBM.Data.DB2) includes DB2Connection, DB2Command, DB2DataAdapater, and DB2DataReader objects.

Many more integration features were delivered in the V8.1.2 update for .Net developers. You can learn more at www.ibm.com/developerworks/offers/lp/db2/?S_TACT=103AMW30&S_CMP=dmwinDB2Intranet.

The MERGE statement (which is an INSERT combined with an UPDATE statement) is now supported in the V8.1.2 update, as well as other developer enhancements, including the following:

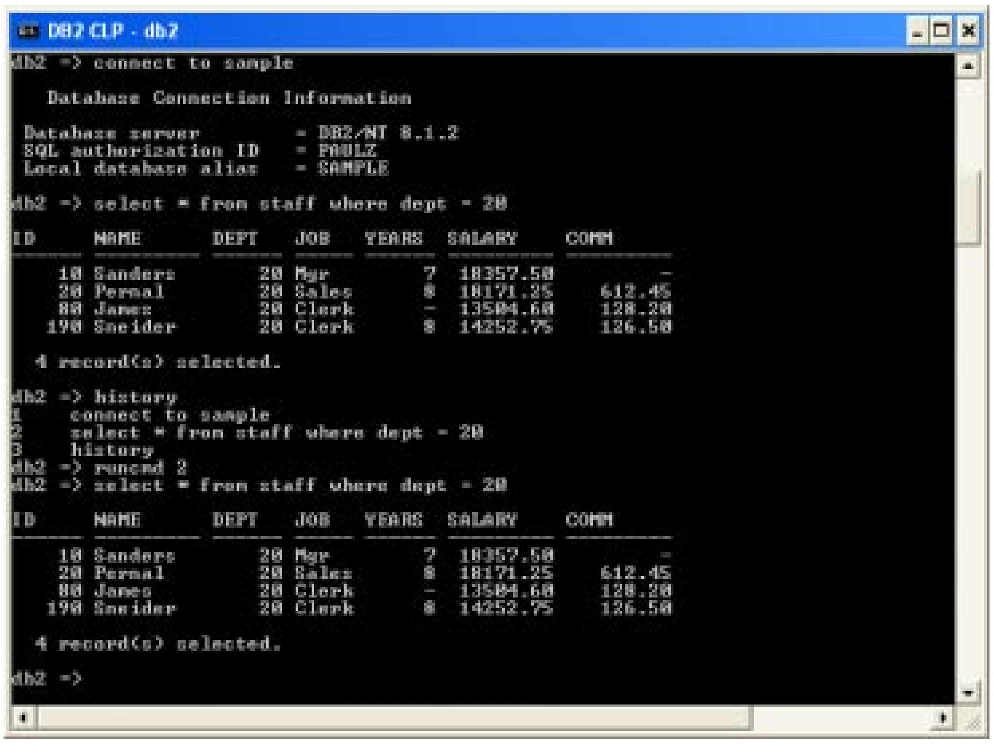
- Various enhancements to the DB2 UDB Development Center.
- Full 64-bit support.
- Federated support for SQL Assist.
- Various usability enhancements.
- More flexibility to the SET CURRENT ISOLATION statement.
- XA transaction support (COM+) enhancements for tightly and loosely coupled transactions.
- Support for the ANSI standard MERGE SQL statement (UPDATE if not INSERT logic).

- Query sampling.
- Support for sequences and identity columns in a partitioned database environment.

ENHANCEMENTS FOR DBAS

The most significant enhancement in the V8.1.2 update for DBAs is the ability to throttle the DB2 UDB BACKUP and REBALANCE utilities.

Maintenance of DB2 UDB databases involves routinely executing utilities such as BACKUP and REBALANCE. However, since these utilities can be fairly resource-intensive (they are designed to run fast, but will leverage all the available resources of a system to do so), DBAs typically schedule these jobs to run during off-peak hours (in 'batch windows') to avoid impacting users of the production system. The batch process approach is becoming a more and more unacceptable management



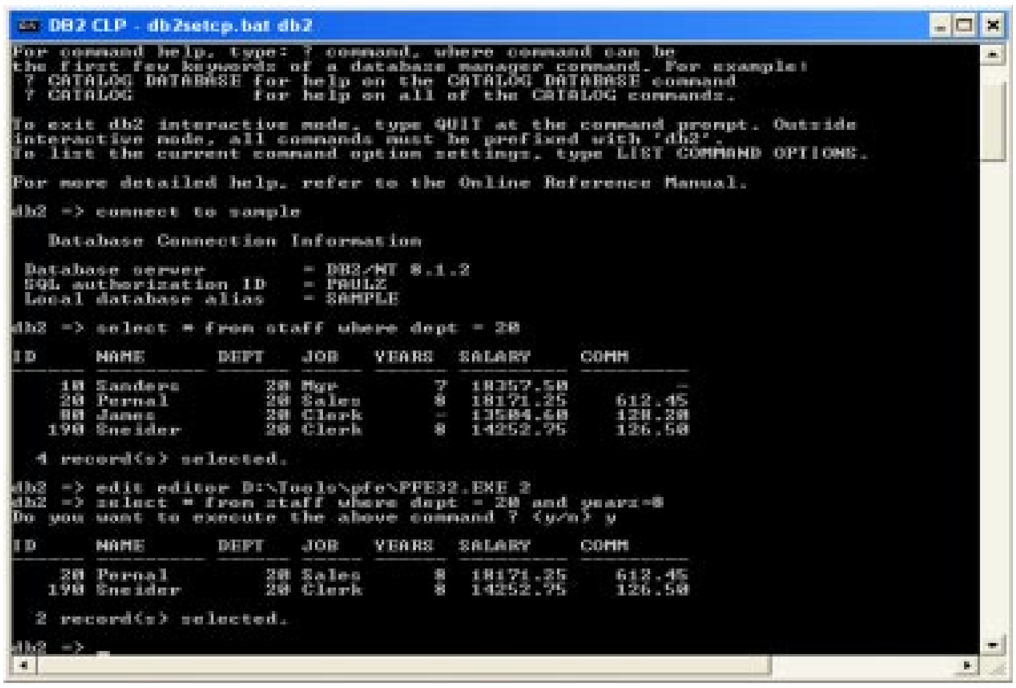
```
db2 => connect to sample

   Database Connection Information

 Database server        = DB2/NT 8.1.2
 SQL authorization ID   = PAULZ
 Local database alias   = SAMPLE

db2 => select * from staff where dept = 20

ID     NAME      DEPT   JOB   YEARS  SALARY     COMM
------ --------- ------ ----- ------ ---------- ----------
    10 Sanders       20 Mgr        7   18357.50          -
    20 Pernal        20 Sales      8   18171.25     612.45
    80 James         20 Clerk      -   13504.60     128.20
   190 Sneider       20 Clerk      8   14252.75     126.50

  4 record(s) selected.

db2 => history
1     connect to sample
2     select * from staff where dept = 20
3     history
db2 => runcmd 2
db2 => select * from staff where dept = 20

ID     NAME      DEPT   JOB   YEARS  SALARY     COMM
------ --------- ------ ----- ------ ---------- ----------
    10 Sanders       20 Mgr        7   18357.50          -
    20 Pernal        20 Sales      8   18171.25     612.45
    80 James         20 Clerk      -   13504.60     128.20
   190 Sneider       20 Clerk      8   14252.75     126.50

  4 record(s) selected.

db2 =>
```

Figure 1: Tracking command history

methodology as businesses push to 24x7 operations across a competitive global landscape.

The new throttling feature gives DBAs the power to regulate the performance impact of online utilities such that they can be assigned more computing resource during off-peak hours, and scaled back during periods of high-resource demand. The feature can also be an invaluable asset for DBAs trying to ensure that maintenance operations do not violate SLA commitments.

The maximum impact percentage for all utilities (that have been enabled for throttling) running within an instance can be controlled through the new UTIL_IMPACT_LIM database manager configuration parameter. This parameter is set as a percentage of the total resources of the system when it is idle. Its value is dynamic in nature and can be set while utilities are running. Changes to thresholds are dynamic so there is no need to stop and restart the instance for the new levels to be honoured. The default for this parameter is 100%. This setting would represent an environment where the supported utilities are not throttled. This parameter can be updated via an API as well.



```
For command help, type: ? command, where command can be
the first few keywords of a database manager command. For example:
 ? CATALOG DATABASE for help on the CATALOG DATABASE command
 ? CATALOG          for help on all of the CATALOG commands.

To exit db2 interactive mode, type QUIT at the command prompt. Outside
interactive mode, all commands must be prefixed with 'db2'.
To list the current command option settings, type LIST COMMAND OPTIONS.

For more detailed help, refer to the Online Reference Manual.

db2 => connect to sample

   Database Connection Information

 Database server        = DB2/NT 8.1.2
 SQL authorization ID   = PAULZ
 Local database alias   = SAMPLE

db2 => select * from staff where dept = 20

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ----------
    10 Sanders       20 Mgr       7  18357.50         -
    20 Pernal        20 Sales     8  18171.25    612.45
    80 James         20 Clerk     -  13504.60    128.20
   190 Sneider       20 Clerk     8  14252.75    126.50

  4 record(s) selected.

db2 => edit editor D:\Tools\pfe\PFE32.EXE 2
db2 => select * from staff where dept = 20 and years=8
Do you want to execute the above command ? (y/n) y

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM
------ --------- ------ ----- ------ --------- ----------
    20 Pernal        20 Sales     8  18171.25    612.45
   190 Sneider       20 Clerk     8  14252.75    126.50

  2 record(s) selected.

db2 => _
```

Figure 2: Re-using commands or SQL statements

For example, the following command syntax could be used with a throttleable utility:

```
SET UTIL_IMPACT_PRIORITY for <utility_id> to <priority 1-1ØØ>
```

Currently, IBM is looking at extending the throttle feature in future releases to other commonly-used utilities – DBAs, stay tuned!

Another manageability feature added to the DB2 UDB V8.1.2 update is the ability to maintain a history of DB2 commands, as well as perform roundtrip query editing in the DB2 UDB Command Line Processor (DB2 UDB CLP).

The historical tracking feature works only in the interactive DB2 UDB CLP. Figure 1 shows the DB2 UDB CLP tracking the history of commands that were run in its session.

In addition to historical tracking of DB2 UDB commands, the DB2 UDB CLP also supports the ability to open a previously-run DB2 UDB command or SQL statement, edit it, and rerun the command or statement without having to retype its entire contents. This feature is illustrated in Figure 2. (What isn't shown in the figure is that a text editor of your choice is launched to edit the query or statement.)

Other manageability enhancements that are part of the DB2 UDB V8.1.2 update include:

- Support for Tivoli GRANT and REVOKE statements, which make it easier for DBAs to restore back-ups on TSM between different nodes.
- LDAP support on xSeries and zSeries Linux distributions.
- Dynamic LPAR support on AIX and Dynamic System Domain support on Solaris for dynamic allocations of computing resources.
- Support for the ALTER TABLE command for VARGRAPHIC columns.
- Various Health Center enhancements with more detailed health indicators and messages.

- Support for the BCP file format for the LOAD utility.
- Various enhancements to DB2 Connect for use with queries that are run on zSeries machines that host DB2 data:
  - support for 2-MB SQL statements.
  - support for tagging zSeries-bound SQL statements for identification.
  - support for SELECT from INSERT statements.

MISCELLANEOUS ENHANCEMENTS

Various enhancements aimed at improving the performance of DB2 UDB were added in the DB2 UDB V8.1.2 update.

DB2 UDB V8.1.2 extends the DB2 UDB V8.1 prefetching enhancements with a redesign to certain aspects of the prefetchers to reduce contention and context switching. This enhancement should help with query reporting-like workloads.

A new feature called Automatic Relationship and Association Management (ARAM) detection has also been added in the recent update. This feature is implemented by an internal utility that is automatically invoked through the RUNSTATS utility. It automatically discovers relationships between columns (the performance impact to RUNSTATs is unnoticeable). These relationships could be Referential Integrity (RI) relationships or correlation information. This information is used and maintained to deliver faster performance through the generation of exception-Materialized Query Tables (MQTs). This is all transparent to users and applications.

DB2 UDB V8.1.2 also includes improved performance for hash joins by selecting bit filters more often (in situations where they will help). This ‘behind-the-scenes’ feature will also improve performance of larger hash joins because of the addition of a 32-bit hash code.

Additionally, a number of Web enablement features are included in the DB2 UDB V8.1.2, such as:

- An embedded copy of the IBM WebSphere Application Server as part of DB2 installation (its use is subject to certain terms and conditions).
- A new JDBC Type 2 Driver (J2EE 1.4).
- New SOAP UDFs.
- MQ Listener.
- Various Web services enhancements.
- Transactional support for MQ UDFs.

NEW PACKAGES

A number of new packages and editions were released around the V8.1 update timeframe.

DB2 UDB Express

DB2 UDB Express is a low-cost entry point into a full-fledged DB2 UDB server. It is every bit the same database engine that is used to power other editions of DB2 UDB, except it is limited to deployments on two-way Linux (AMD or Intel-based) and Windows workstations, and priced by the DB2 UDB per named user model.

DB2 UDB Express retails for US$499 (prices are subject to change without notice and may differ in your area) for the server licence, with an additional $99 per named user. It is also eligible for high availability pricing.

You can learn more about DB2 UDB Express at www-3.ibm.com/software/data/info/db2express/.

DB2 UDB Information Integrator

In DB2 UDB v7.2, read and write access to heterogeneous data stores that managed traditional and non-traditional data were supported from a single product or combination of products. These included (but were not limited to) DB2 UDB DataJoiner, DB2 UDB Relational Connect, DB2 DataPropagator (DPropR), Life Sciences Data Connect, and more.

In DB2 UDB V8.1, these products were removed as orderable offerings. This left a temporary gap in the features of DB2 UDB V8.1 because these products helped businesses leverage disparate data stores in a cost-effective manner, via a single standard SQL API, with included optimization and function compensation. (However, DB2 UDB V8.1 could federate to any member of the DB2 family and Informix databases; this is part of the core engine capabilities.)

A new offering, DB2 Information Integrator (DB2 II), was made available within the DB2 UDB V8.1.2 timeframe to provide customers with read and write access to structured and unstructured data, as shown below.

Specific integration functions delivered by DB2 II include the federation of cacheing for improved federated query performance (through MQTs), full support for the DB2 SQL API, content and life sciences federation, XML and materialized query integration, IBM Lotus Extended Search, ODBC, OLE DB, DB2 UDB NSE, heterogeneous relational replication, and more.

DB2 UDB II is available in four editions:

- *DB2 Information Integrator Replication Edition*. This package includes a limited-use data store (for use as a replication control server) as well as connectors that are priced on a per-connection basis.
- *DB2 Information Integrator Standard Edition*. This package includes a limited-use data store (for use as a local cache), the DB2 UDB NSE, and connectors priced on a per-connection basis. DB2 UDB V7.2 customers who purchased DB2 DataJoiner, DB2 Relational Connect, and DB2 Life Sciences Data Connect are entitled to an upgrade to DB2 II Standard Edition.
- *DB2 Information Integrator Advanced Edition*. This package includes a full-function data store, the DB2 UDB NSE, and connectors priced on a per-connection basis.

- *DB2 Information Integrator Advanced Edition Unlimited*. This package includes a full-function data store, the DB2 UDB NSE, and an unlimited number of connectors to heterogeneous data stores.

DB2 II charge metrics include per-processor charges and per-connector charges. The upgrade path for customers who purchased DataJoiner, Relational Connect, or Life Science Data Connect is DB2 II Standard Edition.

Another offering, called DB2 II for Content, is also available. This product is the follow-up product for Enterprise Information Portal (EIP) and it retains the same packaging as its predecessor.

You can learn more about DB2 II at http://www-3.ibm.com/software/data/integration/.

DB2 UDB Data Warehouse Edition

IBM is a leader in Business Intelligence (BI), with technology built into the DB2 product line, enabling real-time information integration, insight, and decision-making. Two new editions of DB2 UDB combine the strength of DB2 with essential IBM business intelligence infrastructure elements. DB2 UDB Data Warehouse Enterprise Edition (DB2 UDB DWEE) provides a comprehensive BI platform with everything needed by customers to deploy, and by partners to build, a powerful platform for next-generation business intelligence solutions.

DB2 UDB DWEE is a powerful business intelligence platform that includes DB2 UDB, federated data access, data partitioning, integrated OLAP, advanced data mining, enhanced ETL, and workload management, and supports BI for the masses. In particular, this package includes:

- DB2 UDB Enterprise Server Edition.
- DB2 Data Partitioning Feature.
- DB2 Multidimensional Metadata Management.
- DB2 Intelligent Miner Modeling.

- DB2 Intelligent Miner Visualization V8.1.
- DB2 Intelligent Miner Scoring V8.1.
- IBM Office Connect Enterprise Web Edition.
- DB2 Query Patroller (the new Version 8 release).
- DB2 Warehouse Manager Standard Edition.
- DB2 Information Integrator Standard Edition.

If you were to add up the retail price for all of these products, it would cost about US$139,600 per processor. The price of DB2 UDB DWEE is US$50,000 per processor.

This package is also available as a Standard Edition (SE) offering, whose composition of products would retail for about US$92,500 per processor. DB2 UDB DWSE retails for US$15,000 per processor. You can learn more about this new edition of DB2 UDB at www.ibm.com/software/data/bi/.

Other noticeable packaging-related enhancements that occurred around the V8.1.2 update include:

- Support for DB2 UDB for Linux on iSeries and pSeries.
- Certification of DB2 UDB V8 on Windows 2003 workstations and support for DB2 UDB v7.4 with FP6 or later server.
- Support for 64-bit Linux on AMD and Intel.
- Support for DB2 UDB for Windows in a 64-bit environment.

CONCLUSION

As you can see, the DB2 UDB V8.1.2 is full of new features and functions that appeal to the entire spectrum of DB2 UDB users. You can download a copy of the update from www.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/index.d2w/report, as well as the *Release Notes*, which explain in detail the features that I have covered in this article.

Paul Zikopoulos, BA, MBA (paulz_ibm@msn.com)
Database Specialist
IBM (Canada)

Spiffing up SPUFI

Using SPUFI, you normally first go to a panel where you must specify a DB2 subsystem ID; next you must enter some details on a SPUFI panel called DSNESP01, (optionally) edit your specified input dataset, and use PF3 to save the data. Then it runs all the SQL in that input dataset and you browse the SPUFI output dataset to see the result(s). When you exit from the browse via PF3 you are returned to the SPUFI panel again, then PF3 finally lets you exit from SPUFI. I find that a bit cumbersome.

Here is a way to invoke SPUFI directly from an EDIT (or VIEW) of some SQL statements. It invokes SPUFI for the whole data or for only the lines you select. You will see only the SPUFI output dataset, then PF3 returns you directly to your EDIT (without you seeing the normal SPUFI panel at all).

This code also permits you to execute SQL directly from a CLIST or EXEC by invoking SPUFI, then you could browse the output or run an output parsing routine to get what you want.

Since this requires some changes to the standard SPUFI panel, I have also enhanced that for normal use – now it will automatically add a matching bracket or quote at the end of any dataset name you specify (if required).

HOW IT WORKS

The ESQL edit macro copies the selected SQL to an input dataset for SPUFI, then invokes EXSPUFI EXEC to allocate the necessary DB2 libraries and invoke SPUFI via the standard IBM CLIST DSNESC01.

The SPUFI panel DSNESP01 is modified to carry out the actions without being seen. It substitutes your desired values and executes the SQL immediately. Then after you browse the output, the DSNESP01 panel restores the previous values and exits.

USAGE

Invoke it from any EDIT (or VIEW) by using the command **ESQL** or **ESQL ssid**. To select particular lines of SQL to run, use the **CC** block line commands, otherwise it runs all that you are editing. You don't need to save what you run; you can easily CANCEL the edit or UNDO any changes that you made.

If you want to run SPUFI from your own EXEC or CLIST, you can invoke EXSPUFI passing all necessary parameters as explained in the comment box at the start of EXSPUFI.

ISPF COMMAND

Here is how you can invoke the standard SPUFI by entering **SPUFI** (or **SPUFI ssid**) on the command line of any ISPF panel. Then the (modified) DSNESP01 SPUFI panel will be shown, ready for input.

- If your ESQL EXEC has *spufi_screen* = *'NEW'*, a new ISPF logical screen will be opened for SPUFI.
- If your ESQL EXEC has *spufi_screen* = *'ONTOP'*, the SPUFI screen will be over the top of where you entered the **SPUFI** command.

You add the following ISPF command to one of your active command tables (preferably a user or site table):

```
Verb   = SPUFI
Trunc  = Ø
Action = SELECT CMD(%ESQL &ZPARM) NEWAPPL
Description = 'Invoke SPUFI'
```

ESQL MACRO

Check the default values at the start of the EXEC and update them as desired:

- Set a default DB2 subsystem ID.
- SPUFI can be run either in a new logical split screen (NEW) or over the top of the EDIT (ONTOP).

- An output dataset name can be generated and definitely used (YES), or used on the condition that no previous name was defined (COND), or the previously used name should be used (NO).
- It also sets *ISOLATION = 'CS'* and *AUTOCOMMIT = 'YES'*.
- Check *libry_alloc = 'DB2LIBS'*; change it to the name of your own library allocation routine if you don't want to use DB2LIBS.

Check section *CREATE_SPUFIDSN* where it creates the name for the SPUFI input dataset. In this code it will be called userid.SPUFIN or tsoprefix.userid.SPUFIN, conforming to IBM standards. However, if the user has set TSO PROFILE NOPREFIX, it gets a value from RACF and that may not be valid for your company. (It may also generate a name for the SPUFI output dataset, and that will be based on the name for the SPUFI input dataset.)

```
/*============================> REXX <===============================*/
/* ESQL: Invoke SPUFI                                                */
/*-------------------------------------------------------------------*/
/*       a) If invoked from EDIT:                                    */
/*           ie enter 'ESQL' or 'ESQL ssid' on command line, and     */
/*               select the lines of SQL via 'CC' line commands      */
/*          Then the SQL will be invoked immediately and the SUPFI   */
/*          output file shown, returning directly to EDIT via PF3.   */
/*          Thus, the user does not see the normal SPUFI panel.      */
/*                                                                   */
/*       b) If invoked via command:                                  */
/*           ie enter 'SPUFI' or 'SPUFI ssid' on command line, where */
/*               that is defined as: SELECT CMD(%ESQL &ZPARM) NEWAPPL */
/*               in an active ISPF command table.                    */
/*          Then the normal SPUFI panel (DSNESPØ1) will be shown.    */
/*                                                                   */
/*------------------------------------------------------------------*/
/* Externals:                                                        */
/*       EXSPUFI   Rexx   .. call DB2LIBS, then call DSNESCØ1        */
/*       DB2LIBS   Rexx   .. allocate DB2 libraries                  */
/*       DSNESCØ1  Clist  .. as supplied by IBM (to invoke SPUFI)    */
/*       DSNESPØ1  Panel  .. modified version of IBM-supplied panel  */
/*-------------------------------------------------------------------*/
/* Written: 2ØØ3/Ø6/Ø5     Last Updated: 2ØØ3/Ø6/18     by Ron Brown  */
/*===================================================================*/
   default_ssid = 'ssid'           /* ssid used when none specified */
   spufi_loc    = ''               /* remote location               */
   spufi_screen = 'ONTOP'           /* 'NEW' or 'ONTOP'                */
   spufi_out    = 'COND'            /* generate output dataset name? */
                                    /* .... 'NO', 'YES' or 'COND'    */
   spufi_br     = 'YES'             /* browse output? 'YES' or 'NO'  */
   spufi_iso    = 'CS'              /* SPUFI isolation? 'CS' or 'RR' */
   spufi_com    = 'YES'             /* autocommit? 'YES' or 'NO'     */
   libry_alloc  = 'DB2LIBS'         /* routine to alloc SPUFI libs   */
   Address ISPEXEC "CONTROL ERRORS RETURN"  /* handle any errors here */
   Address ISPEXEC "CONTROL DISPLAY SAVE"
   dset = ''       /* name of SPUFI input dsn (if SQL copied into it) */
   spufiout = ''   /* name of SPUFI output dsn to be used            */
   Address ISREDIT "MACRO (ssid) NOPROCESS"
   macro_rc = rc
   If macro_rc > Ø Then            /* if NOT running from Edit/View  */
      Arg ssid                     /* get specified DB2 ssid         */
   If ssid = '' Then ssid = default_ssid    /* user specified no ssid */
   If macro_rc = Ø Then Do         /* running from Edit/View         */
      Call GET_SQL                 /* get specified SQL statements   */
      Call CREATE_SPUFIDSN         /* write SQL into SPUFI input file*/
      End
   /*-- create parameter list for EXSPUFI ---*/
   ex_parms = "ALLOC("libry_alloc")",     /* alloc DB2 libs   */
              "SSID("ssid")",             /* DB2 sub-system   */
              "LOC("spufi_loc")",         /* connect location */
              "INDSN("dset")",            /* input dataset    */
              "OUTDSN("spufiout")",       /* output dataset   */
              "BR("spufi_br")",           /* browse output    */
              "ISO("spufi_iso")",         /* SPUFI isolation  */
              "COM("spufi_com"))"         /* auto-commit      */
   /*-- use program ISPSTRT to open a new logical (split) screen ----*/
   /*-- this only works if EXSPUFI in SYSEXEC or SYSPROC library ----*/
   If spufi_screen = 'NEW' Then
      Address ISPEXEC "SELECT PGM(ISPSTRT) PARM(CMD(%EXSPUFI" ex_parms,
                      "NEWAPPL("ssid") SCRNAME(SPUF"ssid"))"
   /*-- this invokes it in the same logical screen (over the top) --*/
   If spufi_screen = 'ONTOP' Then
      Address ISPEXEC "SELECT CMD(%EXSPUFI" ex_parms,
                      "NEWAPPL("ssid") PASSLIB"
   Call DELETE_SPUFIDSN            /* delete the SPUFI input file    */
   Address ISPEXEC "CONTROL DISPLAY RESTORE"
   Return
/*=================================================================*/
/* put the specified SQL into REXX array: input                    */
/*-----------------------------------------------------------------*/
GET_SQL:
   /*-- get CC range into variables fl & ll --*/
   Address ISREDIT "PROCESS RANGE C "
   proc_rc = rc
   Select
     When proc_rc = Ø then Do
```

```
         Address ISREDIT "(fl) = LINENUM .ZFRANGE"
         Address ISREDIT "(ll) = LINENUM .ZLRANGE"
         End
      When proc_rc = 4 then Do /* no range specified, take whole d'set */
         Address ISREDIT "(fl) = LINENUM .ZF"
         Address ISREDIT "(ll) = LINENUM .ZL"
         End
      Otherwise   /* eg rc=16 for incomplete or conflicting line cmds */
         Address ISPEXEC "SETMSG MSG(ISRZØØ2)"
         Exit proc_rc
      End
   /*-- write the selected SQL into input. array --*/
   j = Ø
   Do i = fl to ll
      j = j + 1
      Address ISREDIT "(line) = LINE "i
      input.j = line
      End
   input.Ø = j
   Return
/*=================================================================*/
/* write specified SQL into input dataset for SPUFI                */
/*-----------------------------------------------------------------*/
CREATE_SPUFIDSN:
   /*-- check DCB of SQL input dataset --*/
   Address ISREDIT "(fm1,fm2) = RECFM"
   Address ISREDIT "(lrec)    = LRECL"
   If fm1 <> 'F' | lrec <> 8Ø Then Do
      ZEDLMSG = 'This is RECFM='fm1||Strip(fm2)',',
               'LRECL='Strip(lrec,'L','Ø')', but SPUFI',
               'accepts only RECFM=F or FB, LRECL=8Ø'
      Address ISPEXEC "SETMSG MSG(ISRZØØ1)"
      Exit 8
      End
   /*-- set name of SPUFI input dataset which will be created --*/
   spufidsn = Userid()||'.SPUFIN'
   tsopref = Sysvar(SYSPREF)
   /* default TSO PREFIX here is the same as the user's RACF ACCOUNT */
   If tsopref = '' Then Do    /* if user has set TSO PROFILE NOPREFIX */
      w = Outtrap('racfmsg.')
      Address TSO "LU "||Userid()||" NORACF TSO" /* get RACF TSO info */
      Parse Var racfmsg.5 . tsopref .           /* 'ACCTNUM= tsopref' */
      x = Outtrap('OFF')
      End
   If tsopref <> Userid() & tsopref <> '' Then
      spufidsn = tsopref||'.'||spufidsn
   /*-- allocate the SPUFI input dataset --*/
   oldstat = MSG('OFF')                /* ensure TSO messages are OFF */
   Address TSO "ALLOC F(SPUFIN) DSN('"spufidsn"') NEW CATALOG" ,
      "SPACE(1,1) CYL RECFM(F B) LRECL(8Ø) BLKSIZE(Ø)"
   alloc_rc = rc
   a = MSG(oldstat)                    /* reinstate TSO messages      */
   /*-- copy SQL into the SPUFI input dataset --*/
   Address TSO "EXECIO * DISKW SPUFIN (STEM input. FINIS"
   If rc = Ø Then Do
      dset = "'"spufidsn"'"    /* dset non-null only when copy was OK */
      /*-- set name of SPUFI output dataset --*/
      If spufi_out = 'YES' Then spufiout =,
         "'"Left(spufidsn,Length(spufidsn)-1)||'OUT.'||ssid"'"
      If spufi_out = 'COND' Then spufiout =,
         "'"Left(spufidsn,Length(spufidsn)-1)||'OUT.'||ssid"',COND"
      If spufi_out = 'NO' Then spufiout = ''
      End
   Return
/*=================================================================*/
/* delete SQL input dataset for SPUFI                              */
/*----------------------------------------------------------------*/
DELETE_SPUFIDSN:
   If alloc_rc = Ø Then Do
      oldstat = MSG('OFF')
      Address TSO "FREE FILE(SPUFIN)"
      Address TSO "DELETE '"spufidsn"'"
      a = MSG(oldstat)
      End
   Return
```

EXSPUFI EXEC

This needs no changes.

```
/*============================> REXX <==============================*/
/* EXSPUFI: Execute SPUFI                                           */
/*------------------------------------------------------------------*/
/*       a) Command ('ESQL' or 'ESQL ssid') is entered on the command */
/*          line of an EDIT/VIEW of some SQL.  ESQL edit macro runs   */
/*          and it invokes EXSPUFI (supplying all parameters and with */
/*          "NEWAPPL(ssid)" ).                                        */
/*                                                                    */
/*       b) If invoked from another EXEC or CLIST:                    */
/*          That allows you to run SQL without using the REXX-DB2 or  */
/*          DSNTEP2 programs, then you could browse the output or run */
/*          some output parsing routine to extract what you want.     */
/*          Here is some example pseudo code:                         */
/*             1 write the SQL into 'input.dsn' (RECFM=FB,LRECL=8Ø)   */
/*             2 allocate file 'output.dsn'     (RECFM=FB or VB)      */
/*             3 SELECT CMD(%EXSPUFI SSID(ssid) INDSN('input.dsn')    */
/*                        OUTDSN('output.dsn') BR(NO)) NEWAPPL(ssid)  */
/*             4 parse file 'output.dsn' to check the SQLCODE         */
/*          It runs the SQL in 'input.dsn' and outputs to 'output.dsn'*/
/*          (but does not show the output).  Any parameters which are */
/*          not specified, will be defaulted in the DSNESPØ1 panel.   */
/*-------------------------------------------------------------------*/
/* Written: 2ØØ3/Ø6/Ø5     Last Updated: 2ØØ3/Ø6/16     by Ron Brown  */
/*===================================================================*/
   Address ISPEXEC                    /* commands go to ISPF services */
   /*----------------------*/
   /* get start parameters */
   /*----------------------*/
   Arg parms
   Parse Upper Value ' 'parms With ,
      1 ' ALLOC(' lib_alloc ')',          /* DB2 library alloc  */
      1 ' SSID(' dsneovØ1   ')',          /* DB2 sub-system     */
      1 ' LOC('  spufiloc   ')',          /* connect location   */
      1 ' INDSN('  spufidsn ')',          /* input dataset      */
      1 ' OUTDSN(' spufiout ')',          /* output dataset     */
      1 ' BR('   spufibro   ')',          /* browse output      */
      1 ' ISO('  spufiiso   ')',          /* SPUFI isolation    */
      1 ' COM('  spuficom   ')'           /* auto-commit        */
  "VPUT (DSNEOVØ1,SPUFILOC,SPUFIDSN,SPUFIOUT,SPUFIBRO,",
        "SPUFIISO,SPUFICOM) PROFILE"
  /*-----------------------------------*/
  /* DB2 library allocation for SPUFI */
  /*-----------------------------------*/
  If lib_alloc <> 'NO' Then Do
     If lib_alloc = '' Then
        lib_alloc = 'DB2LIBS'           /* default allocation routine */
     "SELECT CMD(%"lib_alloc dsneovØ1")"
     End
  /*-------------------------------------------*/
  /* Invoke normal SPUFI via standard IBM CLIST */
  /*-------------------------------------------*/
   "VGET DSNESV1W PROFILE"   /* is this the first time for this ssid? */
   If DSNESV1W <> 'SECOND TIME' Then Do /* set missing SPUFI defaults */
     Parse Value '; 25ØØ 4Ø92 4Ø96 VB SYSDA 33 256 NAMES C SECOND TIME',
        With DSNESV2B DSNESV2D DSNESV2C DSNESV21 DSNESV22 DSNESV2E,
             DSNESV24 DSNESV25 DSNESV26 DSNESV3Z DSNESV1W
      "VPUT (DSNESV2B DSNESV2D DSNESV2C DSNESV21 DSNESV22 DSNESV2E",
            "DSNESV24 DSNESV25 DSNESV26 DSNESV3Z DSNESV1W) PROFILE"
      End
  "SELECT CMD(%DSNESCØ1 FUNC(SPUFI))"
  Return
```

DB2LIBS EXEC

Here is a sample EXEC to allocate the required DB2 libraries, which must be changed to meet your local standards. You probably have a similar allocation routine already. Note that for our purposes you need to concatenate a library which contains the modified DSNESP01 panel before the standard IBM library.

An alternative approach could be to allocate your panel library to DDNAME = ISPPUSR, then it is automatically concatenated ahead of the libraries which are specified in the LIBDEF.

```
/*============================> REXX <================================*/
/* DB2LIBS: Allocate DB2 TSO/ISPF libraries for SPUFI                */
/*                                                                    */
/* Parameters Passed:                                                 */
/* dsmbr - Data Sharing Member or Data Sharing Group Name             */
/*-------------------------------------------------------------------*/
/* Created: 2003/06/05     Last Updated: 2003/06/11     by Ron Brown  */
/*====================================================================*/
  Address ISPEXEC                     /* commands go to ISPF services */
  Parse Arg dsmbr .                       /* DB2 ssid or group        */
  If Right(dsmbr,1) = '0' Then Do    /* If group - find local ssid    */
     smfid = MVSVAR('SYSSMFID')           /* MVS name (eg. S101,S102) */
     DSMBR = Left(dsmbr,3)||Right(smfid,1)/* DB2 ssid (eg. SDB1,SDB2) */
     End
  dsgrp = Left(dsmbr,3)||'0'         /* Data Sharing Group (eg. SDB0)  */
  "LIBDEF ISPLLIB DATASET ID('SYS1.DB2"dsgrp".SDSNEXIT'",
                           "'SYS1.DB2"dsgrp".SDSNLOAD'",
                           "'"dsgrp"."dsmbr".RUNLIB.LOAD')"
  "LIBDEF ISPMLIB DATASET ID('SYS1.DB2.SDSNSPFM') STACK"
  Call SET_PANELIBS
  "LIBDEF ISPPLIB DATASET ID("panelibs") STACK"
  "LIBDEF ISPSLIB DATASET ID('SYS1.DB2.SDSNSPFS') STACK"
  "LIBDEF ISPTLIB DATASET ID('SYS1.DB2.SDSNSPFT') STACK"
  "LIBDEF DSNSPFT LIBRARY ID(ISPPROF)"
  Address TSO "ALTLIB ACT APPLICATION(CLIST) UNCOND ",
              "DATASET('SYS1.DB2.NEW.SDSNCLST')"
  Return
  /*==================================================================*/
  /* Make list of panel libraries, including one which contains a     */
  /* modified DSNESP01 panel to run SQL without displaying the panel. */
  /*------------------------------------------------------------------*/
SET_PANELIBS:
  panelibs = "'SYS1.DB2.SDSNSPFP' 'SYS1.DB2.SDSNPFPE'"    /* IBM libs */
  panelibs = "'our.spufi.panelib'" panelibs
  Return
```

DSNESP01 PANEL

This is not the full panel definition. Start with a copy of the IBM panel and apply the following additions/updates:

- Insert this at the start of the *)ATTR* section (before IBM's comments):

```
/*********************************************************************/
/* Modified to enable SPUFI to be run directly from an EXEC or CLIST, */
/* without displaying this panel.                                    */
/*                                                                   */
/* a) When a dataset name is specified in SPUFIDSN (Profile pool), it */
/*    has come from REXX EXEC EXSPUFI.  Current field values will be  */
/*    saved, replaced by values for immediate execution of the dsname */
/*    from SPUFIDSN, and <ENTER> key simulated (to invoke the SQL     */
/*    processing without displaying the panel).  On return to this    */
/*    panel the original field values are written back to the pool,  */
/*    and <END> key simulated (to exit without displaying the panel). */
/*    Therefore, the user does not see this panel at all.            */
/*                                                                   */
/*    Note that the above actions only occur if SPUFIDSN holds a name */
/*    from EXSPUFI; otherwise it functions as usual.  Edit macro ESQL */
/*    invokes EXSPUFI and other execs/clists could potentially do the */
/*    same.                                                          */
/*                                                                   */
/* b) DATA SET NAME  fields add final quote or bracket automatically  */
/*                                                                   */
/*                                              Ron Brown  June 2ØØ3    */
/*********************************************************************/
```

- Put your company name in the title to show it is a modified panel. The second line of the *)BODY* section is as follows (the rest of the section remains unchanged):

```
%                          SPUFI for Spiffy Computer Co.      SSID:
&DSNEOVØ1
```

- Insert this at the end of the *)INIT* section (the rest of the section is unchanged):

```
   /*-----------------------------------------------------------------*/
   /* SAVE & RESTORE VALUES TO RUN SPUFI WITHOUT DISPLAYING THIS PANEL*/
   /*-----------------------------------------------------------------*/
VGET (SPUFIDSN) PROFILE              /* dsn from EXSPUFI exec?   Ron */
IF (&SPUFIDSN ¬= &Z)                 /* if dsname found          Ron */
     &EXSPUFI = Y                    /* invoked by EXSPUFI       Ron */
   /*-- Input Dataset Name --*/                                /* Ron */
     &OLDESV15 = &DSNESV15           /* save Dataset Name        Ron */
     &DSNESV15 = &SPUFIDSN           /*                          Ron */
     &SPUFIDSN = &Z                  /*  remove dsname from pool Ron */
     VPUT (SPUFIDSN) PROFILE         /*                          Ron */
   /*-- Output Dataset Name --*/                               /* Ron */
     &OLDESV16 = &DSNESV16           /* save Output Dataset Name Ron */
```

```
  VGET (SPUFIOUT) PROFILE         /*                          Ron */
  IF (&SPUFIOUT ¬= &Z)            /*  new out-dsname supplied Ron */
      &OUTDSN = TRUNC(&SPUFIOUT,',') /*  out-dsname           Ron */
      &OUTCOND =.TRAIL            /*  'COND' or blank         Ron */
      IF (&DSNESV16 = &Z OR &OUTCOND = &Z)                 /* Ron */
          &DSNESV16 = &OUTDSN     /*                          Ron */
      &SPUFIOUT = &Z              /*  remove dsname from pool Ron */
      VPUT (SPUFIOUT) PROFILE     /*                          Ron */
/*-- Change Defaults? --*/                                 /* Ron */
  &OLDESV1A = &DSNESV1A           /* save Change Defaults     Ron */
  &DSNESV1A = NO                  /*                          Ron */
/*-- Edit Input Dataset? --*/                              /* Ron */
  &OLDESV17 = &DSNESV17           /* save Edit Input          Ron */
  &DSNESV17 = NO                  /*                          Ron */
/*-- Execute SQL? --*/                                     /* Ron */
  &OLDESV18 = &DSNESV18           /* save Execute SQL         Ron */
  &DSNESV18 = YES                 /*                          Ron */
/*-- Autocommit? --*/                                      /* Ron */
  &OLDESV1D = &DSNESV1D           /* save Autocommit          Ron */
  VGET (SPUFICOM) PROFILE         /*                          Ron */
  IF (&SPUFICOM ¬= &Z)            /*   value supplied         Ron */
      &DSNESV1D = &SPUFICOM       /*                          Ron */
      &SPUFICOM = &Z              /*                          Ron */
      VPUT (SPUFICOM) PROFILE     /*   remove value from pool Ron */
  ELSE                            /*                          Ron */
      &DSNESV1D = YES             /*                          Ron */
/*-- Browse Output Dataset? --*/                           /* Ron */
  &OLDESV19 = &DSNESV19           /* save Browse Output       Ron */
  VGET (SPUFIBRO) PROFILE         /*                          Ron */
  IF (&SPUFIBRO ¬= &Z)            /*   value supplied         Ron */
      &DSNESV19 = &SPUFIBRO       /*                          Ron */
      &SPUFIBRO = &Z              /*                          Ron */
      VPUT (SPUFIBRO) PROFILE     /*   remove value from pool Ron */
  ELSE                            /*                          Ron */
      &DSNESV19 = YES             /*                          Ron */
/*-- Connect Location --*/                                 /* Ron */
  &OLDESV1L = &DSNESV1L           /* save Connect Location    Ron */
  VGET (SPUFILOC) PROFILE         /*                          Ron */
  IF (&SPUFILOC ¬= &Z)            /*   value supplied         Ron */
      &DSNESV1L = &SPUFILOC       /*                          Ron */
      &SPUFILOC = &Z              /*                          Ron */
      VPUT (SPUFILOC) PROFILE     /*   remove value from pool Ron */
  ELSE                            /*                          Ron */
      &DSNESV1L = &Z              /*                          Ron */
/*-- Isolation Level --*/                                  /* Ron */
  &OLDESV2F = &DSNESV2F           /* save Isolation Level     Ron */
  VGET (SPUFIISO) PROFILE         /*                          Ron */
  IF (&SPUFIISO ¬= &Z)            /*   value supplied         Ron */
      &DSNESV2F = &SPUFIISO       /*                          Ron */
```

```
          &SPUFIISO = &Z              /*                         Ron */
          VPUT (SPUFIISO) PROFILE     /*   remove value from pool Ron */
      ELSE                            /*                         Ron */
          &DSNESV2F = CS              /*                         Ron */
    /*-- Simulate <ENTER> key --*/                            /* Ron */
      .RESP = ENTER                   /* OK - let's do it        Ron */
                                      /*----------------------------*/
ELSE                                  /* &SPUFIDSN = &Z          Ron */
/*-- &SPUFIDSN = &Z in normal use, or after SQL was run for EXSPUFI --*/
     IF (&EXSPUFI = Y)                /* if invoked by EXSPUFI   Ron */
       /*-- restore all the values --*/                       /* Ron */
          &DSNESV15 = &OLDESV15       /* restore Dataset Name    Ron */
          VPUT (DSNESV15) PROFILE     /*                         Ron */
          &DSNESV16 = &OLDESV16       /* restore Output Dsname   Ron */
          VPUT (DSNESV16) PROFILE     /*                         Ron */
          &DSNESV1A = &OLDESV1A       /* restore Change Defaults Ron */
          VPUT (DSNESV1A) PROFILE     /*                         Ron */
          &DSNESV17 = &OLDESV17       /* restore Edit Input      Ron */
          VPUT (DSNESV17) PROFILE     /*                         Ron */
          &DSNESV18 = &OLDESV18       /* restore Execute SQL     Ron */
          VPUT (DSNESV18) PROFILE     /*                         Ron */
          &DSNESV1D = &OLDESV1D       /* restore Autocommit      Ron */
          VPUT (DSNESV1D) PROFILE     /*                         Ron */
          &DSNESV19 = &OLDESV19       /* restore Browse Output   Ron */
          VPUT (DSNESV19) PROFILE     /*                         Ron */
          &DSNESV1L = &OLDESV1L       /* restore Location        Ron */
          VPUT (DSNESV1L) PROFILE     /*                         Ron */
          &DSNESV2F = &OLDESV2F       /* restore Isolation Level Ron */
          VPUT (DSNESV2F) PROFILE     /*                         Ron */
        /*-- Simulate <END> key --*/                          /* Ron */
          IF (.MSG ¬= &Z) .MSG = &Z   /* Ignore any message      Ron */
          .RESP = END                 /* .. we're finished now   Ron */
                                      /*----------------------------*/
```

- Change the end of the *)PROC* section to the following.

```
     lines have the comment '** Ron */' and the rest are unchanged:
  IF (&DSNESV18 = 'YES')               /* EXECUTION REQUESTED       */
    VER(&DSNESV15,NONBLANK,MSG=DSNE35Ø) /* IS INPUT DS SPECIFIED ?   */
    VER(&DSNESV15,DSNAMEPQ)       /* SYNTAX CHECKING 'OTHER' DS ** Ron*/
    VER(&DSNESV16,NONBLANK,MSG=DSNE359) /* IS OUT DATA SET SPECIFIED? */
    VER(&DSNESV16,DSNAMEQ)        /* SYNTAX CHECKING OUTPUT  DS ** Ron*/
    IF (&DSNESV16 = SYSOUT)            /* IF SYSOUT SPECIFIED       */
     .MSG = DSNE36Ø                    /* WE DONT SUPPORT IT        */
  IF (&DSNESV18 = 'NO ')               /* EXECUTION NOT REQUESTED   */
    IF (&DSNESV17 = 'YES')             /* EDIT OPTION SELECTED      */
     VER(&DSNESV15,NONBLANK,MSG=DSNE35Ø) /* IS INPUT DS SPECIFIED ?   */
     VER(&DSNESV15,DSNAMEPQ)      /* SYNTAX CHECKING 'OTHER' DS ** Ron*/
    IF (&DSNESV19 = 'YES')             /* BROWSE OPTIONS SELECTED   */
     VER(&DSNESV16,NONBLANK,MSG=DSNE359)/*IS OUT DATA SET SPECIFIED ? */
     VER(&DSNESV16,DSNAMEQ)       /*SYNTAX CHECKING OUTPUT DS   ** Ron*/
)END
```

INSTALLATION

- DSNESP01 should be copied to a special library and modified as detailed above. Do not modify your SDSNPFPE SMP/E target library except via an SMP/E usermod!
- ESQL, EXSPUFI, and (optionally) DB2LIBS should be copied into a library in your SYSEXEC (or SYSPROC) concatenation.
- Update ESQL to set the best default DB2 ssid, and any other defaults that you wish to change. Check the code for generating the name for SPUFI input and output datasets and amend it if necessary.
- Update DB2LIBS to your local library naming standards, or alternatively update ESQL to invoke your own allocation routine instead of DB2LIBS.

SUMMARY

This provides a quick and easy way to run SQL from an EDIT, and also provides ways to invoke SPUFI from an ISPF command or from your own EXEC or CLIST. It makes you wonder why IBM didn't do it!

Ron Brown
Principal Consultant
Triton Consulting (Germany)

DB2 UDB load times

When loading data into a table there has always been a requirement to do it as quickly as possible, to minimize the time that the table is unavailable. Bearing this in mind, does it matter whether you create the indexes for the table before or after you load the data, and what about taking an in-stream back-up of the data whilst doing the load – will that reduce the unavailable time? This note looks at four possible methods of creating a table and loading data into it, and seeing which method results in the lowest unavailable time.

Let's use as an example a 600,000 row table. I ran the loads on a Windows 2000 machine running DB2 UDB 8.1 FP1. The table was placed on the C: drive and the back-up on the D: drive (two physically separate disks). I used the SAMPLE database with LOGRETAIN set to YES (so that we can use the COPY option of the LOAD command).

The DDL of the table is unimportant, what we are interested in is the relative timings of the load methods. What we want to end up with is a table with three indexes, ready for insert operations. To get there, consider the four methods shown in the table below, which create the indexes before and after the data is loaded, and loading the data with and without an in-stream image copy. Each method was run, and the component and total times recorded.

The load command we will use is of the form:

```
>db2 load from hm.tbl of del modified by coldel| messages hm.out insert
into hmØ1
```

and if we are doing an in-stream copy, we just add:

```
copy yes to d:\backup
```

The table below shows the results of the tests. The elapsed time for the CREATE TABLE command is minimal and will be the same for each test – therefore it is not shown in the resulting times. Load data+I/C indicates that we are doing a load with an inline image copy. So, for example, test A says that the table was created, the data loaded into it, the three indexes created and an online image copy taken. The individual component times show that the load data step took 1 minutes 39 seconds, the create index step took 0:27 min, and the online image copy step took 2:46 min, giving a total time of 4:52 min.

The results look like:

A   Load data, create indexes, image copy

	1:39+0:27+2:46 = 4:52	
B	Create indexes+load data, image copy	1:56+0:30 = 2:26
C	Load data+I/C, create indexes	1:26+2:3 = 3:59
D	Create indexes+(load data&I/C)	1:54 = 1:54

You can see that if you load the data, create the indexes, and perform an image copy in three separate steps (method A), then the combined total time for all three steps is significantly longer than if you first create the indexes and then perform a load with an inline image copy (method D). Also, creating the indexes before loading the data seems to offer better times, irrespective of whether the image copy is inline or not.

If you are only going to SELECT from the table after the load (not INSERT, UPDATE, or DELETE), then you do not need to take an image copy, so option B without the image copy might be the best one. However, in the above test there is very little difference between options B and D, but the difference may be more significant for larger amounts of data.

You can see that there is a noticeable improvement if you specify that the image copy is taken when doing the load rather than as a separate step afterwards. This could prove significant if you are loading large amounts of data.

Be aware that if you are using the COPY option of the LOAD command, the copy taken is not shown if you issue the command:

```
>db2 list history backup all for sample
```

You cannot use the copy you take as part of the LOAD command in a RESTORE command. To see any copies taken during the LOAD process, you need to issue the command:

```
>db2 list history load all for sample
```

I would therefore recommend taking a ‘restorable’ back-up as soon as the table is available (remembering that an online image copy doesn’t impact on table availability).

The above tests were performed on only a single table, but I hope I have shown that choosing the correct index create, load, and image copy step order can have a significant impact on the time taken to get the table in a state where you can perform insert operations, and it is worth seeing how you load data in your shop to see if you can reduce your load times.

C Leonard
Freelance Consultant (UK)

How DB2 supports and exploits Web services technologies

This article describes DB2 and Web services, with techniques for integrating information from multiple Web service providers and exposing the collective information through Web services. This article primarily explains why Web services are important to DB2 and how DB2 is being extended to provide optimized support for Web services. DB2 users may take advantage of Web services in two ways – as a provider and as a requestor.

WE ALL KNOW ABOUT WEB SERVICES

We are all aware of Web services as a part of an emerging technology that offers the dual promise of simplicity and pervasiveness because of the common XML foundation that underlies most Web service protocols. In the most primitive sense, Web services can be viewed as any mechanism by which an application service may be provided to other applications on the Internet. Web services are described in WSDL (Web Services Description Language). The WSDL description may be registered in the UDDI (Universal Description, Discovery, and Integration) repository. UDDI provides a set of APIs to register and search for Web services. Publicly available UDDI has been provided in DB2 and WebSphere.

CASE STUDY

A high-profile client company is in the telecom equipment-manufacturing sector. They work with a number of suppliers around the world. They were looking to implement cost-effective and speedy business processes that would help their purchase agents to get price quotes for proposed purchases, to make a purchase, and to check on the order status from the suppliers more quickly and efficiently than they currently have been doing using a traditional vendor/supplier relationship. By implementing these capabilities as Web services, purchase agent users in the company are able to call these services from a wide variety of platforms and application environments, while suppliers (as providers of the Web services) are free to implement the services in any manner they choose. Internal purchasing applications can use these Web services in a variety of ways. For instance, a simple purchasing application may use a private UDDI registry to look up a supplier in a user-specified city to get a price quote on a product. The UDDI search finds a supplier that offers the product and returns a link to the supplier's Web service operations. The purchasing agents can then issue a Web service request to obtain a price quote from this supplier, provide it to a buyer, and allow the buyer to place the order. The application can then be used to query the order status by invoking the appropriate operation for the supplier. So to summarize, each supplier offers three Web service operations that can be expressed with the following abstract signatures:

- GetQuote (in String partNum, in Integer qty, in Date desiredDate, out Decimal price, out String currency, out Date proposedDate).
- Purchase (in String partNum, in Integer qty, out Strong poNumber, out Date commitDate).
- GetStatus (in String poNum, out String status).

In the next two sections we discuss how these Web services can be invoked by DB2 applications, and how the Web services Object Run-time Framework (WORF) may implement them.

WEB SERVICE EXPLOITATION

The basic scenario, so far, has shown how an application can find and work with a single supplier. However, we often need to perform these operations over sets or groups. We discuss two examples: first, to find the best quote from a set of suppliers, and second, to report on the order status for all overdue orders. The remainder of this section illustrates how such set-oriented operations can easily be implemented by using Web services within database queries.

DB2 allows users to define new functions that may be invoked from SQL, thus extending the SQL language. These User-Defined Functions (UDFs) may be used for many purposes – calculations, transformations, or even to send messages. Using this facility, we can define a new function, GET_STATUS, to perform the getStatusOperation:

```
varchar(2Ø) GET_STATUS (url varchar(8Ø),po_num varchar(2Ø))
```

Here the return value is the purchase order (PO_status), and the input parameters are the URL to which the request is to be sent and the identity of the purchase order in question. To find the status of a specific order, say 12345, from a supplier that offers this service at http://www.Asupplier.com/getStatus, we could issue the following SQL statement:

```
values GET_STATUS (‘http://www.Asupplier.com/getStatus’,’12345’)
```

Figure 1 shows outstanding purchase orders and Figure 2 contains information about the Web service operations each supplier offers.

Figure 2 has already been populated in advance by queries to UDDI, or it could be replaced by calls to UDDI.

```
SELECT supplier, po_num, GET_STATUS('http://www.Asupplier.com/
getStatus',po_num) AS po_status
FROM purchase_orders
WHERE supplier = 'Asupplier'
```

In this simple example, we explicitly state the address of the service to be invoked. To find the status of all outstanding purchase orders for suppliers who offer a Web service interface, we could issue the following query:

```
SELECT p.supplier, p.po_num, GET_STATUS(s.url,p.po_num) AS po_status
FROM purchase_orders p, supplier_ops s
WHERE p.supplier = s.supplier
AND s.operation = 'getStatus'
```

Supplier	*po_num*	*date*	*part_num*	*qty*
ASupplier	12345	3/20/03	A4	34
BSupplier	12347	6/20/03	C7	43
CSupplier	34656	5/04/03	D7	3

Figure 1: Table for purchase_orders

If this query is commonly issued, it might be convenient to define a view to provide a simpler interface. The definition of this view would be:

```
CREATE VIEW order_status AS SELECT
p.supplier,p.po_num,GET_STATUS(s.url,p.po_num) As po_status
FROM purchase_orders p, supplier_ops s
WHERE p.supplier = s.supplier
ANS s.operation = 'getStatus'
```

To get the status, the following simple query could then be used:

```
SELECT *
FROM order_status
```

This query could, of course, be extended to exploit features of SQL. For instance, to sort the result by supplier, we simply

Supplier	*Operation*	*URL*
ASupplier	getStatus	http://www.ASupplier.com/getStatus
ASupplier	getQuote	http://www.ASupplier.com/getQuote
BSupplier	getQuote	http://www.BSupplier.com/getQuote
BSupplier	getStatus	http://www.BSupplier.com/getStatus

Figure 2: Table for supplier_ops

append an order by clause, such as:

```
SELECT po_num, supplier, status
FROM order_status
ORDER BY supplier
```

All the examples so far show how a Web service that returns a single value can be integrated with DB2 SQL, but we may need to handle multiple return values. The signature for the getQuote Web service is shown in Figure 3.

In order to access it from DB2 we turn this service into a DB2 table function with input and output parameters as shown in Figure 4.

To provide more meaningful context, the table function includes as outputs all the interesting input parameters. The GET_QUOTE table function is invoked within a query such as:

```
SELECT *
FROM TABLE (GET_QUOTE ('Asupplier','http://www.Asupplier.com/
getQuote,'52435FFA',25,'7/1/2ØØ1')) As t
```

This statement returns a table containing a single row with the response from this supplier. In order to deal with suppliers in other countries, the GET_QUOTE function contains currency units. To convert the price to dollars, we could try to maintain a table of currency conversion data manually. Given the volatile

	Name	*Type*
Input	partNum	string
	qty	integer
	desiredDate	date
Output	price	decimal
	currency	string
	proposedDate	date

Figure 3: Parameters for the getQuote Web service

	Name	*Type*
Input	supplier	varchar(30)
	url	varchar(80)
	part_num	varchar(20)
	qty	integer
	desired_date	date
Output	supplier	varchar(30)
	url	varchar(80)
	part_num	varchar(20)
	qty	integer
	desired_date	date
	price	decimal
	currency	varchar(10)
	proposed_date	date

Figure 4: Parameters for the GET_QUOTE table function

nature of foreign exchange, it would be better to invoke another Web service, perhaps provided by a foreign-exchange trading firm, to perform the conversion using the most current data. Input and output parameters for the DBS function to invoke this service are shown in Figure 5 below.

Using this additional service, we can now get a more accurate quote with a query such as:

```
SELECT t.supplier, t.part_num, t.qty, (t.desired_date – t.proposed_date)
As timeliness,
TO_DOLLARS(t.currency,t.price) AS cost
FROM TABLE (GET_QUOTE ('Asupplier',http:/www.Asupplier.com/
getQuote,'52435FFA',25,'7/1/2ØØ1')) AS t
```

	Name	*Type*
Input	currency	varchar(10)
	amount	decimal
Output	amount	decimal

Figure 5: Parameters for the TO_DOLLARS UDF

Here we make the columns explicit and, using the power of SQL, define an output column, 'timeliness', to reflect the difference between our desired date and the date proposed by the supplier for the part. We also use the currency conversion Web service to convert the quoted price to United States currency. This query returns a single row with the quote from a single vendor for a single part. Now, consider the case where we require quotes for a list of parts. We define a table, needed_parts, as shown below in Figure 6.

To get quotes on all of these parts from our supplier we can issue:

```
SELECT t.supplier,n.part_num,n.qty,(n.desired_date – t.proposed_date) AS
timeliness,TO_DOLLARS(t.currency,t.price)
FROM needed_parts n, TABLE(GET_QUOTE ('Asupplier','http:/
www.Asupplier.com/getQuote',n.part_num,noqty,n.desired_date)) t
```

This query returns a table of quotes for each part listed in the needed_parts table from one supplier.

To get quotes from each of our suppliers we can issue the following query:

part_num	*qty*	*desired_date*
34dsaf	20	7/1/2003
35gfds	34	8/1/2003
809gds	10	6/30/2003

Figure 6: Table and data for needed_parts

```
SELECT n.part_num,t.supplier,n.qty,(n.desired_date – t.proposed_date) AS
timeliness, TO_DOLLARS(t.currency,t.price)
FROM needed_parts n, supplier_ops s, TABLE (GET_QUOTE (s.supplier,
s.URL,n.part)num,n.qty, n.desired_date)) t
WHERE s.operation = 'GET_QUOTE'
ORDER BY n.part_num,timeliness
```

This query generates quotes for all the needed parts from all the suppliers that offer the getQuoteWeb service and returns a table of these quotes ordered by part number and timeliness. The queries use very powerful yet simple, standard DB2 SQL.

Finally, this query may be exposed as a Web service itself so that the purchasing agents can invoke the query from any location where they have access to the Internet. DB2 7.2 provides a simple mechanism that allows Web services to be created in support of such queries.

These examples show how Web services can be exploited within a database. By invoking Web services as UDFs, we can take advantage of the full power of SQL to perform queries across combinations of Web services and persistent data.

PROVIDING WEB SERVICES

Web services insulate users of the service from its implementation. The three services offered by suppliers in our scenario can be implemented in any manner that fulfils the contract defined by the service. Some might use J2EE platform, Extended Edition J2EE programming model implemented by the Websphere application server, while others might use the .Net model from Microsoft.

DB2 provides the Web services Object Run-time Framework (WORF) facility that can be used in conjunction with the Web-Sphere Application server to perform SQL queries, utilize DB2 XML extender routines to manipulate XML data, and invoke stored procedures. Within our scenario, a supplier could use this facility to implement the three Web services just described. Assume that the supplier has a table for orders in their order management database as shown in Figure 7 .

po_num	*customer*	*part_num*	*qty*	*commit_date*	*status*
78453	yourMfgCo	A4	23	5/23/02	ONTIME
12347	myMfgCo	C7	200	6/20/02	ONTIME
53456	theirMfgCo	B12	5	4/23/02	COMPLETE
35335	yourCo	A3	7	4/15/02	SHIPPED

Figure 7: Table and data for orders

The getStatus service could be implemented by effectively wrapping a Web service around the query:

```
SELECT status FROM orders WHERE po_num = :input
```

In this query the parameter input would be provided within the Web service request and the WORF runtime code would execute the query and return the result (status) in the Web service response. In a later section we discuss the details of the WORF facility.

DB2 AS A WEB SERVICE REQUESTOR

A database is a powerful vehicle for information integration. The ability to pull information from a variety of service providers puts databases in a unique position to analyse and combine information and to provide powerful querying capabilities. As we discussed in an earlier section, we want to make the use of Web services a natural extension to the DB2 SQL environment. To achieve this we must address two sets of problems. First, the signature of the UDF must be mapped to the signature of the Web service it implements. Then this data must be used to construct and send the SOAP message to the indicated service provider. After the response is received, the reply must be decomposed into the set of result parameters that the user expects. Our implementation architecture uses two layers of functions – a set of SOAP UDFs that are specific for each WSDL operation and a set of underlying functions that actually perform the Web service invocation. The following sections describe the design of these functions in further detail.

WEB SERVICE CONCEPT IN SQL

For a service requestor to send an invocation to a service provider, the following information is necessary:

- The URI (Uniform Resource Identifier) of the target object, including optional header information, such as SOAP action.
- The name of an operation to execute, including its input and output message format.
- Binding information with respect to transport protocol, encoding style, name spaces, etc.

The abstract interface (operations and messages), the protocol bindings, and the access ports for deployed services are described in WSDL. The WSDL description of a sample Web Service that returns the current stock quote for a given stock symbol is shown below:

```
<?xml version=“1.0” encoding=“UTF-8”?>
<definitions name=“StockQuoteServiceRemoteInterface”
targetNamespace=“http://www.stockquoteservice.com/definitions/
StockQuoteServiceRemoteInterface”
xmlns=“http://schemas.xmlsoap.org/wsdl/”
xmlns:tns=“http://www.stockquoteservice.com/definitions/
StockQuoteServiceRemoteInterface”
xmlns:xsd=“http://www.w3.org/2001/XMLSchema”
xmlns:soap=“http://schemas.xmlsoap.org/wsdl/soap/”>
<message name=“getQuoteRequest”>
<part name=“symbol” type=“xsd:string”/>
</message>
<message name=“getQuoteResponse”>
<part name=“result” type=“xsd:float”/>
</message>
<portType name=“StockQuoteServiceJavaPortType”>
<operation name=“stockQuote”>
<input name=“getQuoteRequest” message=“tns:getQuoteRequest”/>
<output name=“getQuoteResponse” message=“tns:getQuoteResponse”/>
</operation>
</portType>
<binding name=“StockQuoteServiceBinding”
```

```
type="tns:StockQuoteServiceJavaPortType">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/
http"/>
<operation name="stockQuote">
<soap:operation soapAction="" style="rpc"/>
<input name="getQuoteRequest">
<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
namespace="http://tempuri.org/StockQuoteService"/>
</input>
<output name="getQuoteResponse">
<soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"
namespace="http://tempuri.org/StockQuoteService"/>
</output>
</operation>
</binding>
<service name="StockQuoteServiceService">
<port name="StockQuoteServicePort"
binding="binding:StockQuoteServiceBinding">
<soap:address location="http://localhost:8080/soap/servlet/rpcrouter"/>
</port>
</service>
</definitions>
```

Below is a SOAP request envelope. The request is submitted through HTTP to a service endpoint as specified in the HTTP header. The SOAP body shows the method name and the name space for the method, as well as the input parameters.

```
POST /soap/servlet/rpcrouter HTTP/1.0
Host: localhost:8080
Connection: Keep-Alive
Content-Type: text/xml
SOAPAction: ""
Content-Length: 393
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/
envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns:getQuoteRequest xmln:ns="http://tempuri.org/StockQuoteService">
<symbol xsi:type="xsd:string">IBM</symbol>
</ns:getQuoteRequest>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

The following is a SOAP response envelope. The response is returned through HTTP to the requestor.

```
HTTP/1.0 200 OK
Date: Fri, 15 Mar 2002 00:19:47 GMT
Status: 200
Content-Type: text/xml; charset=utf-8
Servlet-Engine: WebSphere Application Server (JSP 1.1; Servlet 2.2; Java
1.3.0; Linux 2.4.7-10smp
x86; java.vendor=IBM Corporation)
Content-Length: 465
Set-Cookie: JSESSIONID=JvGWBVyjLplbVdPzNG92M04d;Path=/soap
Server: WebSphere
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:getQuoteResponse xmlns:ns1="http://tempuri.org/StockQuoteService"
SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:float">57.0</return>
</ns1:getQuoteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
HTTP/1.0 200 OK
Date: Fri, 15 Mar 2002 00:19:47 GMT
Status: 200
Content-Type: text/xml; charset=utf-8
Servlet-Engine: WebSphere Application Server (JSP 1.1; Servlet 2.2; Java
1.3.0; Linux 2.4.7-10smp
x86; java.vendor=IBM Corporation)
Content-Length: 465
Set-Cookie: JSESSIONID=JvGWBVyjLplbVdPzNG92M04d;Path=/soap
Server: WebSphere
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi=http://www.w3.org/1999/XMLSchema-instance
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body>
<ns1:getQuoteResponse xmlns:ns1="http://tempuri.org/StockQuoteService"
SOAP-ENV:encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/">
<return xsi:type="xsd:float">57.0</return>
</ns1:getQuoteResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

At a conceptual level, the deployed Web service in the first example has an abstract interface, such as:

```
Float stockQuote (symbol string)
```

SQL is extensible through functions, which may be either built-in or user-defined. Functions accept input parameters and return scalar values in the case of scalar functions, or entire tables in the case of table functions. SQL functions provide the necessary language hooks for using Web services. A SQL function for a Web service acts as a SOAP requestor, and we call these functions SOAP UDFs. SOAP UDFs compose SOAP requests, submit the request to the provider, receive the response, and return the response to the SQL engine. For composing and sending the SOAP request, the SQL function needs the service end-point (URL of the service provider), name of the method, name space, and the input and output parameters.

The abstract interface for the stock quote Web service just shown could be registered as a SQL function as below:

```
CREATE FUNCTION STOCK_QUOTE (symbol char(3))
                 RETURNS double;
```

The implementation of the SQL function STOCK_QUOTE generates a SOAP request envelope as shown above, sends the request to the service provider, receives the response as also shown above, retrieves the stock quote return value, and returns it as a function result.

An SQL statement that combines the results from calling the stockQuote Web service with data from the stock_watch table is:

```
SELECT name,symbol,STOCK_QUOTE(symbol) AS quote
FROM stock_watch
```

name	*symbol*	*quote*
International Business Machines	IBM	114.35
Microsoft	MSFT	70.58

Figure 8: Results from execution of SL statement

The results from execution of this SQL statement are shown in Figure 8.

Web services may require complex input parameters and generate complex XML output. A database application might directly feed XML input to the function, retrieve the entire XML output, and process it in native XML. Other applications may require a simpler function interface that provides basic input parameters and returns basic output parameters.

The following subsections describe the option of decomposing complex SQL into basic SQL output parameters.

Let's use the stock quote example for demonstration purposes.

The input message:

```
<stockticker>
  <symbol>IBM</symbol>
</stockticker>
```

might result in the output message:

```
<stock_quotes>
  <stock_quote>
    <last> 113.40 </last>
    <ask> 113.50 </ask>
    <bid> 113.26 </bid>
    <change> +0.310005188 </change>
    <pctchange> +0.27% </pctchange>
    <symbol> IBM </symbol>
    <time> 1:38pm </time>
  </stock_quote>
</stock_quotes>
```

XML input and output

The user of the SOAP UDF provides the input parameters in XML format, and the SOAP UDF returns the service provider output in XML format. Using WSDL, we can register an SQL-bodied SOAP UDF, which sends a stock-ticker XML fragment in a SOAP envelope to the service provider and returns the stock quote as an XML fragment, as follows:

```
CREATE FUNCTION STOCK_QUOTE3 (stockticker varchar(5))
RETURNS table
```

```
(last varchar(8),
ask varchar(8),
bid varchar(8),
change varchar(15),
pctchange varchar(1Ø),
symbol varchar(5),
time varchar(2Ø))
LANGUAGE SQL READS SQL DATA
EXTERNAL ACTION NOT DETERMINISTIC
RETURN
WITH
--1. Perform type conversions and prepare SQL input parameters --
-- for SOAP envelope
soap_input (in)
AS
(VALUES XMLElement(NAME "getRTQuote",
XMLElement(NAME "stockticker",
XMLElement(NAME "symbol", stockticker)))),
--2. Submit SOAP request with input parameter and receive SOAP --
-- response
soap_output (out)
AS
(values soaphttp (
'http://localhost:8Ø8Ø/soap/servlet/rpcrouter'
(SELECT in FROM soap_input)))

--3. Shred SOAP response and perform type conversions to get SQL
-- output parameters
SELECT x.last, x.ask, x.bid, x.change, x.pctchange,
x.symbol, x.time
FROM Table (TableEXTRACT (
(select out from soap_output),
'/stock_quotes/stock_quote',
'./last', './ask', './bid', './change',
'./pctchange', './symbol', './time')
AS x (last AS varchar(8), ask AS varchar(8),
bid AS varchar(8), change AS varchar(15),
pctchange AS varchar(1Ø), symbol AS varchar(5),
time AS varchar(2Ø));


CREATE FUNCTION STOCK_QUOTE1 (stockticker XML) RETURNS XML
LANGUAGE SQL CONTAINS SQL
EXTERNAL ACTION NOT DETERMINISTIC
RETURN (VALUES soaphttp('http:/localhost:8Ø8Ø/soap/servlet/
rpcrouter','',stockticker));
```

A SQL application can use SQL/XML function to construct the XML input and retrieve XML output:

```
SELECT symbol,STOCK_QUOTE1(
                                        XMLElement(NAME "getRTQuote",
                                        XMLElement(NAME "stockticker",
                               XMLElement(NAME "symbol",symbol))) AS quote
FROM myportfolio;
```

SOAPHTTP() is the DB2 SOAP requestor. This function generates the SOAP envelope and communicates with the SOAP provider through HTTP. It receives the SOAP response, parses the response, and returns the SOAP body to the invoker.

Editor's note: this article will be concluded next month.

Vikas Baruah
Senior Technical Specialist
American Management Systems (USA)

November 2000 – October 2003 index

Items below are references to articles that have appeared in *DB2 Update* since November 2000. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

DB2 news

Embarcadero Technologies has announced Version 2.0 of DT/Studio, its extraction, transformation, and loading application that transforms raw data into useful business information.

Embarcadero DT/Studio 2.0 offers a visual development experience, which is meant to enable rapid data integration. This new version also provides enterprises with transparency and access to data.

DT/Studio 2.0 has message queue support, allowing users to integrate real-time data sources with more traditional static information.

The new version has a model-driven design environment to analyse source systems and to design and implement target data structures. It integrate data across a wide variety of platforms, including DB2, Oracle, Microsoft SQL Server, Sybase, flat files, message queues, or any JDBC accessible data source.

For further information contact:
Embarcadero Technologies, 425 Market Street, Suite 425, San Francisco, CA 94105, USA.
Tel: (415)834 3131.
URL: http://www.embarcadero.com/products/dtstudio/index.asp.

* * *

VERITAS Software has announced Cluster Server agents for automated recovery and increased availability of Linux enterprise environments running DB2, MySQL, and Oracle databases.

Increased IT service levels can now be achieved by constantly monitoring the health of databases with VERITAS Cluster Server agents in conjunction with technologies that manage server and application performance and automate server provisioning.

VERITAS Cluster Server agents continually monitor the status of each cluster resource, up to 32 nodes, including the database, disks, application, file system, volumes, and network. When a fault is detected, agents automatically initiate a failover and restart the system, thereby minimizing disruption and increasing utilization of applications.

VERITAS Indepth increases application performance by proactively monitoring, analysing, and tuning applications running on DB2 UDB and Oracle databases, as well as for J2EE applications during the development, testing, and production phases.

VERITAS OpForce 3.0 automates routine, scheduled, or urgent priority tasks helping customers provide better service levels at lower costs and with fewer risks by managing resources according to fluctuating demand.

For further information contact:
VERITAS, 350 Ellis Street, Mountain View, CA 94043, USA.
Tel: (650) 527 8000.
URL: http://www.veritas.com/products/category/ProductDetail.jhtml?productId=clusterserver.