95

DB2 update

September 2000

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 1997 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

DB2 Update on-line

Code from *DB2 Update* can be downloaded from our Web site at http://www.xephon.com/db2update.html; you will need the user-id shown on your address label.

Editor

Trevor Eddolls

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Contributions

Articles published in *DB2 Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Printed in England.

Shadow system catalog

One of the most frequent problems in a DB2 environment with very large system tables is the slow performance during query catalog access. In order to improve the performance and reduce lock contention among users on catalog tables, I have created a shadow system catalog, which is an image of the real DB2 catalog. This alternative catalog, refreshed with a batch REXX procedure, can be personalized to your own needs, building additional indexes on any tables, and reorganized.

Figure 1 shows the differences between the system catalog (marked B for before) and the shadow system catalog (marked A for after). The rows marked with an asterisk are the new indexes.

PROCEDURE DESCRIPTION

The procedure is divided into two phases. The first analyses and builds jobs, the second is for the execution of the refresh of the shadow system catalog. With correct parameter customization, the tool performs the following steps:

- Start DB2 catalog RO – it's read-only to avoid any updates during data unload.
- Unload DATA catalog and update creator in syspunch datasets. To unload data, the procedure uses a modified version of the program DSNTIAUL called DSNTIA01 – described below.
- Start DB2 Catalog RW.
- Sort data.
- Load data into the shadow system catalog.
- Reset the 'copy pending state' of spacenams.
- Delete work areas.
- Runstats shadow system catalog tablespaces.

- Perform a a reorg of the shadow system catalog tables that need to be reorganized.
- Runstats shadow system catalog tablespaces.

PARAMETER DESCRIPTION

The following parameters describe how you can customize the REXX EXEC:

- SUBSYS – the DB2 subsystem name.
- CREALT – the creator of the shadow system catalog.
- DATAB – the database name of the shadow system catalog.
- STOGROUP – the stogroup of the shadow system catalog.
- AUTOSUB – this can have a value of '*', 'NO', or 'YES'. When the parameter is 'YES', the refresh of the shadow system catalog will be executed automatically. When the value is '*' or 'NO', only certain functions will be executed.
- DB2CRE – the creator of the DB2 work view. This can have a value of '*' or 'other user-id'. When the value is '*' the creator is equal to the TSO user-id.
- CATNAM – the creator of the DB2 catalog for the reorg procedure. This can have a value of '*', 'SYSIBM', or 'other creator'.When the value is '*', the catalog creator is equal to SYSIBM.

The procedure has been tested in MVS 4.3.0 and can be used in a DB2 Version 2.3 or Version 3.1 environment. Currently, the tool is used on a development DB2 with approximately 100 databases and 55,000 tablespaces and indexes. It takes about 2 hours for a complete refresh.

JOB DESCRIPTION

At the end of the first phase, the procedure builds sequential files that contain the jobs. The dataset names are:

- Hiwork.subsys.database.JOBUNLO.

```
 INDEX NAME TSNAME        CLUSTERING  CLUSTERED     NLEAF     NLEVELS  CLUSTERRATIO
                            B   A      B   A        B     A    B   A      B    A

* DSNCAXØ2  SYSCOLAUTH          N          N              4        2     89
* DSNCAXØ1  SYSCOLAUTH          Y          Y              4        2    1ØØ
  DSNTNXØ1  SYSCOLDIST      N   Y      N   Y      141    82    3   2     89  1ØØ
  DSNTPXØ1  SYSCOLDISTSTATS N   Y      Y   Y       46    28    2   2     98  1ØØ
  DSNTCXØ1  SYSCOLSTATS     N   Y      Y   Y       57    35    2   2     98  1ØØ
  DSNDCXØ1  SYSCOLUMNS      N   N      N   Y     6568  3859    4   3     93  1ØØ
  DSNUCHØ1  SYSCOPY         N   Y      N   Y     23Ø3  1ØØ6    3   3     46  1ØØ
  DSNUCXØ1  SYSCOPY         N   N      Y   N      26Ø   134    3   4     99   45
  DSNDDHØ1  SYSDATABASE     Y   Y      N   Y        1     1    1   1     81  1ØØ
  DSNADHØ1  SYSDBAUTH       N   N      N   Y        4     2    2   2     71  1ØØ
  DSNADXØ1  SYSDBAUTH       N   N      N   N        2     2    2   2     72   28
* DSNDRXØ1  SYSDBRM             Y          Y             12        2         1ØØ
* DSNFDXØ1  SYSFIELDS           Y          Y             66        2         1ØØ
* DSNFDXØ2  SYSFIELDS           N          N             52        2          82
* DSNFKXØ1  SYSFOREIGNKEYS      Y          Y             79        2         1ØØ
  DSNDXXØ1  SYSINDEXES      N   N      N   Y      336   188    3   3     86  1ØØ
  DSNDXXØ2  SYSINDEXES      N   N      N   Y      227   125    3   2     82   97
* DSNDXXØ3  SYSINDEXES          N          N            2ØØ        3          79
* DSNIPXØ1  SYSINDEXPART        Y          Y            31Ø        3         1ØØ
  DSNTXXØ1  SYSINDEXSTATS   N   Y      Y   Y       34    16    2   2     99  1ØØ
* DSNKEXØ1  SYSKEYS             Y          Y            3ØØ        3         1ØØ
* DSNKEXØ2  SYSKEYS             N          N            3ØØ        3          84
  DSNDDFLL  SYSLOCATIONS    Y   Y      Y   Y        1     1    1   1    1ØØ  1ØØ
  DSNDDFLM  SYSLUMODES      Y   Y      Y   Y        Ø     Ø    Ø   Ø      Ø    Ø
  DSNDDFLN  SYSLUNAMES      Y   Y      Y   Y        1     1    1   1    1ØØ  1ØØ
  DSNDDFMS  SYSMODESELECT   Y   Y      Y   Y        Ø     Ø    Ø   Ø      Ø    Ø
  DSNKKXØ1  SYSPACKAGE      N   N      N   Y     325Ø  1828    4   4     94  1ØØ
  DSNKKXØ2  SYSPACKAGE      N   N      N   Y     1575   876    3   3     94  1ØØ
  DSNKAXØ1  SYSPACKAUTH     N   N      N   Y     1762  1Ø14    3   3     79  1ØØ
  DSNKAXØ2  SYSPACKAUTH     N   N      N   N    11625   591    4   3     91   95
  DSNKAXØ3  SYSPACKAUTH     N   N      N   Y     16Ø2   881    3   3     79   99
  DSNKDXØ1  SYSPACKDEP      N   N      N   Y    6Ø1Ø5  1631    4   3     95  1ØØ
  DSNKDXØ2  SYSPACKDEP      N   N      N   N     1837  1155    3   3     64   27
  DSNKLXØ1  SYSPACKLIST     N   N      N   Y      275   177    3   3     88   99
  DSNKLXØ2  SYSPACKLIST     N   N      N   N    1Ø168   527    4   3     77   76
* DSNKLXØ3  SYSPACKLIST         Y          Y            13Ø        2         1ØØ
  DSNKSXØ1  SYSPACKSTMT     N   N      Y   Y    41989 23627    4   4     96  1ØØ
  DSNKYXØ1  SYSPKSYSTEM     N   N      Y   Y        Ø     Ø    Ø   Ø      Ø    Ø
  DSNPPHØ1  SYSPLAN         Y   Y      N   Y       85    41    2   2     82  1ØØ
  DSNAPHØ1  SYSPLANAUTH     N   N      N   Y      19Ø    82    3   2     81  1ØØ
  DSNAPXØ1  SYSPLANAUTH     N   N      N   N       47    15    2   2     72   82
  DSNGGXØ1  SYSPLANDEP      N   N      N   Y       88    22    3   2     65  1ØØ
  DSNKPXØ1  SYSPLSYSTEM     N   N      Y   Y        Ø     Ø    Ø   Ø      Ø    Ø
```

Figure 1a: Catalog differences

```
  INDEX NAME TSNAME        CLUSTERING  CLUSTERED    NLEAF     NLEVELS  CLUSTERRATIO
                            B   A      B   A      B     A     B   A      B    A

  DSNDLXØ1  SYSRELS         N   N      N   Y     56    26    2   2     58  1ØØ
* DSNRLXØ3  SYSRELS             N          N           28        2           74
  DSNAGHØ1  SYSRESAUTH      Y   Y      N   Y     13     6    2   2     84  1ØØ
  DSNAGXØ1  SYSRESAUTH      N   N      N   N     1Ø     5    2   2     74   55
* DSNSTXØ1  SYSSTMT             Y          Y          349        3          1ØØ
* DSNSTXØ2  SYSSTMT             N          N           55        2           89
  DSNSSHØ1  SYSSTOGROUP     Y   Y      N   Y      1     1    1   1     81  1ØØ
  DSNSSXØ1  SYSSTRINGS      N   N      N   Y      1     1    1   1     93  1ØØ
  DSNDYXØ1  SYSSYNONYMS     N   N      N   Y    436   2Ø8    3   3     73  1ØØ
  DSNATXØ1  SYSTABAUTH      N   N      N   Y   1116   26Ø    3   2     65  1ØØ
  DSNATXØ2  SYSTABAUTH      N   N      N   N  18784  33Ø3    4   3     57   6Ø
* DSNTAXØ1  SYSTABLEPART        Y          Y          2Ø3        3          1ØØ
* DSNTAXØ2  SYSTABLEPART        N          N          2Ø3        3           84
  DSNDTXØ1  SYSTABLES       N   N      N   Y    469   2Ø8    3   3     74  1ØØ
  DSNDTXØ2  SYSTABLES       N   N      N   N    471   236    3   3     82   93
  DSNDTXØ3  SYSTABLES           N          N          222        3           79
* MIGØØ     SYSTABLES           N          N           49        2           28
  DSNDSXØ1  SYSTABLESPACE   Y   N      N   N    217   114    3   2     86   79
* DSNTSXØ1  SYSTABLESPACE       N          N          114        2           79
* DSNTSXØ2  SYSTABLESPACE       N          Y          15Ø        2           99
  DSNTTXØ1  SYSTABSTATS     N   Y      Y   Y     34    16    2   2     99  1ØØ
  DSNAUHØ1  SYSUSERAUTH     Y   Y      N   Y      1     1    1   1     9Ø  1ØØ
  DSNAUXØ2  SYSUSERAUTH     N   N      N   N      1     1    1   1     9Ø   5Ø
  DSNDDFUN  SYSUSERNAMES    Y   Y      Y   Y      1     1    1   1    1ØØ  1ØØ
  DSNGGXØ2  SYSVIEWDEP      N   N      N   Y     7Ø    33    2   2     69  1ØØ
* DSNVWXØ2  SYSVIEWS            Y          Y          158        3          1ØØ
* DSNVWXØ1  SYSVIEWS            N          Y          158        3           99
  DSNVTHØ1  SYSVTREE        Y   Y      N   Y    1ØØ    36    2   2     58  1ØØ
```

Figure 1b: Catalog differences

- Hiwork.subsys.database.JOBSORT.
- Hiwork.subsys.database.JOBLOAD.
- Hiwork.subsys.database.JOBSTRT.
- Hiwork.subsys.database.JOBDELE.
- Hiwork.subsys.database. JOBRUNS.

- Hiwork.subsys.database.JOBREOR.

CHECKLIST FOR INSTALLATION

You should take the following steps to install the components of the REXX procedure:

- Allocate a USER.LIBRARY.
- Copy all REXX, macros, and procs into the USER.LIBRARY:
  - REXX – DB2SSC0, DB2SSC1, DB2SSC2, DB2FOR0, and DB2PAR0.
  - Macro – MDB2008, MDB2032, MDB2006, and MDB2031.
  - Proc – DB2REXX0, DB2REXX1, and DSNUPROD.
  - Sample jobs – run procedure, create shadow system catalog.
- Customize DB2PAR0 REXX for the global environment.
- Customize DB2REXX1 and DSNUPROD to suit your environment.
- Create the shadow system catalog using the sample job called ‘Sample batch to create shadow system catalog database’. You can modify the number of spaces to suit your own DB2 catalog.
- Copy the program DSNTIAUL with a new name, modify the new source according to the following changes, make bind, and grant the new plan.
- Customize the job ‘Sample batch submit procedure’ and submit the procedure.

Editor’s note: the code for this program is available from our Web site at www.xephon.com/extras/shadocat.txt.

Giuseppe Rendano
DB2 System Programmer (Italy)

JDBC and SQLJ

INTRODUCTION

The Java programming language has become a mainstay for building corporate Internet applications since Sun Microsystems launched the technology in 1995.

The first releases of Java were able to process only sequential files. This was a major limitation for anyone attempting to develop ‘industrial’ applications that needed database access.

That ‘missing’ common database interface rolled out shortly after Java itself – its name is Java Database Connectivity (JDBC).

By using the JDBC programming interface, Java programmers can request a connection with a database, then send query statements using SQL and receive the results for processing.

JDBC handles the actual connection, sending queries and data to and from the database. According to Sun, specialized JDBC drivers are available for all major databases – including relational databases from Oracle, IBM, Microsoft, Informix, and Sybase – as well as for any data source that uses Microsoft’s Open Database Connectivity system.

However, JDBC is only a dynamic SQL model.

Static SQL has a number of advantages over dynamic SQL, especially in the areas of performance, ease of use, and manageability. This is why Oracle, Tandem, and IBM introduced, in April 1997, SQLJ, another programming interface, which implements a static SQL model for Java applications.

This article will discuss how you can use JDBC and SQLJ to write Java applications that access DB2 for OS/390 databases.

It provides an overview that explains what JDBC and SQLJ are, and guidelines for running Java JDBC and SQLJ programs and configuring these environments.

It also provides sample applications, which will help readers to understand JDBC and SQLJ programming concepts.

As Java programs on OS/390, DB2 for OS/390 JDBC and SQLJ programs run in the OS/390 OpenEdition environment, so the OpenEdition environment must be initialized on your OS/390 system in order to use JDBC or SQLJ to access DB2.

JDBC

JDBC is a Java API that Java applications use to access any relational database.

JDBC enables you to write Java applications that access local DB2 data or remote relational data on a server that supports DRDA.

Sun Microsystem’s JavaSoft developed the specifications for a set of APIs that allow Java applications to access relational data. The APIs are defined within 16 Java classes that support basic SQL functionality for connecting to a database, executing SQL statements, and processing results. Together, these interfaces and classes represent the JDBC capabilities by which a Java application can access relational data.

JDBC applications use a dynamic SQL model and do not require precompiles or binds.

DB2 for OS/390 JDBC implementation

The DB2 for OS/390 JDBC driver is implemented as a type 1 driver.

A type 1 driver is implemented as a JDBC-ODBC bridge. This type of JDBC driver uses an existing ODBC driver and translates all the JDBC method calls into ODBC function calls.

DB2 for OS/390 Support for ODBC is implemented via the DB2 Call Level Interface (CLI).

This is why, in order to run DB2 JDBC applications, you should first install, as a pre-requisite, the DB2 Call Level Interface FMID.

Installing and configuring DB2 JDBC

As explained above, in order to use JDBC, you should first install the CLI FMID which brings ODBC DB2 support.

Installing CLI/ODBC

To install CLI/ODBC, you should RECEIVE/APPLY/ACCEPT the corresponding FMID during the DB2 for OS/390 installation.

For DB2 Version 510 it's ODBC FMID JDB5517. For Version 610, it's JDB6617.

Installing JDBC

To install JDBC, you should then RECEIVE/APPLY/ACCEPT the JDBC FMID.

For DB2 Version 510, it's JDBC FMID JDB5512. For 610, it's JDB6612.

During the installation of JDBC, an Open Edition HFS will be created.

This HFS will be mounted in the default directory /usr/lpp/db2.

JDBC programming structure

A typical JDBC application should execute the following steps in sequence:

- Import the JDBC package.
- Load the JDBC driver.
- Identify the target DB2 subsystem.
- Connect to the DB2 subsystem.
- Create a SQL statement.
- Create a result set.
- Execute the SQL query.
- Display the result set.

- Close the result set.
- Close the SQL statement.
- Disconnect from the DB2 subsystem.

The following sections will describe how you should write Java statements to implement these different steps.

Importing the JDBC package

Before you can invoke any JDBC functions in your application program, you must first import the JDBC package:

```
import  java.sql.*;           // import JDBC package
```

Loading the JDBC driver

The Java application should load the JDBC driver.

The Class.forName method loads the appropriate driver, in this case, the DB2 for OS/390 JDBC driver (ibm.sql.DB2Driver):

```
Class.forName("ibm.sql.DB2Driver");
```

Identifying the target DB2 subsystem

The Java application must identify the DB2 subsystem it wants to connect to, by passing a URL to the 'DriverManager'.

The basic structure of the URL for a DB2 for OS/390 subsystem is:

```
jdbc:db2os390:<location-name>
```

where 'location-name' is the DB2 location name specified in DDF (Distributed Data Facility) parameters:

```
String url = "jdbc:db2os390:DB2SDRDA";
```

Connecting to the DB2 subsystem

Then, the programmer must use the 'getConnection()' method to create a 'Connection' instance to connect to the database.

This is done by specifying the URL of the DB2 subsystem:

```
Connection con = DriverManager.getConnection (url);
```

Creating a SQL statement

The programmer must also create a SQL Statement instance using the 'createStatement()' method:

```
Statement stmt = con.createStatement();
```

Creating a result set and executing the query

In order to get the result of the query from DB2, the Java application must use the 'ResultSet' Java class. A result set is a Java object that you can use to retrieve rows from a SQL query.

The 'executeQuery' method executes the query and generates a ResultSet instance:

```
ResultSet rs =
     stmt.executeQuery("SELECT NAME, CREATOR
                          FROM SYSIBM.SYSDATABASE");
```

Displaying the result set

The 'ResultSet' class implements several methods that are very useful for manipulating data from SQL queries:

- The 'next()' method skips to the next record of the result set and returns 'false' if the current record is the last record of the result set.
- The 'getString(n)' method returns the *n*th column of the result row.

For example, the following code displays all the rows returned by the previous SQL query:

```
while (rs.next())
 {
   String c1 = rs.getString(1);     // return the first column
   String c2 = rs.getString(2);     // return the second column
   System.out.println("Result : " + c1 + "  " + c2);
 }
```

Closing the result set and SQL statement

When the result of the query has been manipulated, the result set and the SQL statement should be closed.

The ‘close()’ method closes the result set and the statement and frees all resources associated with the statement:

```
Rs.close() ;
stmt.close();
```

Disconnecting from the DB2 subsystem

The last step is to disconnect from the DB2 susbsystem.

It is done using the ‘close()’ method, which closes the DB2 connection and frees all resources associated with this connection:

```
con.close();
```

Handling SQL errors and SQL warnings

The programmer should use the JDBC class Java.sql.SQLException for error handling.

JDBC generates a SQLException when a SQL statement returns a negative SQLCODE.

You can use the getErrorCode method to retrieve SQLCODEs and the getSQLState method to retrieve SQLSTATEs.

To handle SQL errors in your JDBC application, import the Java.sql.SQLException class, and use the Java error handling try/catch blocks to modify program flow when a SQL error occurs.

For example:

```
try
 {
  String url = "jdbc:db2os390:DB2SDRDA";
  conn = DriverManager.getConnection(url);
 }
catch(SQLException e)
 {
  System.out.println("SQLCODE returned: " + e.getErrorCode());
 }
```

Because SQL warnings do not generate SQLExceptions, you must invoke the GetWarnings method after you execute a SQL clause to check for a SQL warning. GetWarnings returns the first warning code that a SQL statement generates:

```
SQLWarning SQLWarn;
ResultSet rs = stmt.executeQuery("SELECT * FROM SYSIBM.SYSTABLES");
if (SQLWarn = execCtx.GetWarnings() |= null)
  then System.out.println("SQLWarning " + SQLWarn);
```

Preparing and running JDBC programs

After writing a JDBC application, you must generate an executable form of the JDBC programs.

Procedure to compile a JDBC java source program

In order to compile a JDBC program, you should create a script like export_compile.sh, which exports the required DB2 library in the CLASSPATH environment variable:

```
BROWSE
 -- /u/i990557/java/db2/jdbc/export_compile.sh Line 00000000 Col 001 070
 Command ===>                                              Scroll ===> PAGE
**************************** Top of Data ******************************
  export  CLASSPATH=/usr/lpp/db2/classes/db2jdbcclasses.zip:$CLASSPATH
  export  CLASSPATH=.:$CLASSPATH
**************************** Bottom of Data ****************************
```

A possible alternative is to modify your /etc/profile to include these libraries. Doing it that way, these libraries will be directly included in your CLASSPATH during log-on processing to Unix services.

Then, you should run this script to modify your CLASSPATH:

```
I990557:/u/i990557/java/db2/jdbc: >. export_compile.sh
I990557:/u/i990557/java/db2/jdbc: >echo $CLASSPATH
.:/usr/lpp/db2/classes/db2jdbcclasses.zip:.:/usr/lpp/java/J1.1/lib/
classes.zip
```

At this point, you are able to invoke javac, to compile your JDBC source program :

```
I990557:/u/i990557/java/db2/jdbc: >javac program_name.java
I990557:/u/i990557/java/db2/jdbc: >
```

Procedure to run a JDBC program

In order to run your Java program, you should modify two other environment variables – LIBPATH and DSNAOINI:

```
export  LIBPATH=/usr/lpp/db2/lib:$LIBPATH
export  DSNAOINI="/u/i990557/java/db2/jdbc/DSNAOINI"
```

Because DB2 JDBC is implemented as a type 1 driver, it is dependent on DB2 for OS/390 CLI support.

The CLI parameters are set up in the DSNAOINI file:

```
BROWSE -- /u/i990557/java/db2/jdbc/DSNAOINI -- Line 00000000 Col 001 027
 Command ===>                                            Scroll ===> PAGE
***************************** Top of Data ******************************
; This is a comment line...
[COMMON]
MVSDEFAULTSSID=DB2S
; Example SUBSYSTEM
[DB2S]
MVSATTACHTYPE=CAF
PLANNAME=DSNACLI
**************************** Bottom of Data ****************************
```

After exporting these two variables, you are able to start your Java program:

```
I990557:/u/i990557/java/db2/jdbc: >java JDBC_Query_Main SYSADM
*** Running JDBC_Query_Main.java ***
*** Query for CREATOR = SYSADM ***
*** Connected to DB2 subsystem ***
*** Creating result set ***
DSNCV    SYSADM
DSNDB07  SYSADM
DSNDDF   SYSADM
DSNRGFDB SYSADM
DSNRLST  SYSADM
PPREDB04 SYSADM
*** Closing result set ***
*** Closing statement ***
I990557:/u/i990557/java/db2/jdbc: >
```

SQLJ

With SQLJ, Java applications containing SQLJ clauses are translated by a Java precompiler to produce modified Java code and a platform-independent description of the SQLJ clauses called a profile.

The profile is then customized to produce a DB2 for OS/390-dependent DBRM (Database Request Module), which is then bound into a package or plan, and the application is executed through the JVM provided by Java for OS/390.

SQLJ was developed to complement the dynamic JDBC SQL model with a static SQL model. Unlike the ODBC and JDBC dynamic models, the static model provides strong type checking at application translation time, better manageability of data access through separation of package owner from package runner, and, because all SQL is compiled, a vehicle for better performance.

SQLJ provides support for embedded static SQL in Java applications.

Some of the major differences between SQLJ and JDBC are:

- SQLJ follows the static SQL model, and JDBC follows the dynamic SQL model.
- SQLJ source programs are smaller than equivalent JDBC programs because certain code that the programmer must include in JDBC programs is generated automatically by SQLJ ‘clauses’.
- SQLJ source programs cannot be directly compiled using javac. SQLJ source files should have the ‘.sqlj’ extention and must be translated to standard Java source (extension ‘.java’) using the SQLJ translator.

SQLJ clauses

In a SQLJ program, all statements that are used for database access are in SQLJ clauses.

A SQLJ clause begins with the characters #sql and contains a SQL statement that is enclosed in curly brackets.

An example of a SQLJ clause is:

```
#sql {DELETE FROM EMP};
```

Java variables and host variables

To pass data between a Java application program and DB2, you must use host variables.

A Java host variable is a Java simple identifier preceded by a colon.

The following SQLJ clause uses a host variable that is a simple Java variable named empname:

```
String empname ;
#sql {SELECT LASTNAME INTO :empname FROM EMP WHERE EMPNO= ‘000010’};
```

Installing and configuring DB2 SQLJ

In DB2 Version 610, support for SQLJ is integrated in the base FMID of JDBC (JDB6612).

In DB2 Version 510, this support is provided by APARs PQ19814 and PQ18939 (PTFs UQ22819) and needs FMID JDB5512.

SQLJ programming structure

The general structure of a SQLJ program is very similar to a JDBC program except that the coding of a SQLJ program is simpler because certain steps are automatically generated by SQLJ clauses.

A typical SQLJ application should execute the following steps in sequence:

- Import JDBC and SQLJ packages.
- Load the SQLJ driver.
- Identify the target DB2 subsystem.
- Connect to the DB2 subsystem.
- Create a SQL statement.
- Create a result set.
- Execute the SQL query using a result set iterator.
- Display the result set.
- Disconnect from the DB2 subsystem.

The following sections will describe how you should write Java and SQLJ statements to implement these different phases.

Importing JDBC and SQLJ packages

To import the Java packages for SQLJ and JDBC, you must include these lines at the top of your application program:

```
import java.sql.*;              /* JDBC support               */
import sqlj.runtime.*;          /* SQLJ runtime support       */
```

Generating a connection context class

A connection context is a SQLJ concept that identifies a DB2 subsystem. First, you should use the sqlj clause ‘context’ to generate a connection context class:

```
#sql context SQLJ_context;    // Generate the SQLJ_context class
```

This class will be used to instantiate the ‘connection’ that will be used in the SQLJ program:

```
SQLJ_context my_context;
```

Loading the SQLJ driver

To load the DB2 for OS/390 SQLJ JDBC driver and register it with the DriverManager, you must invoke the ‘Class.forName()’ method with ‘COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver’ as an argument:

```
Class.forName("COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver");
```

Identifying the target DB2 subsystem

This step is similar to the corresponding JDBC step.

The Java application must identify the DB2 subsystem it wants to connect by passing a URL to the ‘DriverManager’.

The basic structure of the URL for a DB2 for OS/390 subsystem is:

```
jdbc:db2os390sqlj:<location-name>
```

where ‘location-name’ is the DB2 location name specified in the DDF (Distributed Data Facility) parameter:

```
String url = "jdbc:db2os390sqlj:DB2SDRDA";
```

Connecting to the DB2 subsystem

The application programmer should invoke the JDBC ‘Java.sql.DriverManager.getConnection()’ method to create a connection to the DB2 subsystem.

The argument for ‘Java.sql.DriverManager.getConnection()’ is the

URL describing the target DB2 subsystem:

```
Connection conn;
conn = DriverManager.getConnection(url);
```

Then the connection context instance is initialized using the constructor for the connection context class. The argument of the constructor is the JDBC connection resulting from the 'Java.sql.DriverManager. getConnection()' method:

```
my_context = new SQLJ_context(conn);
```

Executing a SQL query with 'named' result set iterator and Java variable

In DB2 application programs that are written in traditional host languages, you use a cursor to retrieve individual rows from the result table that is generated by a SELECT statement.

The SQLJ equivalent of a cursor is a result set iterator. A result set iterator is a Java object that you use to retrieve rows from a result table.

You define a result set iterator using an iterator declaration clause, which specifies a list of Java data types. Those data type declarations represent columns in the result table and are referred to as columns of the result set iterator:

```
#sql iterator SQLJ_iterator(String NAME, String CREATOR);
```

Then, you can use the result set iterator to execute the SQL query:

```
#sql [my_context] my_iterator =
   { SELECT NAME, CREATOR FROM SYSIBM.SYSDATABASE
                          WHERE CREATOR = :parm_creator } ;
```

Displaying the result set

When SQLJ encounters a named iterator declaration, it generates a named iterator class with the same name that you use in the iterator declaration clause.

In the named iterator class, SQLJ generates an accessor method for each column name in the iterator declaration clause.

The accessor method name is the same name as the column name in the iterator declaration clause.

You can use the accessor method to retrieve data from the corresponding column of the result set.

The ‘next()’ method skips to the next record of the result set and returns ‘false’ if the current record is the last record of the result set:

```
while(my_iterator.next())
  {
   System.out.println(my_iterator.NAME()+" "+my_iterator.CREATOR());
  }
```

Disconnecting from the DB2 subsystem

The last step is to disconnect from the DB2 subsystem. It is done using the ‘close()’ method, which closes the DB2 connection and frees all resources associated with this connection:

```
con.close();
```

Handling SQL errors and warnings

SQLJ SQL errors and warnings management is similar to the JDBC exceptions management. You should refer to this paragraph.

Preparing and running SQLJ programs

Procedure to compile a SQLJ program

After you write a SQLJ application, you must generate an executable form of the application. This procedure involves several steps:

- Translating the source code to produce modified Java source code and serialized profiles.
- Compiling the generated Java source program.
- Customizing the serialized profiles to produce DBRMs.
- Binding the DBRMs into packages and binding the packages into a plan, or binding the DBRMs directly into a plan.

Translating SQLJ source code

The first step in preparing an executable SQLJ program is to use the

SQLJ translator to generate a Java source program and one or more serialized profiles.

In order to translate a SQLJ program, you should create a script like export_compile.sh, which exports the required DB2 library in the CLASSPATH variable. You also need to modify the PATH variable to be able to access the SQLJ translator:

```
BROWSE
 -- /u/i990557/java/db2/sqlj/export_compile.sh Line 00000000 Col 001 067
 Command ===>                                           Scroll ===> HALF
***************************** Top of Data ******************************
export CLASSPATH=/usr/lpp/db2/classes/db2sqljclasses.zip:$CLASSPATH
export CLASSPATH=.:$CLASSPATH
export PATH=/usr/lpp/db2/bin:$PATH
*************************** Bottom of Data *****************************
```

Then, you should run this script to modify your environment variables:

```
I990557:/u/i990557/java/db2/jdbc: >. export_compile.sh
```

At this point, you are able to invoke ‘sqkj’, to translate your SQLJ source program (with a ‘.sqlj’ extension):

```
I990557:/u/i990557/java/db2/sqlj: >ls
export_DBRMLIB.sh  export_compile.sh  export_run.sh      sample02.sqlj
I990557:/u/i990557/java/db2/sqlj: >sqlj sample02.sqlj
I990557:/u/i990557/java/db2/sqlj: >ls
export_DBRMLIB.sh        export_run.sh            sample02.sqlj
export_compile.sh        sample02.java
sample02_SJProfile0.ser
I990557:/u/i990557/java/db2/sqlj: >
```

This step produces two files:

- The generated Java source program – program_name.java.
- A serialized profile file – program_name_SJProfile0.ser.

Compiling the generated Java source program

After the translation of the SQLJ source file, you should use the javac command to compile the generated Java source file.

You can compile the generated Java source file as you would compile any other Java program:

```
I990557:/u/i990557/java/db2/sqlj: >javac sample02.java
I990557:/u/i990557/java/db2/sqlj: >
```

Customizing the serialized profile

After you use the SQLJ translator to generate serialized profiles for a SQLJ program, customize the serialized profile to produce a standard DB2 for OS/390 DBRM and a serialized profile that is customized for DB2 for OS/390.

In order to store the DBRM in an MVS partitioned dataset, you should first define the new environment variable DB2SQLJDBRMLIB with the export command :

```
I990557:/u/i990557/java/db2/sqlj: >export
DB2SQLJDBRMLIB=SMAINT.I990557.JAVA.SQLJ.DBRMLIB
```

In order to customize the serialized profile, you should create a script like export_customize.sh, which exports the required environment variables:

```
BROWSE
 -- /u/i990557/java/db2/sqlj/export_customize. Line 00000000 Col 001 054
 Command ===>                                            Scroll ===> HALF
***************************** Top of Data ******************************
export LIBPATH=/usr/lpp/db2/lib:$LIBPATH
export LD_LIBRARY_PATH=/usr/lpp/db2/lib
export PATH=/usr/lpp/db2/lib:$PATH
export DB2SQLJDBRMLIB=SMAINT.I990557.JAVA.SQLJ.DBRMLIB
**************************** Bottom of Data ****************************
```

To customize a serialized profile, execute the db2profc command:

```
I990557:/u/i990557/java/db2/sqlj: >. export_customize.sh
I990557:/u/i990557/java/db2/sqlj: >db2profc -pgmname=sample02
sample02_SJProfile0.ser
Serialized Profile sample02_SJProfile0.ser has been customized for DB2
for OS/390.Bind.
```

Binding a plan for the SQLJ program

After you have customized the serialized profiles for your SQLJ application program, you must bind the DBRMs that are produced by the SQLJ customizer. You can bind the DBRMs directly into a plan or bind the DBRMs into packages and then bind the packages into a plan.

```
 DSN SYSTEM(DB2S)
 RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP2)
 BIND PLAN(SQLJPLAN) MEMBER(SAMPLE02)    -
                     ACTION(REP)         -
                     VALIDATE(BIND)      -
```

```
                                    ISOLATION(CS)         -
                                    ACQUIRE(USE)          -
                                    RELEASE(COMMIT)       -
                                    EXPLAIN(NO)
```

Procedure to run a SQLJ program

In order to run the SQLJ program, you need to set up three other environment variables:

- DB2SQLJPLANNAME, which specifies the name of the plan associated with the SQLJ application.
- DB2SQLJSSID, which specifies the name of the DB2 subsystem.
- DB2SQLJATTACHTYPE, which specifies the type of attachment facility to use to connect to the DB2 subsystem (CAF or RRSAF).

This can be done using the following script:

```
BROWSE-/u/i990557/java/db2/sqlj/export_run.sh -Line 00000000 Col 001 067
 Command ===>                                              Scroll ===> HALF
***************************** Top of Data ******************************
export DB2SQLJPLANNAME=SQLJPLAN
export DB2SQLJSSID=DB2S
export DB2SQLJATTACHTYPE=CAF
**************************** Bottom of Data ****************************
```

At this point, you are able to execute the SQLJ program:

```
I990557:/u/i990557/java/db2/sqlj: >java sample02
```

SAMPLES

To illustrate the previous concepts, you will find in the following sections a JDBC and a SQLJ sample application.

The programming logic of these applications is the same:

- They receive as an argument a string that represents a CREATOR name.
- They query the SYSIBM.SYSDATABASE catalog table to retrieve the databases referenced by this CREATOR.

These Java applications are written using two Java programs: a main class that receives the CREATOR argument and then instantiates a specialized class which manages the DB2 connection, and the SQL query.

JDBC

JDBC_Query_Main.Java

```
/**************************/
/* JDBC_Query_Main class  */
/**************************/
import  java.sql.*;           // import JDBC package
class JDBC_Query_Main
       {
        public static void main(String argv[])
          {
           JDBC_Query jdbc_query = new JDBC_Query();
           String parm_creator="SYSIBM";
           System.out.println("*** Running JDBC_Query_Main.java ***");
                     // Retrieve input parameter
           if (argv.length > 0)
                 {
                  parm_creator = argv[0];
     System.out.println("*** Query for CREATOR = "+parm_creator+" ***");
                  try
                    {
                     jdbc_query.open_connection();
                     jdbc_query.process_query(parm_creator);
                     jdbc_query.close_connection();
                    }
                  catch(Exception e)
                    {
                    /* to avoid additional error messages when an error
                    occurs during open_connection                     */
                    }
                 }
                else
                 {
               System.out.println("*** Input parameter is missing ***");
                 }
          }
       }
```

JDBC_Query.Java

```
/********************/
/* JDBC_Query class */
/********************/
```

```
import  java.sql.*;           // import JDBC package
class JDBC_Query
       {
        Connection conn;
        /*************************/
        /* open DB2 connection   */
        /*************************/
        public void open_connection()
          {
           try
             {
                     /*==========================================*/
                     /* The forName method loads the JDBC driver */
                     /*==========================================*/
              Class.forName("ibm.sql.DB2Driver");
                    // URL passed to the DriverManager
              String url = "jdbc:db2os390:DB2SDRDA";
                     /*===============================================*/
                     /* getConnection() creates a connection instance */
                     /* to the db2 subsystem                          */
                     /*===============================================*/
              conn = DriverManager.getConnection(url);
              System.out.println("*** Connected to DB2 subsystem ***");
             }
           catch(SQLException e)
             {
              manage_SQLException(e);
             }
           catch(ClassNotFoundException e)
             {
              manage_ClassNotFoundException(e);
             }
          }
        /*************************/
        /* close DB2 connection  */
        /*************************/
        public void close_connection()
          {
           try
             {
                     /*========================================*/
                     /* close() closes the connection instance */
                     /*========================================*/
              conn.close();
             }
           catch(SQLException e)
             {
              manage_SQLException(e);
             }
          }
        /************************/
        /* process SQL query    */
        /************************/
        public void process_query(String parm_creator)
          {
           try
             {
                     /*==============================================*/
                     /* createStatement() method creates a statement */
                     /* instance                                     */
                     /*==============================================*/
              Statement stmt = conn.createStatement();
                     /*==============================================*/
                     /* SQL query which will be sent to the DB2      */
                     /* subsystem                                    */
                     /*==============================================*/
              String query =
                "SELECT NAME, CREATOR FROM SYSIBM.SYSDATABASE where
creator = '"+parm_creator+"'";
              System.out.println("*** Creating result set ***");
                     /*===============================================*/
                     /* Execute the query and store the result in the */
                     /* result set                                    */
                     /*===============================================*/
              ResultSet rs = stmt.executeQuery(query);
                     /*===============================================*/
                     /* Enter in a loop to display the rows in the */
                     /* result set                                    */
                     /*===============================================*/
              boolean next = rs.next();
              if (next)                  // any row to display ?
                {
                   while(next)
                      {
                       String dbname  = rs.getString(1);
                       String creator = rs.getString(2);
                       System.out.println(dbname+" "+creator);
                       next = rs.next();
                      }
                }
              else
                {
System.out.println("===> There is no row for CREATOR =
"+parm_creator+" ***");
                }
                     /*===============================================*/
                     /* Closes the result set and the statement       */
                     /*===============================================*/
              System.out.println("*** Closing result set ***");
              rs.close();
```

```
         System.out.println("*** Closing statement ***");
         stmt.close();
        }
      catch(SQLException e)
        {
         manage_SQLException(e);
        }
     }
   /*************************/
   /* Exceptions Management */
   /*************************/
   public void manage_SQLWarning(SQLWarning w)
     {
      while(w |= null)
       {
         System.out.println("===> SQLWarning detected :");
         System.out.println("===> SQLState: "+ w.getSQLState());
         System.out.println("===> Error Code: "+ w.getErrorCode());
         System.out.println("===> Message: "+ w.getMessage());
         w = w.getNextWarning();
       }
     }
   public void manage_SQLException(SQLException e)
     {
      while(e |= null)
       {
         System.out.println("===> SQLException detected :");
         System.out.println("===> SQLState: "+ e.getSQLState());
         System.out.println("===> Error Code: "+ e.getErrorCode());
         System.out.println("===> Message: "+ e.getMessage());
         e = e.getNextException();
       }
     }
public void manage_ClassNotFoundException(ClassNotFoundException e)
     {
      System.out.println("===> ClassNotFoundException:");
      e.printStackTrace();
     }
  }
```

Preparing and running the JDBC application

Compiling the application

Use the following script to compile the JDBC application:

```
BROWSE -- JDBC_Query_compile.sh               Line ØØØØØØØØ Col ØØ1 Ø68
 Command ===>                                          Scroll ===> PAGE
***************************** Top of Data ******************************
```

```
export  CLASSPATH=/usr/lpp/db2/classes/db2jdbcclasses.zip:$CLASSPATH
export  CLASSPATH=.:$CLASSPATH

echo 'Compiling JDBC_Query.java ...'
javac JDBC_Query.java
echo 'Compiling JDBC_Query_Main.java ...'
javac JDBC_Query_Main.java
**************************** Bottom of Data ****************************
```

Running the application

In order to run the JDBC application, you can use the following script:

```
BROWSE
- /u/i990557/java/db2/jdbc/JDBC_Query_run.sh Line 00000000 Col 001 052
 Command ===>                                            Scroll ===> PAGE
***************************** Top of Data ******************************
export  LIBPATH=/usr/lpp/db2/lib:$LIBPATH
export  DSNAOINI="/u/i990557/java/db2/jdbc/DSNAOINI"

java JDBC_Query_Main SYSADM
**************************** Bottom of Data ****************************
```

You must also set up the CLI configuration file DSNAOINI:

```
BROWSE -- /u/i990557/java/db2/jdbc/DSNAOINI -- Line 00000000 Col 001 027
 Command ===>                                           Scroll ===> PAGE
***************************** Top of Data ******************************
[COMMON]
MVSDEFAULTSSID=DB2S
[DB2S]
MVSATTACHTYPE=CAF
PLANNAME=DSNACLI
**************************** Bottom of Data ****************************
```

You get the following type of result:

```
I990557:/u/i990557/java/db2/jdbc: >. JDBC_Query_run.sh
*** Running JDBC_Query_Main.java ***
*** Query for CREATOR = SYSADM ***
*** Connected to DB2 subsystem ***
*** Creating result set ***
DSNCV    SYSADM
DSNDB07  SYSADM
DSNDDF   SYSADM
DSNRGFDB SYSADM
DSNRLST  SYSADM
PPREDB04 SYSADM
*** Closing result set ***
*** Closing statement ***
I990557:/u/i990557/java/db2/jdbc: >
```

SQLJ

SQLJ_Query_Main.Java

```
/**************************/
/* JDBC_Query_Main class  */
/**************************/
class SQLJ_Query_Main
       {
        public static void main(String argv[])
          {
           SQLJ_Query sqlj_query = new SQLJ_Query();
           String parm_creator="SYSIBM";
           System.out.println("*** Running SQLJ_Query_Main.java ***");
                     // Retreive input parameter
           if (argv.length > Ø)
                 {
                  parm_creator = argv[Ø];
     System.out.println("*** Query for CREATOR = "+parm_creator+" ***");
                  try
                    {
                     sqlj_query.open_connection();
                     sqlj_query.process_query(parm_creator);
                     sqlj_query.close_connection();
                    }
                  catch(Exception e)
                    {
                     /* to avoid additional error messages when an error
                        occurs during open_connection                   */
                    }
                 }
                else
                 {
               System.out.println("*** Input parameter is missing ***");
                 }
          }
       }
```

SQLJ_Query.sqlj

```
/******************/
/* SQLJ_Query.sqlj */
/******************/
import java.sql.*;              /* JDBC support              */
import sqlj.runtime.*;          /* SQLJ runtime support      */
            /**************************************/
            /* Generate connection context class    */
            /**************************************/
#sql context SQLJ_context;    // Generate the SQLJ_context class
```

```
            /****************************************/
            /* Generate result set iterator class   */
            /****************************************/
#sql iterator SQLJ_iterator(String NAME, String CREATOR);
public class SQLJ_Query
  {
        Connection conn;
        SQLJ_context my_context;
    /*************************/
    /* open DB2 connection   */
    /*************************/
    public void open_connection()
      {
        try
           {
                              /******************************/
                              /* load SQLJ JDBC driver      */
                              /******************************/
             Class.forName("COM.ibm.db2os390.sqlj.jdbc.DB2SQLJDriver");
                   // URL passed to the DriverManager
             String url = "jdbc:db2os390sqlj:DB2SDRDA";
                    /*===============================================*/
                    /* getConnection() creates a connection instance */
                    /* to the db2 subsystem                          */
                    /*===============================================*/
             conn = DriverManager.getConnection(url);
             my_context = new SQLJ_context(conn);
           }
        catch(SQLException e)
           {
            manage_SQLException(e);
           }
        catch(ClassNotFoundException e)
          {
           manage_ClassNotFoundException(e);
          }
      }
    /*************************/
    /* close DB2 connection  */
    /*************************/
    public void close_connection()
      {
        try
           {
            conn.close();
           }
        catch(SQLException e)
           {
            manage_SQLException(e);
           }
      }
      /*************************/
      /* process SQL query     */
      /*************************/
      public void process_query(String parm_creator)
        {
         SQLJ_iterator my_iterator;
         try
           {
                    /*===============================================*/
                    /* Issue select                                  */
                    /*===============================================*/
             #sql [my_context] my_iterator = { SELECT NAME, CREATOR FROM
SYSIBM.SYSDATABASE WHERE CREATOR = :parm_creator } ;
             while(my_iterator.next())
               {
       System.out.println(my_iterator.NAME()+" "+my_iterator.CREATOR());
               }
           }
        catch(SQLException e)
           {
            manage_SQLException(e);
           }
        }
        /*************************/
        /* Exceptions Management */
        /*************************/
        public void manage_SQLWarning(SQLWarning w)
          {
           while(w |= null)
            {
              System.out.println("===> SQLWarning detected :");
              System.out.println("===> SQLState: "+ w.getSQLState());
              System.out.println("===> Error Code: "+ w.getErrorCode());
              System.out.println("===> Message: "+ w.getMessage());
              w = w.getNextWarning();
            }
          }
        public void manage_SQLException(SQLException e)
          {
           while(e |= null)
            {
              System.out.println("===> SQLException detected :");
              System.out.println("===> SQLState: "+ e.getSQLState());
              System.out.println("===> Error Code: "+ e.getErrorCode());
              System.out.println("===> Message: "+ e.getMessage());
              e = e.getNextException();
            }
          }
     public void manage_ClassNotFoundException(ClassNotFoundException e)
```

```
          {
           System.out.println("===> ClassNotFoundException:");
           e.printStackTrace();
          }
  }
```

Preparing and running the SQLJ application

Compiling the application

Use the following script to translate, compile, and customize the SQLJ application:

```
BROWSE -- SQLJ_Query_compile.sh                Line 00000000 Col 001 067
 Command ===>                                          Scroll ===> PAGE
***************************** Top of Data ******************************
export CLASSPATH=/usr/lpp/db2/classes/db2sqljclasses.zip:$CLASSPATH
export CLASSPATH=.:$CLASSPATH
export PATH=/usr/lpp/db2/bin:$PATH
export LIBPATH=/usr/lpp/db2/lib:$LIBPATH
export LD_LIBRARY_PATH=/usr/lpp/db2/lib
export PATH=/usr/lpp/db2/lib:$PATH
export DB2SQLJDBRMLIB=SMAINT.I990557.JAVA.SQLJ.DBRMLIB

echo 'Translating SQLJ_Query.sqlj ...'
sqlj SQLJ_Query.sqlj
echo 'Compiling SQLJ_Query.java ...'
javac SQLJ_Query.java
echo 'Compiling SQLJ_Query_Main.java ...'
javac SQLJ_Query_Main.java
echo 'Customizing serialized profile ...'
db2profc -pgmname=SQLJQUER SQLJ_Query_SJProfile0.ser
**************************** Bottom of Data ****************************
```

Binding the DB2 plan

You should bind the generated DBRM using:

```
//STEP002  EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//DBRMLIB  DD DISP=SHR,DSN=SMAINT.I990557.JAVA.SQLJ.DBRMLIB
//SYSTSIN  DD *
 DSN SYSTEM(DB2S)
 RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP2)
 BIND PACKAGE(SPKG02) MEMBER(SQLJQUER) ACT(REP) ISOLATION(CS)
 BIND PLAN(SQLJPLAN)  PKLIST(SPKG02.*)   ACT(REP)
/*
```

Running the application

To run the SQLJ application, you can use the following script:

```
BROWSE -- /u/i990557/java/db2/sqlj/SQLJ_Query_run.sh Line 00000000 Col
001 067
 Command ===>                                                Scroll ===> PAGE
***************************** Top of Data ******************************
export CLASSPATH=/usr/lpp/db2/classes/db2sqljclasses.zip:$CLASSPATH
export CLASSPATH=.:$CLASSPATH
export PATH=/usr/lpp/db2/bin:$PATH
export LIBPATH=/usr/lpp/db2/lib:$LIBPATH
export LD_LIBRARY_PATH=/usr/lpp/db2/lib
export DB2SQLJDBRMLIB=SMAINT.I990557.JAVA.SQLJ.DBRMLIB
export DB2SQLJPLANNAME=SQLJPLAN
export DB2SQLJSSID=DB2S
export DB2SQLJATTACHTYPE=CAF

java SQLJ_Query_Main SYSADM
**************************** Bottom of Data ****************************
```

You get the following type of result:

```
I990557:/u/i990557/java/db2/sqlj: >. SQLJ_Query_run.sh
*** Running SQLJ_Query_Main.java ***
*** Query for CREATOR = SYSADM ***
DSNCV    SYSADM
DSNDB07  SYSADM
DSNDDF   SYSADM
DSNRGFDB SYSADM
DSNRLST  SYSADM
PPREDB04 SYSADM
I990557:/u/i990557/java/db2/sqlj: >
```

Patrick Renard
CTRNE (France)

Many subscribers reading *DB2 Update* will have met similar problems and come up with quite different solutions. We’d like to hear what your alternative solution is. Contact the editor, Trevor Eddolls, at any of the addresses shown on page 2 for a copy of our *Notes for Contributors*.

Case study – NFU Mutual

NFU Mutual is a UK-based company that was set up in the early 1900s by seven Warwickshire farmers. The company has since grown into the UK's leading rural insurer, catering for the special requirements of the majority of farmers and growers, as well as many other people who live and/or work in the countryside.

NFU Mutual serves its policyholders through a network of 600 field agents, backed by specialist staff at regional and branch offices, as well as at its head office in Stratford-Upon-Avon.

With so many customers and such a variety of products to offer, NFU generates a considerable amount of data. The main policy production systems run on an IBM System/390 mainframe platform, with data held on a DB2 database running on both the mainframe and the agents' laptops. As new customers come on board, and policy and customer details change on an on-going basis, NFU needs to send monthly data updates to its agents. This used to involve NFU's IT team in downloading the data via dial-up lines to 600 agents across the country, which was taking a considerable amount of time.

Updating the agent's laptops could take up to three weeks out of every month,according to Mel Ward, a Senior Database Administrator at NFU Mutual. The dial-up connections could sometimes drop out, which would mean that the process had to be started again. Also, there was no way of checking that the data had loaded properly at the other end.

NFU Mutual decided to address this situation as part of an overall IT strategy revamp. In conjunction with an upgrade of its software systems to ensure year 2000 compliance, NFU decided to reduce the amount of code and the amount of data being sent down the line to agent's laptops in order to reduce download time. NFU put something in place in-house to reduce the amount of code. To address the issue of reducing the amount of data changes being sent to its field agents, the company looked at Grafton's Data Commander Audit', which captures changes made to DB2 data from the DB2 log, and Data

Commander Propagator, which enables NFU to define rules for mapping the changed data to the required agents' laptops. Propagator would also manage the transmission and subsequent application of changed data to the agents' databases.

Using Audit and Propagator provides NFU's IT team with an automated, end-to-end solution for sending only the changed data to the field agents, instead of the whole database. Combined with the reduction in code, this dramatically cut the amount of data being sent via dial-up connections to each laptop.

NFU was sending approximately 16MB of data to each agent every month. This has generally been reduced by over 90 percent to 62KB.

Data Commander Propagator also helped to reduce duplication of effort. Because NFU can define which sets of data are sent to which agent, they would need to send each agent only data changes on their own clients, rather than sending them changes to all clients.

Instead of taking up to three weeks out of every month to send the data changes, the whole process can be completed over a weekend.

The solution provides NFU with significant savings in support costs and time, increases functionality in getting the right data to the right people at the right time, and enables the company to propagate additional data.

An added benefit is the fact that the two products batch process, rather than using an on-line interactive approach, which means there is minimal impact on production DB2 systems and no changes needed to the DB2 applications. In addition, the software will not allow duplication of data, missed batches, or batches being processed out of sequence.

NFU found the software to be very straightforward to implement. For the Propagator software, the amount of implementation and training time depends on the complexity of the set-up, because rules need to be defined for the dissemination of the data from a DB2 view or table. It took about four days with NFU.

NFU initially installed the Audit software on the mainframe to monitor the volume of changed data being generated. Subsequently

Propagator was rolled out to the field agents in phases as the IT team received and configured new laptops for them.

Rowland Middleton
Cotec (UK)

CHANGELIMIT image copy information – part 1

The image copy utility in DB2 Version 5 provides helpful information with the CHANGELIMIT and REPORTONLY keywords.

Syntax:

```
    CHANGELIMIT
               (percent_value1)                  )   REPORTONLY
                               ,  percent_value2
```

Example:

```
    COPY TABLESPACE dbname.tsname CHANGELIMIT(10,20) REPORTONLY
```

The CHANGELIMIT specifies the percentage limit of changed pages in the tablespace when an incremental or full image copy should be taken. REPORTONLY specifies that image copy information is displayed. Image copies are not taken, only recommended. My example above recommends a full image copy if the percentage of changed pages is equal to or greater than 20 percent, and an incremental image copy if the percentage of changed pages is greater than 10 and less than 20 percent. If the percentage of changed pages is 10 percent or less, then no image copy is recommended.

My procedure provides the following main menu:

```
                           date: 29 Feb 2000
 Image Copy Information    time: 2:05pm
                           user: SYSADM
*******************************************

  1 - Image Copy Information
  2 - Image Copy Changelimit Strategy
  3 - Image Copy Changelimit Report
  X - Exit

*******************************************

==>   3                                PF3 - End
```

The options are:

1 Image copy information shows the status of tablespaces (age of the image copy or image copy not found).

2 Image copy changelimit strategy defines the lowlimit value (percent_value1) and highlimit value (percent_value2) for a tablespace(s).

3 Image copy changelimit report, with values (10,20), produces the report below:

```
---------------    Image Copy Changelimit Report   --- Row 1 to 10 of 10
Command ===>                                            Scroll ===>  CSR
------------------------------------------------------------------------
Snapshot date: 2000-02-29 time: 20.21.01                  >  5.50
------------------------------------------------------------------------
                             4KB      Empty    Changed    Percent of
DB.TS            Part      Pages      Pages      Pages    Changed
Ictype
DBTNADI.TSN149      A full image copy  must be taken       100.00      F
DBTNADI.TSN162    ALL         10          5          2      20.00      F
DBTNADI.TSN167    ALL         12          7          4      33.33      F
DBTNADI.TSN169      A full image copy  must be taken       100.00      F
DBTNADI.TSN170    ALL         36         19          2       5.55
NONE
DBTNADI.TSN174    ALL         36         21          2       5.55
NONE
DBTNADI.TSN177      A full image copy  must be taken       100.00      F
DBTNADI.TSN188    ALL         12          8          2      16.66      I
DBTNADI.TSN191    ALL         36          7          3       8.33
NONE
DBTNADI.TSN198    ALL         60          3          6      10.00
NONE
*************************** Bottom of data *****************************
```

ICM – REXX DRIVER PROCEDURE

```
/* Rexx */
/* ICM: Driver procedure                         */
/* trace r */
 zpfctl = 'OFF'
 address ispexec 'vput (zpfctl) profile'
 address ispexec 'addpop row(1) column(10)'
 top:
 date=date()
 time=time(c)
 address ispexec "display panel(icopyp)"
 do while rc=0
    select
      when(x='1') then do
         address ispexec rempop all
         "%ici"
         address ispexec 'addpop row(1) column(10)'
      end
      when(x='2') then do
         address ispexec rempop all
         "%iccs"
         address ispexec 'addpop row(1) column(10)'
      end
      when(x='3') then do
         address ispexec rempop all
         "%iclr"
         address ispexec 'addpop row(1) column(10)'
      end
      when(x='X') then do
         exit
      end
      otherwise rc=0
    end
    date=date()
    time=time(c)
    address ispexec "display panel(icopyp)"
 end
 exit
```

ICI – REXX PROCEDURE

```
/* REXX */
/* Image Copy Information   */
/* trace r */
 zpfctl = 'OFF'
 Y=MSG("OFF")
 /*****************************************************/
 /* Change to your convention standards              */
 prog    = 'PICOPY'
 prog1   = 'PICOPY1'
 plan    = 'PICOPY'
 plan1   = 'PICOPY1'
```

```
llib    = 'SKUPNI.BATCH.LOADLIB'
/******************************************************/
address ispexec 'vput (zpfctl) profile'
Call Aloc
cur='icdb'
TOP:
address ispexec "display panel(ICOPYPØ) cursor("CUR")"
Again:
if rc=8 then do
   Call Free_proc
   ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".DB2.ICOPY'"
   exit
end
/* Check input parameters                            */
if db2=' ' then do
   message = 'Enter db2 ssid. |'
   Call Error 'db2'
end
if icdb=' ' then icdb='%'
if icts=' ' then icts='%'
parm=substr(icdb,1,8)||substr(icts,1,8)
ADDRESS TSO
QUEUE "RUN PROGRAM("prog") PLAN("plan"),
       LIBRARY ('SKUPNI.BATCH.LOADLIB'),
       PARMS ('/"parm"')"
QUEUE "END "
"DSN SYSTEM("db2")"
if rc=12 then do
   "delstack"
   Call Free_proc
   Call Aloc
   Call Init
   message = 'Error.   'db2||' ssid is not valid.'
   Call Error 'db2'
end
"EXECIO * DISKR SYSPRINT (STEM ROW."
if row.2 = 'NO CATALOG ENTRIES FOUND' then do
   Call Free_proc
   Call Aloc
   Call Init
   message = 'No catalog entries found, check Search Fields.'
   Call Error 'icdb'
end
else do
   address ispexec,
    'tbcreate "iclist" names(dbase tspace part tstamp sta)'
   do i=2 to row.Ø BY 1
      if substr(word(row.i,1),1,1)='1'
      then dbase = substr(word(row.i,1),2)
      else dbase = word(row.i,1)
      tspace= word(row.i,2)
      part  = word(row.i,3)
      tstamp= word(row.i,4)
      if tstamp='-'
      then sta =subword(row.i,5)
      else sta = substr(row.i,5Ø)
      address ispexec 'tbadd "iclist"'
   end
   address ispexec 'tbtop "iclist"';
   Call Display_Icopy
end
DIS:
Select
  when(cmd='L') THEN DO
   Call List_of_tables
   cmd=''
   Call Display_Icopy
   Signal DIS
  end
  otherwise cmd=''
End
Call Free_proc
address ispexec 'tbend "iclist"'
Call Aloc
Signal Again
Aloc:
  ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".DB2.ICOPY'"
  "ALLOC DD(SYSPRINT) DSN('"SYSVAR(SYSUID)".DB2.ICOPY') SPACE(24 8),
  TRACK MOD UNIT(339Ø) RECFM(F,B) LRECL(8Ø) BLKSIZE(8Ø ,
  F(SYSPRINT) CATALOG REUSE "
Return
Error:
  ARG cur_par
  cur=cur_par
  address ispexec "setmsg msg(iciØØ1)"
  signal top
Return
Free_proc:
  "execio Ø diskr sysprint (finis"
  address tso "free f(sysprint)"
Return
Display_Icopy:
  address ispexec 'tbdispl "iclist" panel(ICOPYPØ)'
  if rc=8 then do
     Call Free_proc
     address ispexec 'tbend "iclist"'
     Exit
  end
Return
List_of_tables:
  Call Free_proc
  ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".DB2.LIST'"
  "ALLOC DD(SYSPRINT) DSN('"SYSVAR(SYSUID)".DB2.LIST'),
  SPACE(8 8) TRACK MOD UNIT(339Ø) RECFM(F,B) LRECL(8Ø),
  BLKSIZE(8Ø) F(SYSPRINT) CATALOG REUSE "
  parm=substr(dbase,1,8)||substr(tspace,1,18)
  ADDRESS TSO
  QUEUE "RUN PROGRAM("prog1") PLAN("plan1"),
         LIBRARY ('SKUPNI.BATCH.LOADLIB'),
         PARMS ('/"parm"')"
  QUEUE "END "
  "DSN SYSTEM("db2")"
  "EXECIO * DISKR SYSPRINT (STEM tb."
  address ispexec 'tbcreate "tlist",
          names(tbname creator card stime)'
  dbts=dbase||'.'||tspace
  do i=2 to tb.Ø
     tbname  = word(tb.i,1)
     creator = word(tb.i,2)
     card    = right(word(tb.i,3),1Ø)
     stime   = word(tb.i,4)
     address ispexec 'tbadd "tlist"'
  end
  address ispexec 'tbtop "tlist"';
  address ispexec 'tbdispl "tlist" panel(ICOPYP1)'
  address ispexec 'tbend "tlist"'
  Call Free_proc
  ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".DB2.LIST'"
Return
Init:
   dbase = ' '
   tspace= ' '
   part  = ' '
   tstamp= ' '
   sta   = ' '
Return
```

ICLR – REXX PROCEDURE

```
/* REXX */
/* ICLR: Image Copy Changelimit Report                        */
/* trace r */
 file= SYSVAR(SYSUID)||'.ICCS.DATA'
 dsn = sysdsn("'"file"'")
 Call Check
 file= SYSVAR(SYSUID)||'.TEMP.DATA'
 dsn = sysdsn("'"file"'")
 Call Check
 zpfctl = 'OFF'
```

```
Y=MSG("OFF")
pct=Ø
address ispexec 'vput (zpfctl) profile'
TOP:
if datatype(pct,'N')¬=1 | pct > 99.99
then do
  address ispexec
  'tbcreate "crlist" names(dbts num kpag epag cpag ppag ict)'
  zedsmsg = "Error"
  zedlmsg = "Percent field must be numeric"
  if pct > 99.99
  then zedlmsg = "Percent field range is Ø.ØØ - 99.99"
  "setmsg msg(isrzØØ1)"
  'tbdispl "crlist" panel(icopyp2) cursor(pct)'
  if rc=8 then Exit
  'tbend "crlist"'
  signal top
end
pct=format(pct,5,2)
address tso
"ALLOC DA('"SYSVAR(SYSUID)".ICCS.DATA') F(DT) SHR REUSE"
"execio Ø diskr dt (open"
"execio 1 diskr dt (stem one."
hdate = word(one.1,2)
htime = word(one.1,3)
"ALLOC DA('"SYSVAR(SYSUID)".TEMP.DATA') F(IN) SHR REUSE"
"execio Ø diskr in (open"
"EXECIO * DISKR IN (STEM ROW."
address ispexec
'tbcreate "crlist" names(dbts num kpag epag cpag ppag ict)'
count=Ø
num=row.Ø
do i=1 to row.Ø
   if substr(row.i,2,8)='DSNUØ5ØI'
   then dbts = word(row.i,6)
   if word(row.i,1)='DSNU446I'
   then do
        num  = right(A,3)
        kpag = 'full image'
        epag = 'copy  must'
        cpag = 'be taken'
        ppag = right(1ØØ.ØØ,1Ø)
        ict  = center(F,12)
        address ispexec 'tbadd "crlist"'
   end
   if word(row.i,1)='DSNU44ØI'
   then do
     k=i+5
     do j=k to row.Ø while row.j ¬=' '
        dbts = word(row.j,1)||'.'||word(row.j,2)
```

```
         num  = word(row.j,3)
         kpag = right(word(row.j,4),1Ø)
         if words(row.j)=7
         then do
            epag = right(Ø,1Ø)
            cpag = right(word(row.j,5),1Ø)
            ppag = right(word(row.j,6),1Ø)
            ict  = center(word(row.j,7),12)
         end
         else do
            epag = right(word(row.j,5),1Ø)
            cpag = right(word(row.j,6),1Ø)
            ppag = right(word(row.j,7),1Ø)
            ict  = center(word(row.j,8),12)
         end
         if ppag >= pct then
         address ispexec 'tbadd "crlist"'
      end
      i=j
    end
 end
 address ispexec
 'tbtop "crlist"'
 'tbdispl "crlist" panel(icopyp2) cursor(pct)'
 if rc=8 then pct=' '
 'tbend "crlist"'
 address tso
 "execio Ø diskr in (finis"
 "execio Ø diskr dt (finis"
 "FREE F(IN)"
 "FREE F(DT)"
 if pct=' ' then nop
 else signal TOP
 Y=MSG("ON")
 Exit
 Check:
  if dsn ¬= 'OK'
  then do
   say 'Dataset '||file||' not found. Define first IC Strategy'
   Exit
  end
 Return
```

ICCS – REXX PROCEDURE

```
/* REXX */
/* ICCS: Image Copy Changelimit Strategy     */
/* trace r */
 zpfctl = 'OFF'
```

```
Y=MSG("OFF")
/*****************************************************/
/* Change to your convention standards               */
program = 'PICOPY2'
plan    = 'PICOPY2'
llib    = 'SKUPNI.BATCH.LOADLIB'
/*****************************************************/
address ispexec 'vput (zpfctl) profile'
Call Aloc
cur='crec'
TOP:
address ispexec "display panel(icopyp3) cursor("CUR")"
if rc=8 then do
   Call Free_proc
   exit
end
/* Check input parameters                            */
if tsnc=' ' & dbnc=' ' & crec=' ' & tabc=' 'then do
   message='At least one catalog search field must be entered.'
   Call Error 'crec'
end
if llim=' ' then llim=1
if hlim=' ' then hlim=1Ø
if verify(llim,'Ø123456789') >Ø | llim > 1ØØ then do
   message='The LLIM parameter must be an integer '||,
           'number between Ø and 1ØØ'
   Call Error 'llim'
end
if verify(hlim,'Ø123456789') >Ø | hlim > 1ØØ then do
   message='The HLIM parameter must be an integer '||,
           'number between Ø and 1ØØ'
   Call Error 'hlim'
end
parm=substr(crec,1,8)||substr(tabc,1,18)||,
     substr(tsnc,1,8)||substr(dbnc,1,8)
ADDRESS TSO
QUEUE "RUN PROGRAM("program") PLAN("plan"),
       LIBRARY ('"llib"'),
       PARMS ('/"parm"')"
QUEUE "END "
"DSN SYSTEM("db2")"
if rc=12 then do
   "delstack"
   Call Free_proc
   Call Aloc
   message = 'Error.   'db2||' ssid is not valid  |'
   Call Error 'db2'
end
"EXECIO * DISKR SYSPRINT (STEM ROW."
IF SUBSTR(ROW.1,2,7) = 'SQLCODE' THEN DO
```

```
   Call Free_proc
   Call Aloc
   if word(row.1,3) = 100
   then message = 'No catalog entries found, check Search Fields'
   else message = 'SQLCODE = '||word(row.1,3)
   Call Error 'crec'
end
else do
   address ispexec 'tbcreate "iclist" names(db ts)'
   do i=2 to row.0
      db = word(row.i,2)
      ts = word(row.i,3)
      address ispexec 'tbadd "iclist"'
   end
   address ispexec 'tbtop "iclist"'
   address ispexec 'tbdispl "iclist" panel(icopyp4)'
   if rc=8 then do
      Call Free_proc
      address ispexec 'tbend "iclist"'
      Call Aloc
      signal top
   end
end
Call Free_proc
/* JCL Skeleton DB2 Image copy Changelimit Information */
title = 'IMAGE COPY CHANGELIMIT INFORMATION'
date=date()
time=time(c)
user=userid()
tempfile=userid()||'.ICCS.INFO'
address tso
"delete '"tempfile"'"
"free dsname('"tempfile"')"
"free ddname(ispfile)"
"free attrlist(formfile)"
"attrib formfile blksize(800) lrecl(80) recfm(f b) dsorg(ps)"
"alloc ddname(ispfile) dsname('"tempfile"')",
       "new using (formfile) unit(3390) space(1 1) cylinders"
address ispexec
"ftopen"
"ftincl ICCIS"
"ftclose"
zedsmsg = "JCL shown"
zedlmsg = "JCL Changelimit shown"
"setmsg msg(isrz001)"
"edit dataset('"tempfile"')"
address ispexec 'tbend "iclist"'
exit
Aloc:
  ADDRESS TSO "DELETE '"SYSVAR(SYSUID)".ICCS.DATA'"
```

```
   "ALLOC DD(SYSPRINT) DSN('"SYSVAR(SYSUID)".ICCS.DATA') SPACE(24 8),
   TRACK MOD UNIT(3390) RECFM(F,B) LRECL(80) BLKSIZE(800) ,
   F(SYSPRINT) CATALOG REUSE "
 Return
 Error:
   ARG cur_par
   cur=cur_par
   address ispexec "setmsg msg(ici001)"
   signal top
 Return
 Free_proc:
   "execio 0 diskr sysprint (finis"
   address tso "free f(sysprint)"
 Return
```

ICOPYP – PANEL

```
)attr default(%+_)
   [ type (output) intens(low)  color(green) caps(off)
   # type (output) intens(low)  color(white) caps(off)
   ] type (text)   intens(low)  color(white) caps(off)  hilite(reverse)
   _ type (input)  intens(low)  color(yellow) caps(on)  hilite(blink)
   | type (output) intens(low)  color(green)  caps(off)
   + type (text)   intens(low)  color(green)
   / type (text)   intens(low)  color(turq)
   ~ type (text)   intens(high) color(turquoise)
   @ type (text)   intens(high) color(red)   caps(off)  hilite(reverse)
)body window(46,14) expand ($$)
+]                        + date:|date       +
+] Image Copy Information + time:|time      +
+]                        + user: &zuser
/ *******************************************
+
+  [row1                                    +
+  [row2                                    +
+  [row3                                    +
+  [row4                                    +
+
/ *******************************************
+
+@==>+ _x+  #msg                            +
+                                  ] PF3 End +
)init
  &row1= '1 - Image Copy Information'
  &row2= '2 - Image Copy Changelimit Strategy'
  &row3= '3 - Image Copy Changelimit Report'
  &row4= 'X - Exit'
  if (&x = 1,2,3,x)
      &msg = ''
  else
      .attr (msg) = 'color (red)'
      &msg = 'Enter 1, 2, 3 or X.'
  if (&x = 1)
      .attr (row1) = 'color (yellow) caps(on)'
  if (&x = 2)
      .attr (row2) = 'color (yellow) caps(on)'
  if (&x = 3)
      .attr (row3) = 'color (yellow) caps(on)'
)proc
  if (.pfkey = pfØ3) &pf3 = exit
)end
```

ICOPYP0 – PANEL

```
)Attr Default(%+_)
   | type(text)   intens(high) caps(on ) color(yellow)
   ? type(text)   intens(high) caps(on ) color(green) hilite(reverse)
   # type(text)   intens(high) caps(off) hilite(reverse)
   [ type( input) intens(high) caps(on ) just(left )
   ] type( input) intens(high) caps(on ) just(left ) pad('''')
   ¬ type(output) intens(low ) caps(off) just(asis ) color(turquoise)
   } type(output) intens(low ) caps(off) just(asis ) color(yellow)
)Body  Expand(//)
%-/-/- ? Image Copy Information +%-/-/-
%Command ===>_zcmd                                      / /%Scroll
===>_amt +
+SSID[db2 + Database:[icdb    + Tablespace:[icts    +
+-----------------------------------------------------------------------
--------
+Valid cmd:|L+List of tables
+-----------------------------------------------------------------------
--------
#cmd#Database#Tablespace#Part#Timestamp          #Status of Image Copy
+
)Model
 ]z+¬z       ¬z         ¬z   ¬z                  }z
+
)Init
  .ZVARS = '(cmd dbase tspace part tstamp sta)'
  &amt = CSR
)Reinit
)Proc
  VPUT (db2 icdb icts ) PROFILE
)End
```

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

Bernard Zver (Slovenia) © Xephon 2000

DB2 news

IBM is to include Landmark Systems monitoring products in its SystemPac for OS/390. These include The Monitor for CICS (CICS/ESA), The Monitor for DB2, The Monitor for DBControl, The Monitor for MQSeries, The Monitor for MVS, and The Monitor for VTAM, NaviPlex, and NaviGraph.

For further information contact:
Landmark Systems, 8000 Towers Crescent Drive, Vienna, VA 22180-2700, USA.
Tel: (703) 902 8000.
URL: http://www.landmark.com/products/monitorfordb2.htm.

* * *

IBM has announced WebSphere Application Server Version 3.0, Advanced Edition for Linux, expanding its support for Java applications and Enterprise JavaBeans components to support Red Hat Linux 6.2. It uses DB2 for Linux, which is shipped with the product and used for container-managed persistent storage. Java Development Kit 1.1.8 for Linux is used as a Java Virtual Machine base within the run-time environment.

All functionality found in WebSphere Version 3.0 on other current platforms are available for Linux and there's support for Lotus Domino for Linux as well as DB2.

For further information contact your local IBM representative.
URL: http://www.ibm.com.

Merant has announced an agreement with IBM to integrate its SequeLink 5.0 middleware into WebSphere.

SequeLink will enhance IBM's all-embracing e-business development framework by providing transactional Java database connectivity (JDBC) to DB2 UDB, Oracle, SQL Server, Informix, and Sybase data sources.

For further information contact:
Merant, The Lawn, Old Bath Road, Newbury, Berks, RG14 1QN, UK.
Tel: (01635) 32646.
URL: http://www.merant.com.

* * *

IBM has announced Version 2 of its DB2 Administration Tool, which provides modification of tables and their attributes and the facilities to copy data and objects to other DB2 subsystems.

Enhancements include better sort and search capabilities and support for installation-defined line commands.

The tool provides panels and options to see new catalog information and it works on a data-sharing-group member level. It works on any hardware supporting DB2 Version 3.1 or later.

For further information contact your local IBM representative.
URL: http://www.ibm.com.