146

DB2 update

December 2004

In this issue

3 DB2 Version 8 – partitioning index

11 Trimming the installation image for simplified mass deployment of DB2 UDB Version 8.2 for Windows

18 DB2 attachment primer: part 2 – using DB2 attachments

24 Implementing image extender to retrieve a signature database

44 Project Cinnamon

46 DB2 news

A xephon PUBLICATION

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Bob Thomas
E-mail: info@xephon.com

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs $380.00 in the USA and Canada; £255.00 in the UK; £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 2000 issue, are available separately to subscribers for $33.75 (£22.50) 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.

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) 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.

Printed in England.

DB2 Version 8 – partitioning index

In DB2 Version 8, various improvements related to partitioning – especially related to indexes – have been made. This article looks at these enhancements and how they can be used for better application design.

TABLE-CONTROLLED PARTITIONING

In DB2 V8, the partitioning index can be of two types:

1 Index-controlled partitioning (the same as was available in DB2 V7).

2 Table-controlled partitioning (a new style of partitioning introduced in DB2 V8).

Table-controlled partitioning implies that the table definition itself controls the method of partitioning. This is done by using the PARTITION BY clause in the CREATE TABLE statement, as shown below:

```
CREATE TABLE CUSTOMER(ACCOUNT_NUM, Integer)……
          PARTITION BY (ACCOUNT_NUM, ASC)
```

PARTITION MANAGEMENT

DB2 V8 provides the ability to immediately add partitions, rotate partitions, and change the partitioning key values for table-controlled partitioned tables via the ALTER TABLE statement. It also allows the rebalancing of partitions.

Add partition

In previous versions of DB2, to add a partition necessitated dropping the entire tablespace, redefining it with additional partitions, and reloading the data. To avoid this situation, the maximum number of partitions was allocated at the start (most of them having the minimum of space).

With DB2 V8, partitions can be added immediately, simply by using the ADD PARTITION keyword in the ALTER TABLE command. None of the objects needs to be stopped before issuing the ALTER command.

The partition number is not supplied on the ALTER TABLE ADD PARTITION statement; it is selected by DB2, based on the current number of partitions of the table.

The table is quiesced and all related plans, packages, and cached statements are invalidated, necessitating a rebind. This is required because the access path can be optimized for reading certain partitions only.

This allows users to start with a limited number of partitions, as dictated by the current need, and add more on a periodical basis – resulting in fewer objects to be managed.

Rotate partition

Most customers use the partitioning scheme to store data by time range. DB2 V8 now provides the ability to rotate and reuse partitions over time. This provides the option of using rolling partitions, when it is required, to store the data for only a certain period of time.

If it is required to store data pertaining to a year with each partition having data corresponding to a month, you can create a table with 13 partitions. By using the ALTER TABLE ROTATE PARTITION FIRST TO LAST statement, you can specify that the data corresponding to the first (oldest in this case) partition should be deleted and the partition reused for the new set of data.

This is an easy way of using the table space to continuously hold the last 12 months' data – without having to unload, delete, create, load cycle or retain the older data that is not required.

Note that recovery to a previous point-in-time is blocked after running the rotate statement. Because the data in the partition being rolled off is deleted, it may be advisable to do an unload before rotating the partition.

To speed up the delete process, consider doing a LOAD REPLACE with an empty data set before running the ALTER TABLE … ROTATE PARTITION statement.

INDEXES IN V8

In V7, a partitioning index controlled how a table was partitioned. Any index other than the partitioning index was referred to as a secondary index (on a partitioned table), non-partitioned index (NPI), non-partitioning index, or non-clustering index.

In V8, the improvements made to partitioning and indexes have given rise to a different index classification, which predominantly falls into the following three categories:

1 Partitioning and secondary index – based on whether or not the columns in the index correlate to the 'partitioning' columns of the table.

2 Partitioned and non-partitioned – based on whether or not an index is physically 'partitioned'.

3 Clustered and non-clustered – based on whether or not the index determines the clustering of the data.

Partitioning and secondary indexes

Any index that has the same left-most key column(s), in the same order, and using the same collating sequence as the columns that control partitioning on the table is referred to as the partitioning index.

Any index other than the partitioning index can be referred to as a secondary index.

Consider the following three index definitions on the CUSTOMER table:

```
CREATE . . . INDEX PARTIX1
ON CUSTOMER (ACCOUNT_NUM ASC)
PARTITIONED

CREATE . . . INDEX PARTIX2
```

```
ON CUSTOMER (ACCOUNT_NUM ASC, STATE_CD ASC )

CREATE . . . INDEX SECIX1
ON CUSTOMER (CUST_NUM ASC)
PARTITIONED
CLUSTERED
```

Both *PARTIX1* and *PARTIX2* are partitioning indexes because both have ACCOUNT_NUM ASC (based on which the table is partitioned) as the left-most key. A partitioning index can contain all the partitioning columns plus additional columns – as shown in the case of *PARTIX2*.

SECIX1 is a secondary or non-partitioning index.

In effect an index that correlates to the partitioning columns of a table is a partitioning index and one that doesn't is a secondary index.

Partitioned and non-partitioned indexes

A partitioned index has the keyword PARTITIONED specified in the CREATE INDEX statement that defines it. A partitioned index is made up of multiple physical partitions – as many as the number of data partitions. The index keys in each index partition correspond to the rows in the same data partition number – index partition 1 contains only keys for rows in data partition 1, index partition 2 contains only keys for those rows found in data partition 2, and so on.

An index in which the keyword PARTITIONED is not specified in the CREATE INDEX statement is a non-partitioned index.

Note that the partitioning index of an index-controlled partitioning is always partitioned (because the partitioning range is given in the index definition only).

In effect, an index that is physically partitioned is a partitioned index and one that is not physically partitioned is a non-partitioned index.

In the above examples, *PARTIX1* and *SECIX1* are partitioned indexes, whereas *PARTIX2* is non-partitioned.

Partitioned and non-partitioned partitioning indexes

A partitioning index can be either partitioned or non-partitioned. In the above example, *PARTIX1* is referred to as a partitioned partitioning index. *PARTIX2* is referred to as a non-partitioned partitioning index because the keyword PARTITIONED is not specified. This implies that although the index correlates to the table partitioning, it is not physically partitioned.

Clustering and non-clustering indexes

The index that determines the order in which the rows are stored in a table is a clustering index, and the index that does not do this is a non-clustering index.

With index controlled partitioning, the partitioning index must be the clustering index. But in the case of table controlled partitioning, any index (including a secondary index) may be the clustering index. The clustering index may be unique or non-unique.

While the partitioning columns determine the placement of rows in the proper partition, the clustering index controls the location of the row within the partition.

In the above example, *SECIX1* is a clustering index. While the table is partitioned based on the *ACCOUNT_NUM* range, the order of rows within each partition is based on the *CUST_NUM* value.

Only one index can be explicitly defined as a clustering index, and if no explicit clustering index is specified for a table, the first index created on the table acts as the implicit clustering index when ordering data rows.

DATA PARTITIONED SECONDARY INDEX

As mentioned earlier, a partitioned secondary index, also referred to as a Data Partitioned Secondary Index (DPSI), is new in V8. DPSI is an index that is partitioned with the same partitioning scheme as the table. In other words, partition *n* of the index refers only to data in partition *n* of the table.

DPSI is created by specifying the keyword PARTITIONED while defining the secondary index. DB2 doesn't allow you to create the DPSI as UNIQUE, mainly to avoid searching all the partitions to ensure that the key *is* unique.

A DPSI can be either clustering or non-clustering.

The use of DPSIs promotes partition independence, reduces lock contention, and improves index availability. DPSI also provides efficiency for utility processing, partition-level operations (such as dropping or rotating partitions), and recovery of indexes.

DPSI and availability

DPSI allows the breaking up of the index into small sections, just like the earlier feature of breaking up the data. If there is an issue on some part of the index (say a disk failure), the impact or damage is limited to a small subset of the rows pertaining to a specific partition. DPSI also promotes high data availability by facilitating efficient utility processing.

DPSI and efficient utility processing

With DPSI, partition-level operations can take place at the physical rather than the logical level. While deleting the data in a partition, the keys for the rows of that partition must also be deleted from the indexes. Each Non-Partitioning Index (NPI) must be scanned to delete these keys and each NPI is processed serially. DPSI provides an extremely efficient way to take the oldest index partition and throw out the data from that one partition, and then make that the new partition and reload it with the new results.

During REORG, BUILD2 phase processing is not required for DPSIs. Because the keys for a given data partition reside in a single DPSI partition, a simple substitution of the index partition newly built by REORG for the old partition is all that is needed.

With DPSI, there is no contention between LOAD PART jobs because there are no shared pages between partitions. If all

indexes on a table are partitioned, index page contention is eliminated.

DPSI also improves the efficiency of parallel LOAD PART jobs because each LOAD job inserts DPSI keys into a separate index structure, in key order.

DPSI also allows partition parallelism in utilities like REORG, CHECK INDEX, and REBUILD INDEX.

DPSIs also improve the recovery characteristics because they can be copied and recovered at the partition level.

DPSI and data sharing overhead reduction

Data sharing overhead-reducing strategies, such as affinity routing, benefit from the use of DPSIs. Because P-locking occurs at the physical partition level, affinity routing is effective for DPSIs. While designing batch suites, this advantage of data affinities can be better exploited.

DPSI and query performance

DPSI can have both a positive and a negative impact on Query performance – based on the data access pattern.

DPSIs allow for query parallelism and are likely to be picked by the optimizer for queries with predicates on partitioning columns plus predicates on the secondary index columns. The DB2 access path selection can now make use of all leading partitioning key columns when determining the qualified partitions, reducing the range of qualified partitions.

Performance of a query with predicates that reference only the columns of a data-partitioned secondary index is likely to degrade because DB2 must examine each partition of the index for values that satisfy the predicate, if index access is chosen as the access path.

With DPSI, performance of queries with predicates that also restrict the query to a single partition (by also referencing columns of the partitioning index) is likely to improve. If the

query also has predicates on the leading columns of the partitioning key, DB2 does not need to examine all partitions. Thus the query performance can be improved if the predicates are included that allow for pruning of unqualified partitions.

The removal from consideration of inapplicable partitions is known as page range screening, or limited partition scan, or partition pruning. A limited partition scan can be determined at bind time (if the column is compared with a constant) or at run time (if the column is compared with a host variable, parameter marker, or special register).

Applications can take advantage of a limited partition scan when a correlation exists between columns in the partitioning index and in the DPSI. For example, let us assume that your table is partitioned based on DATE, the DPSI is on column ORDERNO and you want to process all the records based on the range of ORDERNO. Assuming also that the ORDERNO and DATE are correlated and you would be able to identify the DATE range corresponding to this ORDERNO range (for example the ORDERNO has the first four digits as the year), then limiting the number of partitions that are to be scanned by also including the DATE column in your predicate will improve the performance of your query.

CONCLUSION

The partitioning and the index improvements, especially DPSI, allow for improved availability and parallelism. Also, now, the application designer has more options in terms of choosing secondary and clustering indexes for partitioned tables. DPSI provides benefits like simpler data and index maintenance, improved availability, and parallelism. Factors like potential negative impact on query performance, increased space requirement in catalogs, etc have to be taken into account when deciding on the appropriateness of DPSI's use for a specific application.

C Sasirekha
Tata Consultancy Services (India)

Trimming the installation image for simplified mass deployment of DB2 UDB Version 8.2 for Windows

DB2 Universal Database (DB2 UDB) installation has always been a customizable operation. However, before DB2 UDB V8.2, the installation image had to be a complete one to accommodate the selection of any valid subset of components. In other words, the image had to be a superset of all the possible components that you could select. This requirement made the distribution of some DB2 UDB products cumbersome and ineffective in some deployment environments. You might have considered it to be less than ideal if you were sure that you didn't want to install certain components, but these components were nevertheless inflating an already large image. Suppose, moreover, that you were responsible for a mass deployment of DB2 UDB; wouldn't it be great if you could actually reduce the size of the installation image by removing those components that you didn't want? What if your scenario includes a large number of remote users who connect over slower (perhaps even dial-up) connections? A smaller installation image means a lightweight deployment that can reduce network traffic, save on storage, and significantly reduce installation time.

THE DB2IPRUNE UTILITY

Well, as of DB2 UDB Version 8.2 for Windows, you can, in fact, trim an installation image. The db2iprune utility is now available. This nifty tool can be used to remove specific components from a DB2 Windows product installation image. The utility removes the cabinet (.cab) files that are associated with certain components (various features, languages, and so on). The result is a new smaller image that can be installed using the regular DB2 installation methods.

You can find the executable (db2iprune.exe) on the installation CD or mounted installation image (but not in the installed product), in the following directory:

```
\db2\Windows\utilities\db2iprune
```

This directory also contains a sample pruning response (.prn) file that you can modify and then use, along with the utility, to prune the installation image.

Let's consider an actual example. We will prune the installation image for DB2 UDB Workgroup Server Edition (WSE) V8.2 for Windows. Although we are pruning a Version 8.2 image, the utility can be used to prune any Version 8 DB2 UDB for Windows image. The first step is to modify the db2wse.prn file, which, as we have stated, can be found in the db2iprune subdirectory on the installation CD. The file contains an informational header, followed by a list of removable languages and product components or features. The sample .prn file that we are going to use for our example is shown below:

```
* Input file for use with the db2iprune utility
* ---------------------------------------------
*
* Comments are made by placing either a * or a # at the start of a line,
* or by placing ** or ## after the start of a line to comment out the
* rest of that line.
*
* To remove the cabinet (.cab) files for a language, as well as the
* documentation files in the image for this language, uncomment the
* equivalent LANG keyword. To remove the .cab files for a feature,
* uncomment the equivalent COMP keyword. The PROD keyword is required to
* identify the product and need not change.

* For descriptions of DB2 features, refer to the db2_features file in
* the db2windowssamples directory on the DB2 installation CD.
*=============================================================================
PROD                            = WORKGROUP_SERVER_EDITION
LANG                            = BR        ** Brazilian Portuguese
LANG                            = CN        ** Chinese, Simplified
LANG                            = CZ        ** Czech
LANG                            = DE        ** German
LANG                            = DK        ** Danish
LANG                            = FI        ** Finnish
LANG                            = FR        ** French
LANG                            = ES        ** Spanish
LANG                            = IT        ** Italian
LANG                            = JP        ** Japanese
LANG                            = KR        ** Korean
LANG                            = NO        ** Norwegian
LANG                            = PL        ** Polish
```

```
LANG                            = RU        ** Russian
LANG                            = SE        ** Swedish
LANG                            = TW        ** Chinese, Traditional
*COMP                            = SYSTEM_BIND_FILES
*COMP                            = MDAC
*COMP                            = ODBC_SUPPORT
*COMP                            = OLE_DB_SUPPORT
*COMP                            = JDBC_SUPPORT
*COMP                            = SQLJ_SUPPORT
*COMP                             = APPLICATION_DEVELOPMENT_TOOLS
*COMP                            = IBM_JRE
*COMP                            = IBM_JDK
*COMP                            = LDAP_EXPLOITATION
*COMP                            = CLIENT_TOOLS
COMP                           = DB2_WEB_TOOLS
COMP                           = DATA_WAREHOUSE_CENTER
COMP                           = INFORMATION_CATALOG_CENTER
COMP                           = INFORMATION_CATALOG_CENTER_WEB
COMP                           = SPATIAL_EXTENDER_CLIENT_SUPPORT
COMP                           = XML_EXTENDER
COMP                           = SATELLITE_SYNCHRONIZATION
COMP                           = REPLICATION_APPLY
COMP                           = REPLICATION_CAPTURE
COMP                           = INFORMIX_DATA_SOURCE_SUPPORT
*COMP                            = DATABASE_TOOLS
COMP                           = DATA_WAREHOUSE_SERVER
*COMP                             = TCPIP_DB2_CLIENT_SUPPORT
*COMP                             = TCPIP_DB2_LISTENER_SUPPORT
COMP                           = NETBIOS_DB2_CLIENT_SUPPORT
COMP                           = NETBIOS_DB2_LISTENER_SUPPORT
COMP                           = NPIPE_DB2_CLIENT_SUPPORT
COMP                           = NPIPE_DB2_LISTENER_SUPPORT
COMP                           = APPC_DB2_CLIENT_SUPPORT
COMP                           = APPC_DB2_LISTENER_SUPPORT
COMP                           = FIRST_STEPS
*COMP                             = CONFIGURATION_ASSISTANT
*COMP                            = COMMAND_CENTER
*COMP                            = CONTROL_CENTER
*COMP                            = ACTIVITY_MONITOR
*COMP                            = EVENT_ANALYZER
*COMP                            = DEVELOPMENT_CENTER
*COMP                            = DB2_SAMPLE_DATABASE
COMP                           = WAREHOUSE_SAMPLE_DATABASE
COMP                           = DB2_SAMPLE_APPLICATIONS
COMP                           = SQLJ_SAMPLES
COMP                           = WAREHOUSE_SAMPLES
COMP                            = INFORMATION_CATALOG_SAMPLES
COMP                            = SPATIAL_EXTENDER_SAMPLES
COMP                           = XML_EXTENDER_SAMPLES
```

The file contains three types of keyword:

- *PROD* identifies the product that is associated with this image. Do not comment out or otherwise modify this line.
- *LANG* identifies a specific language. The .prn file lists the full set of supported languages (other than English), and you can uncomment any language that you want the db2iprune utility to remove from the installation image. We are going to remove them all.
- *COMP* identifies a specific component. The .prn file lists the full set of components, and you can uncomment (remove the asterisk in front of) any component that you want the db2iprune utility to remove from the installation image. As you can see, we are going to remove a substantial number of components.

AN EXAMPLE

We are now ready to create a pruned installation image. For example, to prune the DB2 UDB WSE V8.2 for Windows installation image located on the E: drive using the modified copy of the db2wse.prn file (located in D:\WorkDir), and create the pruned image in the D:\compact_wse directory, issue the following command:

```
E:\db2\Windows\utilities\db2iprune>db2iprune -r D:\WorkDir\db2wse.prn
-p E:\ -o D:\compact_wse
```

The required parameters for this command are:

- *-r input-file* – the full path to the .prn file that specifies the languages and components that are to be removed from the installation image.
- *-p source-image* – the full path to the root directory of the source image (the directory that contains the setup.exe file).
- *-o new-image* – the full path to the directory in which the pruned image is to be created; if this directory does not already exist, the utility will create it.

Of course, we could have created an installation image on another drive first, and then pruned that image instead of the image on the CD. To do that, we could have used the cpysetup batch file, located in \db2\Windows\utilities, to create a source installation image in, for example, D:\wse:

```
E:\db2\Windows\utilities>cpysetup  D:\wse
Copying files...
1 File(s) copied
1800 File(s) copied
Successful completion.
```

If the target directory (D:\wse) did not already exist, cpysetup would have created it.

While the db2iprune utility is running, it writes status information to standard output. For example:

```
Please wait...  The product image is being copied to the destination
specified:
D:\compact_wse.
Deleting .cab files for the feature whose token is DB2_WEB_TOOLS...
Deleting .cab files for the feature whose token is
DATA_WAREHOUSE_CENTER...
...
Deleting .cab files for all features of language: RU...
Deleting .cab files for all features of language: SE...
Deleting .cab files for all features of language: TW...
```

The pruning operation takes a few minutes to complete, and longer if more components are being removed. The end result is an installation image whose structure mirrors the structure of the source image. The operation also generates a log file, db2iprune.log, which is written to the target directory. For example:

```
Directory of D:\compact_wse

09/22/2004  01:13p      <DIR>          .
09/22/2004  01:13p      <DIR>          ..
08/06/2004  02:16p                  27 autorun.inf
09/22/2004  01:12p      <DIR>          DB2
09/22/2004  01:25p              65,129 db2iprune.log
09/22/2004  01:24p      <DIR>          DOC
08/13/2004  11:17a              32,831 setup.exe
```

The log file contains a detailed list of all the deletion actions that

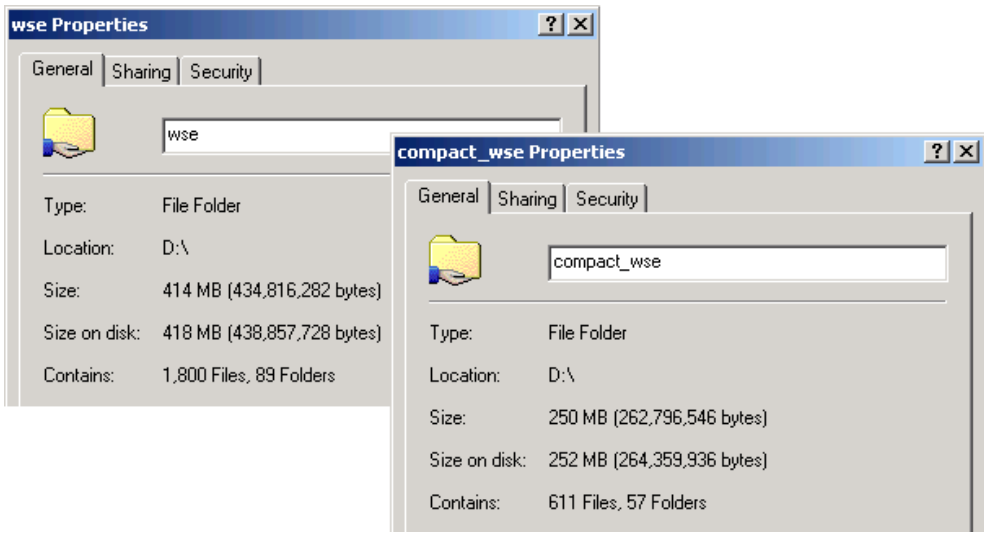

Figure 1: Size comparison

were performed. Below is a partial example of db2iprune.log to give you some idea of its structure:

```
Processing: FEAT_WEBDB2TOOLS.WSE
Deleting: D:\compact_wse\db2\windows\WEBDB2~1.cab
Processing: FEAT_DATA_WH_CENTER.WSE
Deleting: D:\compact_wse\db2\windows\DWC_IC~1.cab
Deleting: D:\compact_wse\db2\windows\DWC_IC~2.cab
Deleting: D:\compact_wse\db2\windows\DWC_IC~3.cab
...
Processing: LANG_INFOPOPS_CC390_zh_TW.WSE
Deleting: D:\compact_wse\db2\windows\IN4DA5~1.cab
```

INSTALLING A PRUNED IMAGE

The end result of using the db2iprune utility to remove any unwanted components (and their prerequisites, which are automatically removed as well) is the same as if you had deselected the specified components at installation time. In this case, however, these components didn't contribute to the size of the image.

Interactive installation

During an interactive installation (using the DB2 setup wizard), only those components that remain in the pruned image are shown. For example, if you removed the DB2 Data Warehouse Center from the installation image, this feature would not be presented as an optional or pre-selected component in the setup wizard. (Note, however, that compact installations represent the minimum file set required by a DB2 UDB product on Windows, and these components cannot be further reduced by the db2iprune utility.)

Figure 1 can be used to compare the size of a full installation image with our pruned image. As you can see, the size reduction is substantial.

Unattended (silent) installation

During a silent installation (using a response file), any components that are specified in the response file but are not in the pruned installation image are ignored.

Applying maintenance

The maintenance application process for a DB2 UDB product is the same regardless of whether or not the installation image was a full image or a pruned image. Maintenance to a DB2 UDB product that was installed from a pruned image is seamless.

In Windows environments, both FixPaks and Updates are fully installable images. This means that you can install a DB2 UDB product at any level, and it will default to ‘try-and-buy’ mode if a licence key isn’t found. Because DB2 UDB for Windows FixPaks and Updates are always fully installable images, the db2iprune utility can be used for maintenance vehicles as well. However, you must ensure that the pruned FixPak or Update image that you create contains all the components that were initially installed. If a component is missing from these pruned images, an error message detailing the missing cabinet (.cab) files will be returned when maintenance is performed.

CONCLUSION

We have shown how you can use the db2iprune utility, available in DB2 UDB V8.2, to delete unneeded components from a DB2 UDB for Windows installation image. The resulting smaller image (associated with significantly reduced network traffic, storage requirements, and installation time) has the potential to greatly simplify your mass deployment scenarios.

Roman B Melnyk (roman_b_melnyk@hotmail.com)
DB2 Information Development team
Paul C Zikopoulos (paulz_ibm@msn.com)
IBM Database Competitive Technology team
IBM (Canada)

DB2 attachment primer: part 2 – using DB2 attachments

DETAILS OF THE DB2 PRECOMPILER PROCESS

The process to prepare a mainframe application program with embedded SQL is shown in Figure 1. Inputs to the DB2 precompiler process include:

- The source code containing the embedded SQL statements.
- Existing high-level language copybooks.
- Previously-defined DB2 DCLGEN copybooks.
- Parameters for the precompiler program.

One of the parameters for the DB2 precompiler program is ATTACH, which is used to specify the type of attachment facility that will be used by the program at execution time. The options for the ATTACH parameter are TSO, CAF, and RRSAF. When not specified, the default parameter is ATTACH(TSO).

Two outputs are generated by the pre-compile process, a

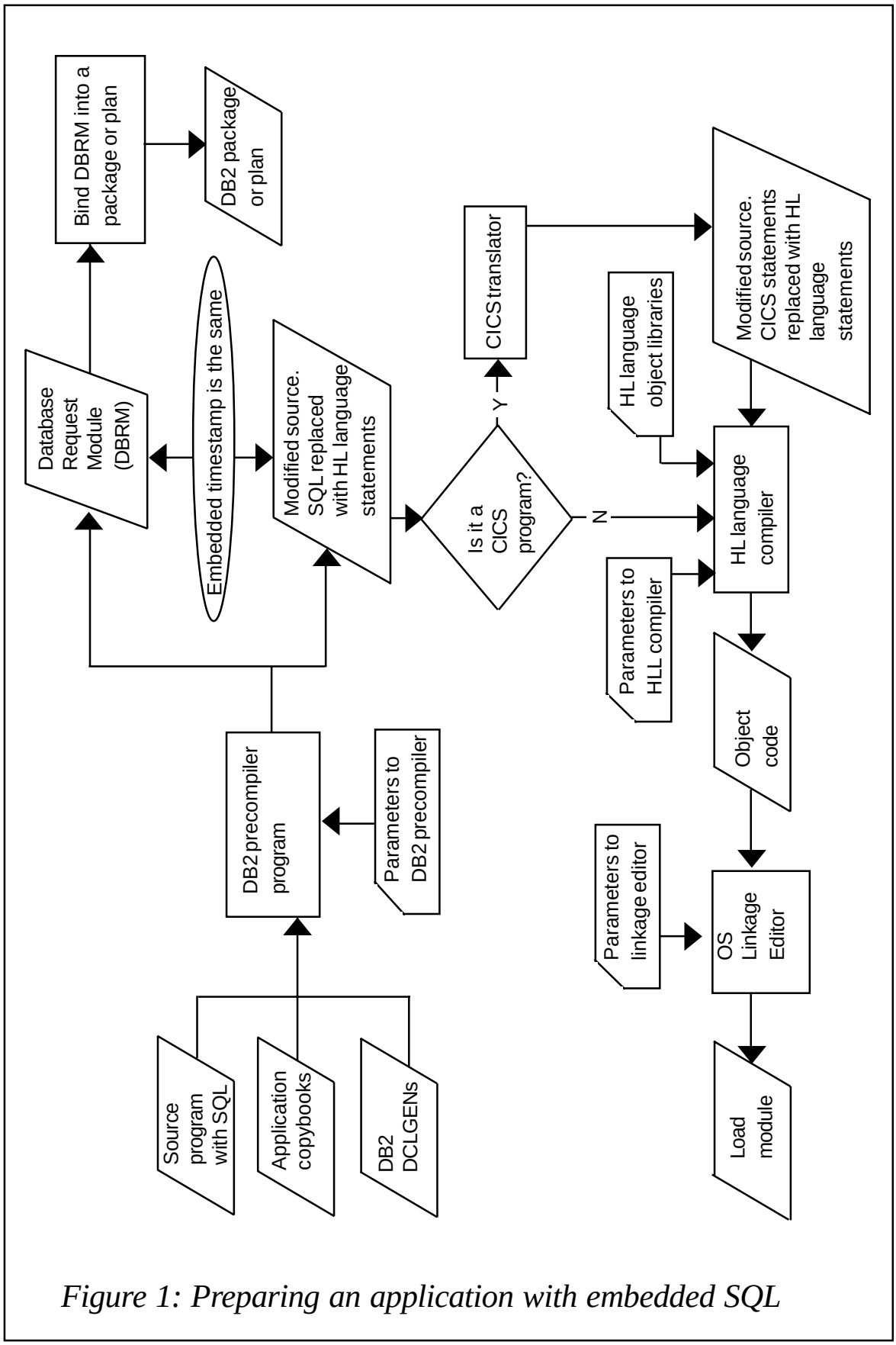

Figure 1: Preparing an application with embedded SQL

Database Request Module (also known as the DBRM) and an updated version of the source code, where all of the SQL statements have been converted into high-level language statements. It is during the pre-compile process that the DB2 program timestamp value is embedded in the source code and the DBRM. This value is used at run-time to ensure that the DBRM and the load module are in sync.

An excerpt from a COBOL program after being pre-compiled with ATTACH(TSO) looks like this:

```
122200*****EXEC SQL COMMIT END-EXEC                    12220015
      PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
      CALL "DSNHLI" USING SQL-PLIST10
      IF SQLCODE < 0 GO TO 2000-COMMIT-ERROR ELSE
      IF SQLCODE > 0 AND SQLCODE NOT = 100
      OR SQLWARN0 = "W" GO TO 2000-COMMIT-ERROR ELSE
      IF SQLCODE = 100 GO TO 2000-COMMIT-ERROR ELSE
      MOVE 1 TO SQL-INIT-FLAG
      END-IF
      END-IF
      END-IF
122300 GO TO 2000-COMMIT-DB2-WORK-EXIT.                12230015
```

Notice how the SQL statements are commented out and replaced by COBOL statements. Pay attention to the COBOL statement CALL "DSNHLI". DSNHLI is the actual name of the DB2 load module that will be called whenever the application program starts executing an SQL statement.

If the very same program had been pre-compiled with the ATTACH(CAF) parameter, the above COBOL statements would look like this:

```
122200*****EXEC SQL COMMIT END-EXEC                    12220015
      PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
      CALL "DSNHLI2" USING SQL-PLIST10
      IF SQLCODE < 0 GO TO 2000-COMMIT-ERROR ELSE
      IF SQLCODE > 0 AND SQLCODE NOT = 100
      OR SQLWARN0 = "W" GO TO 2000-COMMIT-ERROR ELSE
      IF SQLCODE = 100 GO TO 2000-COMMIT-ERROR ELSE
      MOVE 1 TO SQL-INIT-FLAG
      END-IF
      END-IF
      END-IF
122300 GO TO 2000-COMMIT-DB2-WORK-EXIT.                12230015
```

In this case, the name of the DB2 load module that will be called is DSNHLI2.

For ATTACH(RRSAF), the precompiler output would look like this:

```
122200*****EXEC SQL COMMIT END-EXEC                    12220015
      PERFORM SQL-INITIAL UNTIL SQL-INIT-DONE
      CALL "DSNHLIR" USING SQL-PLIST10
      IF SQLCODE < 0 GO TO 2000-COMMIT-ERROR ELSE
      IF SQLCODE > 0 AND SQLCODE NOT = 100
      OR SQLWARN0 = "W" GO TO 2000-COMMIT-ERROR ELSE
      IF SQLCODE = 100 GO TO 2000-COMMIT-ERROR ELSE
      MOVE 1 TO SQL-INIT-FLAG
      END-IF
      END-IF
      END-IF
122300 GO TO 2000-COMMIT-DB2-WORK-EXIT.                12230015
```

In this case, the name of the DB2 load module that will be called is DSNHLIR.

If you were to examine all the statements in the precompiler output, you would find no other differences between the three listings.

DETAILS OF THE HIGH-LEVEL PROGRAMMING COMPILER PROCESS

The purpose of the compiler process is to convert the high-level language statements into a set of formatted machine-independent statements that can be post-processed by the operating system linkage editor and transformed into an executable module.

This program's intermediate state after the compilation process is referred to as object code or object code module, and it is not executable.

There is really not much to be said about the compiler process in itself, other than to suggest that the compiler default parameters be reviewed to ensure that they are the appropriate ones for your environment.

Remember that there are two ways of performing program preparation – one in which the DB2 precompiler is used, and another in which the high-level language compiler itself does the precompilation of the SQL statements. This second approach is not discussed in this article, but it does have several advantages over the traditional program preparation approach, so it is recommended that you review it at your leisure.

DETAILS OF THE OS LINKAGE EDITOR PROCESS

After the object code module is created, the next step is to generate the load module.

OS/390 and z/OS provide multiple utilities, including one called the linkage editor (also know as LKED) and another one called the BINDER. The newer releases of the operating system have replaced LKED with the BINDER, which is a program that processes the output of language translators and compilers into an executable load module. During the remainder of this article, we will continue to refer to the LKED utility, although the information presented would apply to either of them.

The manual *MVS Program Management: User's Guide and Reference* for z/OS V1.4 can be downloaded from http://publibz.boulder.ibm.com/epubs/pdf/iea2b121.pdf, and it describes both the LKED and BINDER utilities in detail.

The LKED utility helps resolve references to external load modules found in the object code created by the high-level language compiler, including the earlier-mentioned references to the DB2 attachment modules DSNHLI, DSNHLI2, and DSNHLIR.

Parameters to LKED tell it how to combine all these load modules with the object code, what type of addressability to use, and what the final name is for the load module, etc.

The important information on the LKED utility is that it helps to resolve the references to the DB2 attachment names embedded in the object code. The time to resolve the references to

external load modules is dictated by the CALL/NOCALL parameter specified at LKED time.

The CALL parameter tells the LKED utility to search in the list of datasets specified by SYSLIB DDNAME looking for a match to an external load module reference. If a match is found (and no overriding commands are specified), then LKED will combine into a single load module the executable version of the application program object code plus all the reference-matching load modules found in the SYSLIB search. Because of this SYSLIB search at LKED time, it is very important to know the order of datasets listed in the SYSLIB DDNAME concatenation. In particular, when linking a CICS program, the CICS libraries should be the first in the SYSLIB DDNAME concatenation. It is the same for IMS programs.

The NOCALL parameter tells the LKED utility not to resolve the external references at this time, but to convert the application program object code into a load module capable of looking for the external load modules at program execution time. So the order of the datasets specified in the STEPLIB DDNAME at execution time then becomes important. In particular, for a CICS program, the CICS load library should be ahead of the DB2 or IMS load libraries in the STEPLIB DDNAME concatenation. The same applies for IMS programs.

Lots of information on how to link edit programs can be found in IBM's DB2 *Application Programming and SQL Guide* manuals.

WHAT ABOUT A DB2 UNIVERSAL ATTACHMENT?

Why didn't IBM, many years ago, come up with a single DB2 universal attachment? It certainly would have simplified the life of everyone working with DB2 on the mainframe. My guess is that until recently, the technology wasn't mature enough to allow it, particularly when trying to maintain the integrity of a unit of work across multiple subsystems.

Now that RRS is in place, why can't it be used as the basis for

a DB2 universal attachment? Is it because it has too much overhead? Or is it because not enough customers have requested it? I leave you with the last question to ponder.

Antonio Salcedo
DB2 Systems Programmer (USA)

Implementing image extender to retrieve a signature database

This article shows an example of implementing the concept of DB2 image extenders. The application that uses DB2 extender's Query by Image Content (QBIC) capability was developed with the intention of aiding in signature counterfeit detection. Minimal prerequisites are IAV extenders 7.1 and JDK 1.1.6.

By using QBIC technology, traditional SQL queries are enhanced with additional types of searches where criteria are similar colours and texture patterns related to the source image. The visual features of an image that queries can reference include average colour, histogram colour, positional colour, and texture. In the application described below, texture, which represents the measure of the coarseness, contrast, and directionality of an image, is used for pattern recognition. Images stored in the database are actually signatures scanned from some documents using the same resolution, set into frames of equal dimensions, and with colours converted to black and white in order to use only visual feature texture for searching. Colour conversion was applied with the intention of eliminating the consequences of bad quality prints on the query results.

For the purpose of testing and statistical analysis, the special naming convention is applied. Data includes signatures made by groups of people that signed as themselves and some who tried to falsify the signatures of other people from the group. Each signature is repeated several times by the same person

regardless of whether it is original or fake. A unique sequence number is assigned to each person in the group and all signatures are stored in the files of type UxxUyyz.jpg, where xx represents the identifier of the person whose signature is falsified, yy the identifier of the 'counterfeiter' and z the ordinal number of an attempt made by person yy to sign on as person xx. The files of type UxxUxxz.jpg represent not counterfeits but the *z*th authentic signature made by person xx.

The file AdminExt.bat consists of the steps necessary to prepare the environment for the actual Query by Image Content. Extender services are started by dmbstart in the previously-initialized command line environment with the db2cmd command. In general the procedure to implement the DB2 image extender application consists of some administrative tasks, like enabling the database, table, and column(s) for image extender and creating the QBIC catalog. When images are stored in the database, image extender computes their visual features and records their corresponding values in a QBIC catalog. In this actual case, the insertion of signatures from files UxxUyyz.jpg into table MMDBSYS.POTPISI is done by program Populate.java, while program QbicQry.java does the search for an image by content. The source image is chosen from among the available image files UxxUyyz.jpg through an Open file dialog, and the texture value for the source is compared with the corresponding values for already-catalogued images. The resulting scores represent a measure of similarity between the source and target images, which, along with identification and current timestamp, are stored in the table MMDBSYS.POTPISISTAT.

The file Query.bat can be used to run queries after the first execution of AdminExt.bat. Environmental variables in both batch files should be customized according to your internal standards. The name of the folder C:\db2extenders\MySamples referenced in programs Populate.java and QbicQry.java can be changed as well.

Similarity scores in the report resulting from the query are

shown in ascending order and with different colours (see Figure 1).

The red colour indicates a signature equal to the source image (UxxUyyz.jpg) and repetitions of that signature (only value z changes), blue indicates other signatures made by the same person yy, and black all other cases where the person in question (yy) is neither a signer nor a counterfeiter.

From analysis of the resulting statistical table, it is obvious that the signatures with scores close to a zero value are the most similar to the source signature. Deviations from zero in some cases of authentic signatures (UxxUxxz) can be explained by the intensity and quality of prints. The ideal situation for scores of similarity (when in an ordered array) is red first, then blue, and lastly black. Our interest is focused on people whose signatures, no matter whether authentic or fake, belong to the ascending array of signatures up to and including the last blue-coloured signature. In order to analyse the validity of results obtained by this method, ordinal numbers are assigned to each score (in ascending order) for each image UxxUyyz.

The pie chart and column chart shown in Figure 3 and Figure 4 result from the query based on the table shown in Figure 2, which includes ordinal numbers of scores for each pattern grouped by colour, and criteria that satisfy our needs.

The conclusion is that in 75% of cases the results obtained by this application can be considered reliable.

When no software for pattern recognition is available, this application can help reduce the size of the population to which the possible counterfeiter may belong. The main effect accomplished by the implementation of this method is that further investigation can be speeded up by checking a significantly smaller group of ‘suspects’.

POPULATE.JAVA

```
/*************************************************************************/
/** Source File Name = Populate.java                                    **/
```

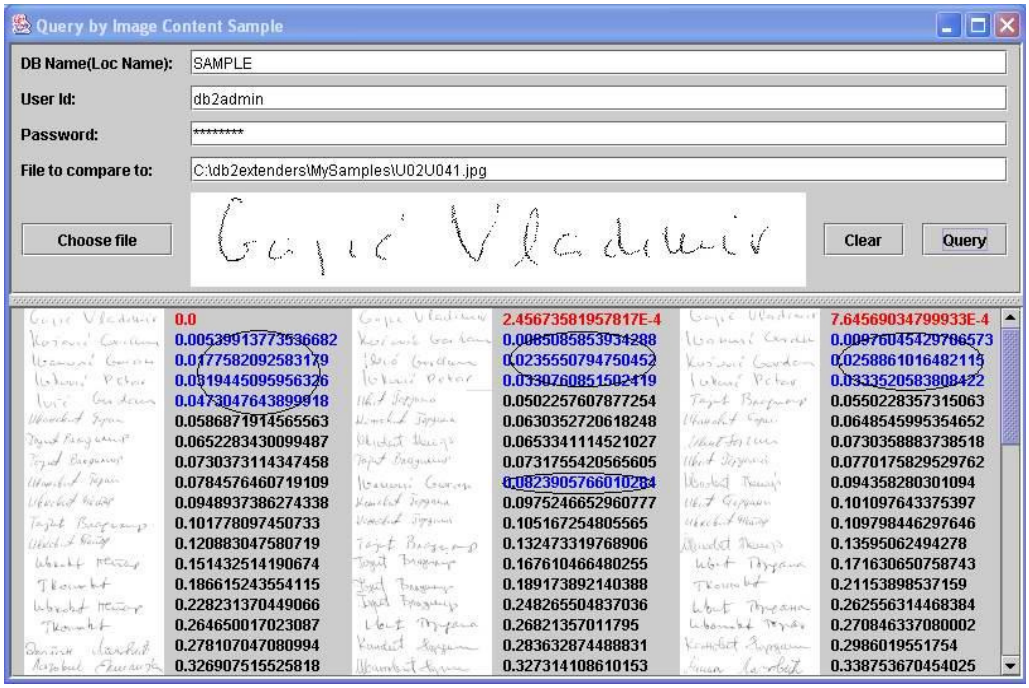

Figure 1: QbicQry Report Screenshot

```
/** FUNCTIONS USED:    DB2Image import udf                                  **/
/** CONFIGURATION:     Database already created and enabled                 **/
/** USAGE:             Populate                                             **/
/**                    the program will prompt for the database name,       **/
/**                    userid, and password. Use the userid and             **/
/**                    password that created the database.                  **/
/** Note: if specifying to store as a blob, the target_file must            **/
/**                    be null                                              **/
/**    To specify a null value in the DB2IMAGE UDF, you must pass in        **/
/**          "CAST(NULL AS LONG VARCHAR)"                                   **/
/**       If the source_file is a JPG, the format must be 'JPG'.            **/
/**          it cannot be 'ASIS'                                            **/
/**     DB2IMAGEIMPORTC1 Store content from buffer or client file in        **/
/**     either a blob or pointer to the target_file on the server           **/
/**       DB2IMAGE(dbname (varchar), content (blob), source_format          **/
/**           (varchar), stortype (integer 1=blob,2=filepointer),           **/
/**           target_file (long varchar), comment (long varchar))           **/
/*************************************************************************/
import COM.ibm.db2.app.*;
import COM.ibm.db2.jdbc.app.*;
```

	PATTERN	RED	BLUE	BLACK
	U04U013	1, 8, 14	22, 26, 27, 38, 47, 51, 54, 58, 60, 61, 78, 83	2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 15, 16, 17, 18,
	U04U021	1, 10, 16	2, 3, 4, 5, 6, 7, 9, 11, 12, 13, 15, 18	8, 14, 17
	U04U022	1, 3, 11	2, 4, 6, 7, 8, 9, 10, 12, 13, 15, 17, 19	5, 14, 16, 18
	U04U023	1, 3, 9	2, 4, 5, 6, 8, 10, 11, 12, 13, 14, 15, 18	7, 16, 17
	U04U031	1, 17, 21	3, 4, 5, 10, 11, 12, 16, 18, 19, 23, 25, 28	2, 6, 7, 8, 9, 13, 14, 15, 20, 22, 24, 26, 27
	U04U032	1, 6, 12	13, 14, 15, 19, 20, 21, 23, 24, 25, 26, 27, 29	2, 3, 4, 5, 7, 8, 9, 10, 11, 16, 17, 18, 22, 28
	U04U033	1, 6, 10	12, 13, 14, 19, 20, 21, 23, 24, 25, 26, 27, 29	2, 3, 4, 5, 7, 8, 9, 11, 15, 16, 17, 18, 22, 28
	U04U041	1, 3, 4	2, 7, 9, 14, 16, 18, 20, 26, 27, 30, 31, 111	5, 6, 8, 10, 11, 12, 13, 15, 17, 19, 21, 22, 23, 24,
	U04U042	1, 2, 4	3, 5, 6, 10, 13, 14, 17, 18, 20, 21, 29, 111	7, 8, 9, 11, 12, 15, 16, 19, 22, 23, 24, 25, 26, 27,
	U04U043	1, 2, 3	4, 5, 7, 12, 13, 16, 17, 22, 23, 24, 29, 111	6, 8, 9, 10, 11, 14, 15, 18, 19, 20, 21, 25, 26, 27,
	U04U051	1, 9, 39	3, 12, 18, 19, 21, 22, 23, 51, 63, 68, 69, 72	2, 4, 5, 6, 7, 8, 10, 11, 13, 14, 15, 16, 17, 20, 24
	U04U052	1, 12, 15	3, 4, 5, 8, 13, 17, 26, 39, 41, 75, 80, 85	2, 6, 7, 9, 10, 11, 14, 16, 18, 19, 20, 21, 22, 23,
	U04U053	1, 15, 41	5, 6, 7, 16, 20, 31, 35, 40, 48, 89, 91, 95	2, 3, 4, 8, 9, 10, 11, 12, 13, 14, 17, 18, 19, 21, 2
	U04U061	1, 2, 7	3, 4, 6, 10, 12, 15, 19, 20, 26, 27, 28, 29, 38,	5, 8, 9, 11, 13, 14, 16, 17, 18, 21, 22, 23, 24, 25,
	U04U062	1, 4, 6	2, 3, 7, 9, 11, 15, 19, 20, 26, 27, 28, 29, 38, 3	5, 8, 10, 12, 13, 14, 16, 17, 18, 21, 22, 23, 24, 25
	U04U063	1, 5, 6	2, 3, 4, 7, 9, 15, 19, 20, 26, 27, 28, 29, 38, 39	8, 10, 11, 12, 13, 14, 16, 17, 18, 21, 22, 23, 24, 2
	U05U011	1, 5, 9	2, 3, 10, 31, 32, 34, 39, 45, 52, 81, 95, 102	4, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,

Figure 2: Table from external program for statistical analysis

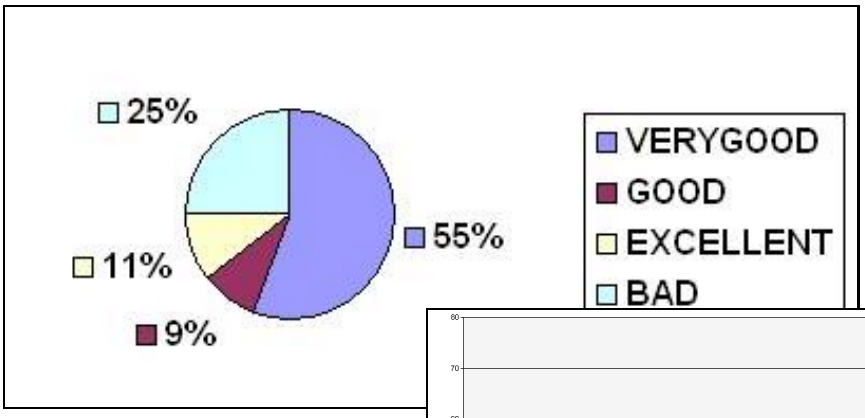

Figure 3: Pie chart

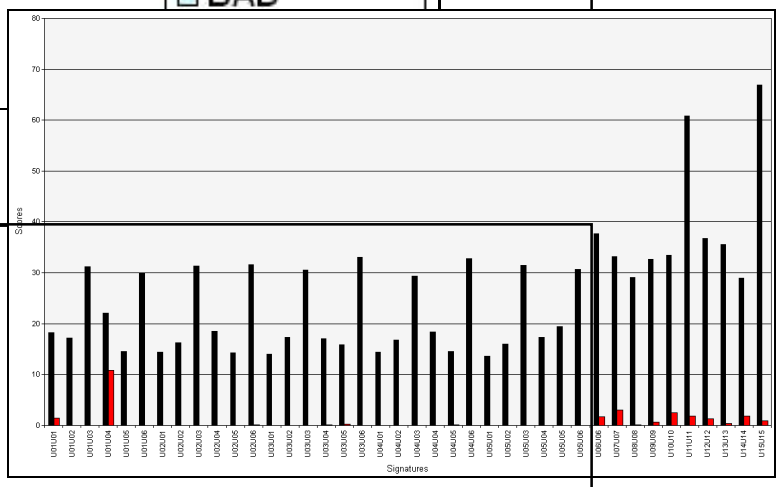

Figure 4: Column chart

```
import java.sql.*;
import java.io.*;
import javax.swing.*;
import java.awt.event.*;

public class Populate extends JPanel implements ActionListener {
  static JFrame frame;
  JButton selectFiles;
  JButton cancelButton;
  JFileChooser chooser;
  static String dbName = null;
  static String uid = null;
  static String pwd = null;

  public Populate() {
    selectFiles = new JButton("Open Files");
    selectFiles.addActionListener(this);
    selectFiles.setMnemonic('O');
    cancelButton = new JButton("    Cancel   ");
    cancelButton.addActionListener(this);
    cancelButton.setMnemonic('C');
    JPanel panel = new JPanel();
    panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));
    panel.add(selectFiles);
    panel.add(cancelButton);
    add(panel);
  }

  public static void main(String argv[]) {
    try {
      if (argv.length > 3 || (argv.length >= 1 && argv[0] == "?")) {
        System.out.print("Syntax for populating data on DB2 UDB
                                                for OS/390 server:\n"+
                  "   java Populate location_name userid password\n\n"+
                     "Syntax for populating data on DB2 UDB server:\n"+
                   "   java Populate database_name userid password\n");
        System.exit(0);
      } else if (argv.length == 0) {
              BufferedReader cl = new BufferedReader (new
InputStreamReader(System.in));
        System.out.print("Enter database name or DB2 location name:\n");
              dbName = cl.readLine();
              System.out.print("Enter userid:\n");
              uid = cl.readLine();
              System.out.print("Enter password:\n");
              pwd = cl.readLine();
            } else if (argv.length == 1) {
                    dbName = argv[0];
                  } else if (argv.length == 3) {
                          dbName = argv[0];
```

```
                            uid = argv[1];
                            pwd = argv[2];
                          }
    } catch (IOException e) {
      System.out.println("\n-- IOException caught --\n");
    }
    Populate panel = new Populate();
    frame = new JFrame("Open Files for Insert in Database");
    frame.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {System.exit(0);}
      });
    frame.getContentPane().add("Center", panel);
    frame.pack();
    frame.setSize(400, 100);
    frame.setVisible(true);
  }

  public void actionPerformed(ActionEvent ev) {
    if (ev.getActionCommand().equals("Open Files")) {
      String dir = "c:\\db2extenders\\MySamples\\";
      chooser = new JFileChooser(dir);
      MyFileFilter filter = new MyFileFilter("jpg", "JPEG Images");
      chooser.addChoosableFileFilter(filter);
      chooser.setDialogTitle("Select an image file(s)");
      chooser.setMultiSelectionEnabled(true);
      int retval = chooser.showDialog(frame, null);
      if (retval == JFileChooser.APPROVE_OPTION) {
        File[] files = chooser.getSelectedFiles();
        Connection con = null;
        String url = null;
        java.sql.Statement stmt = null;
        java.sql.PreparedStatement ps = null;
        String source_format = "\'JPG\'";
        int stortype = 1;
        String target_file = "CAST(NULL AS LONG VARCHAR)";
        try {
           Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
          try {
            url = "jdbc:db2:" + dbName;
            con = DriverManager.getConnection(url, uid, pwd);
            stmt = con.createStatement();
            stmt.executeUpdate("SET CURRENT FUNCTION PATH = mmdbsys,
CURRENT FUNCTION PATH");
            stmt.close();
            String kljuc = null;
            byte[] content = null;
            String comment = null;
            String sql = null;
            for (int i = 0; i < files.length; ++i) {
              kljuc = files[i].getName();
```

```
              content = getContent(files[i].getPath());
              comment = files[i].getName();
              sql = "INSERT INTO POTPISI VALUES ("
                      + "\'" + kljuc + "\', "
                      + "DB2Image(CURRENT SERVER, "
                    +       "CAST(? AS BLOB(" + content.length + ")), "
                      +       source_format + ", "
                      +       stortype + ", "
                      +       target_file + ", "
                      +       "\'" + comment + "\'"
                      + "))";
              System.out.println(sql);
              ps = con.prepareStatement(sql);
              ps.setBytes(1, content);
              ps.execute();
              ps.close();
            }
            con.close();
          } catch (SQLException ex) {
            while (ex != null) {
              System.out.println("\n-- SQLException caught --\n");
              System.out.println("Message:   " + ex.getMessage ());
              System.out.println("SQLState:  " + ex.getSQLState ());
              System.out.println("ErrorCode: " + ex.getErrorCode ());
              ex = ex.getNextException();
            }
          }
        } catch (Exception ex) {
          ex.printStackTrace();
        }
     } else if (retval == JFileChooser.CANCEL_OPTION) {
                JOptionPane.showMessageDialog(frame, "User cancelled
operation. No file was chosen.");
             } else if (retval == JFileChooser.ERROR_OPTION) {
                      JOptionPane.showMessageDialog(frame, "An error
occurred. No file was chosen.");
                    } else {
   JOptionPane.showMessageDialog(frame, "Unknown operation occurred.");
                    }
    }
    if (ev.getActionCommand().equals("    Cancel   ")) {
      System.exit(Ø);
    }
  }

  static private byte[] getContent(String filename) {
    byte[] content = null;
    try {
      java.io.FileInputStream file = new
java.io.FileInputStream(filename);
```

```
      int available = file.available();
      content = new byte[available];
      file.read(content);
      file.close();
    } catch (java.io.FileNotFoundException e) {
      System.out.println("getContent: " + e);
    } catch (java.io.IOException e) {
      System.out.println("getContent: " + e);
    }
    return content;
  }

}
```

MYFILEFILTER.JAVA

```
/*********************************************************************/
/** Source File Name = MyFileFilter.java                            **/
/** USAGE:            Open file dialog                              **/
/*********************************************************************/
import java.io.File;
import java.util.Hashtable;
import java.util.Enumeration;
import javax.swing.*;
import javax.swing.filechooser.*;

public class MyFileFilter extends FileFilter {

    private static String TYPE_UNKNOWN = "Type Unknown";
    private static String HIDDEN_FILE = "Hidden File";

    private Hashtable filters = null;
    private String description = null;
    private String fullDescription = null;
    private boolean useExtensionsInDescription = true;

    public MyFileFilter() {
      this.filters = new Hashtable();
    }
    public MyFileFilter(String extension) {
      this(extension,null);
    }
    public MyFileFilter(String extension, String description) {
      this();
      if(extension!=null) addExtension(extension);
       if(description!=null) setDescription(description);
    }

    public MyFileFilter(String[] filters) {
```

```
      this(filters, null);
    }

    public MyFileFilter(String[] filters, String description) {
      this();
      for (int i = 0; i < filters.length; i++) {
          addExtension(filters[i]);
      }
       if(description!=null) setDescription(description);
    }

    public boolean accept(File f) {
      if(f != null) {
          if(f.isDirectory()) {
            return true;
          }
          String extension = getExtension(f);
          if(extension != null && filters.get(getExtension(f)) != null)
{
            return true;
          };
      }
      return false;
    }

     public String getExtension(File f) {
      if(f != null) {
          String filename = f.getName();
          int i = filename.lastIndexOf('.');
          if(i>0 && i<filename.length()-1) {
            return filename.substring(i+1).toLowerCase();
          };
      }
      return null;
    }

    public void addExtension(String extension) {
      if(filters == null) {
          filters = new Hashtable(5);
      }
      filters.put(extension.toLowerCase(), this);
      fullDescription = null;
    }

    public String getDescription() {
      if(fullDescription == null) {
          if(description == null || isExtensionListInDescription()) {
         fullDescription = description==null ? "(" : description + " (";
            Enumeration extensions = filters.keys();
            if(extensions != null) {
```

```
            fullDescription += "." + (String) extensions.nextElement();
                while (extensions.hasMoreElements()) {
          fullDescription += ", ." + (String) extensions.nextElement();
                }
            }
            fullDescription += ")";
          } else {
            fullDescription = description;
          }
        }
        return fullDescription;
    }

    public void setDescription(String description) {
      this.description = description;
      fullDescription = null;
    }
    public void setExtensionListInDescription(boolean b) {
      useExtensionsInDescription = b;
      fullDescription = null;
    }

    public boolean isExtensionListInDescription() {
      return useExtensionsInDescription;
    }
}
```

QBICQRY.JAVA

```
/*****************************************************************************/
/** Source File Name = QbicQry.java                                         **/
/** FUNCTIONS USED:                                                         **/
/**          MMDBSYS.QbScoreFromStr(query_spec, qbic_colum_name)            **/
/**                    MMDBSYS.Thumbnail(qbic_colum_name)                   **/
/**                    MMDBSYS.Comment(qbic_colum_name)                     **/
/**                    MMDBSYS.Filename(qbic_colum_name)                    **/
/** CONFIGURATION:     Database already created and enabled                 **/
/** USAGE:             QbicQry                                              **/
/**             the program will prompt for the database name,              **/
/**             userid, and password.  Use the userid and password          **/
/**             that created the database.                                  **/
/*****************************************************************************/
import java.awt.*;
import java.io.*;
import java.sql.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
```

```
public class QbicQry extends JFrame implements ActionListener {
  static String dbName = null;
  static String uid = null;
  static String pwd = null;
  int error = 0;
  private String tableName = "POTPISI";
  private String imageColumn = "POTPIS";
  JLabel icon_label = new JLabel(new ImageIcon());
  JLabel score_label = new JLabel();
  Container contentPane = getContentPane();
  GridBagLayout layout = new GridBagLayout();
  GridBagConstraints constraints = new GridBagConstraints();
  JPanel container = new JPanel();
  JScrollPane scrollPane = new JScrollPane(container);
  JSplitPane splitPane = new JSplitPane(JSplitPane.VERTICAL_SPLIT);
  JPanel containerx = new JPanel();
  JScrollPane iconPane = new JScrollPane(containerx);
  JLabel compareToImage;
  JTextField fileName = new JTextField();
  JTextField dbNameF = new JTextField();
  JTextField uidF = new JTextField();
  JPasswordField pwdF = new JPasswordField();
  private String[] retrievedFiles;
  private String retrievedDir = "\\db2extenders\\mysamples\\";

  public QbicQry() {
    setTitle("Query by Image Content Sample");
    containerx.add(icon_label);
    containerx.add(score_label);
    splitPane.setTopComponent(scrollPane);
    splitPane.setBottomComponent(iconPane);
    contentPane.add(splitPane);
    setSize(900, 700);
    scrollPane.createHorizontalScrollBar();
    scrollPane.createVerticalScrollBar();
    iconPane.createHorizontalScrollBar();
    iconPane.createVerticalScrollBar();
    JLabel dbNameLabel = new JLabel("DB Name(Loc Name):");
    dbNameF.setText(dbName);
    JLabel uidLabel = new JLabel("User Id:");
    uidF.setText(uid);
    JLabel pwdLabel = new JLabel("Password:");
    pwdF.setText(pwd);
    JLabel fileNameLabel = new JLabel("File to compare to:");
    JButton fileChooser_button = new JButton("Choose file");
    fileChooser_button.addActionListener(this);
    fileChooser_button.setActionCommand("FILE");
    JButton query_button = new JButton("Query");
    query_button.addActionListener(this);
    query_button.setActionCommand("QUERY");
```

```
JButton clear_button = new JButton("Clear");
clear_button.addActionListener(this);
clear_button.setActionCommand("CLEAR");
constraints.weightx = Ø.3;
constraints.weighty = Ø.6;
constraints.insets = new Insets(2, 7, 2, 7);
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
layout.setConstraints(dbNameLabel, constraints);
constraints.gridwidth = GridBagConstraints.REMAINDER;
layout.setConstraints(dbNameF, constraints);
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
layout.setConstraints(uidLabel, constraints);
constraints.gridwidth = GridBagConstraints.REMAINDER;
layout.setConstraints(uidF, constraints);
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
layout.setConstraints(pwdLabel, constraints);
constraints.gridwidth = GridBagConstraints.REMAINDER;
layout.setConstraints(pwdF, constraints);
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
layout.setConstraints(fileNameLabel, constraints);
constraints.gridwidth = GridBagConstraints.REMAINDER;
layout.setConstraints(fileName, constraints);
constraints.gridwidth = 1;
ImageIcon icon = new ImageIcon();
compareToImage = new JLabel(icon);
constraints.gridwidth = GridBagConstraints.HORIZONTAL;
layout.setConstraints(compareToImage, constraints);
constraints.gridwidth = 1;
constraints.fill = GridBagConstraints.HORIZONTAL;
layout.setConstraints(fileChooser_button, constraints);
constraints.gridwidth = GridBagConstraints.HORIZONTAL;
layout.setConstraints(clear_button, constraints);
constraints.gridwidth = GridBagConstraints.REMAINDER;
layout.setConstraints(query_button, constraints);
container.setLayout(layout);
container.add(dbNameLabel);
container.add(dbNameF);
container.add(uidLabel);
container.add(uidF);
container.add(pwdLabel);
container.add(pwdF);
container.add(fileNameLabel);
container.add(fileName);
container.add(fileChooser_button);
container.add(compareToImage);
container.add(clear_button);
```

```
    container.add(query_button);
    addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent ev) {
        System.exit(0);
      }
    });
  }

  public static void main(String[] argv) {
    try {
      if (argv.length > 3 || (argv.length >= 1 && argv[0] == "?")) {
          System.out.print("Syntax on DB2 UDB for OS/390 server:\n"+
                    "   java QbicQry location_name userid password\n\n"+
                        "Syntax on DB2 UDB server:\n"+
                     "   java QbicQry database_name userid password\n");
          System.exit(0);
      } else if (argv.length == 0) {
                BufferedReader cl = new BufferedReader (new
InputStreamReader(System.in));
       System.out.print("Enter database name or DB2 location name:\n");
                dbName = cl.readLine();
                System.out.print("Enter userid:\n");
                uid = cl.readLine();
                System.out.print("Enter password:\n");
                pwd = cl.readLine();
              } else if (argv.length == 1) {
                        dbName = argv[0];
                      } else if (argv.length == 3) {
                                dbName = argv[0];
                                uid = argv[1];
                                pwd = argv[2];
                              }
    } catch (IOException e) {
      System.out.println("\n-- IOException caught --\n");
    }
    QbicQry q = new QbicQry();
    q.setLocation(60, 10);
    q.setVisible(true);
  }

  public void actionPerformed(ActionEvent ev) {
    if (ev.getActionCommand().equals("FILE")) {
      processSelectFileAction();
    }
    if (ev.getActionCommand().equals("CLEAR")) {
      processClearAction();
    }
    if (ev.getActionCommand().equals("QUERY")) {
      processQueryAction();
    }
```

```
  }

  private void processSelectFileAction() {
    JFileChooser chooser = new JFileChooser(retrievedDir);
    MyFileFilter filter = new MyFileFilter("jpg", "JPEG Images");
    chooser.addChoosableFileFilter(filter);
    chooser.setDialogTitle("Select an image file");
    int returnVal = chooser.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
      String fileSelected = chooser.getCurrentDirectory() + "\\" +
chooser.getSelectedFile().getName();
      fileName.setText(fileSelected);
      ImageIcon icon = new ImageIcon(fileSelected);
      compareToImage.setIcon(icon);
      splitPane.setDividerLocation(Ø.3);
    }
  }

  private void processClearAction() {
    containerx.removeAll();
    containerx.invalidate();
    containerx.repaint();
    containerx.doLayout();
    if (retrievedFiles != null) {
      for (int i = Ø; i < retrievedFiles.length; ++i) {
        File file = new File(retrievedFiles[i]);
        file.delete();
      }
    }
  }

  private void processQueryAction() {
    String url = "jdbc:db2:" + dbNameF.getText().trim();
    error = Ø;
    String file = fileName.getText().trim();
    String qbicQuery = "\'texture file=<server," + file + ">\'";
    try {
      Connection conn = null;
      Statement stmt = null;
      ResultSet rs = null;
      try {
        Class.forName("COM.ibm.db2.jdbc.app.DB2Driver").newInstance();
        try {
          conn = DriverManager.getConnection(url, uidF.getText().trim(),
pwdF.getText().trim());
        } catch (java.sql.SQLException e) {
          throw e;
        } catch (Exception e) {
          throw new java.sql.SQLException(e.toString());
        }
```

```
        String sql = null;
        try {
          stmt = conn.createStatement();
    sql = "SET CURRENT FUNCTION PATH = MMDBSYS, CURRENT FUNCTION PATH";
          stmt.executeUpdate(sql);
        } catch (java.sql.SQLException se) {
        } catch (java.lang.UnknownError ue) {
        }

        sql = "SELECT Thumbnail(" + imageColumn +")"
                  + ",Comment(" + imageColumn +")"
                  + ",Filename(" + imageColumn +")"
  + ",QBScoreFromStr(" + qbicQuery + "," + imageColumn + ") AS SCORE" +
             " FROM " + tableName +
             " ORDER BY SCORE";
        rs = stmt.executeQuery(sql);
        int li = file.lastIndexOf("\\") + 1;
        displayScoredImages(conn, rs, file.substring(li));
      } catch (java.sql.SQLException e) {
        throw e;
      } catch (Throwable e) {
        throw new java.sql.SQLException(e.toString());
      } finally {
        try {
          if (rs != null) {
            rs.close();
          }
          if (stmt != null) {
            stmt.close();
          }
          if (conn != null) {
            conn.close();
          }
        } catch (java.sql.SQLException e) {
          throw e;
        }
      }
    } catch (SQLException ex) {
      error = 1;
      showError(this, ex.getMessage());
    }
  }

  private void displayScoredImages(Connection conn, ResultSet rs, String
file) throws java.sql.SQLException {
    java.util.Vector filesV = new java.util.Vector();
    String[] files = null;
    java.util.Vector scoreV = new java.util.Vector();
    String[] scores = null;
    int li;
```

```
    try {
      int row = 0;
      while (rs.next()) {
        System.out.println("row[" + (++row) + "]");
        if (row == 1) {
        processClearAction();
      }
        int column = 0;
        byte[] thumbnail = rs.getBytes(++column);
        String comment = rs.getString(++column);
        String fileName = rs.getString(++column);
        double score = rs.getDouble(++column);
/* Statistics begin */
        String sqlstat = "INSERT INTO POTPISISTAT VALUES ("
                     + "\'" + file + "\', "
                     + "\'" + comment + "\', "
                        + score + ", "
                     + "current timestamp"
                     + ")";
        java.sql.PreparedStatement ps = conn.prepareStatement(sqlstat);
        ps.execute();
        ps.close();
/* Statistics end   */
        System.out.println("\t thumbnail: "
          + ((thumbnail == null) ? "null" : "byte array")
          + "\t comment: "
          + comment
          + "\t fileName: "
          + fileName
          + "\t score: "
          + score);
        if (fileName != null) fileName = parseFileName(fileName);
        if (thumbnail != null) {
          String thumbnail_format = "GIF";
String useFileName = retrievedDir + row + "thumb_" + ((fileName != null)
            ? (fileName + "." + thumbnail_format)
            : (comment + "." + thumbnail_format));
          writeImageToFile(thumbnail, useFileName);
          filesV.add(useFileName);
          scoreV.add(String.valueOf(score));
        }
      }
    } catch (SQLException e) {
      throw e;
    }
    if (filesV.size() > 0) {
      files = new String[filesV.size()];
      filesV.copyInto(files);
      scores = new String[scoreV.size()];
      scoreV.copyInto(scores);
```

```
    }
    if (files != null) {
      retrievedFiles = files;
      for (int i = 0; i < files.length; ++i) {
        icon_label = new JLabel(new ImageIcon(files[i]));
        containerx.add(icon_label);
        score_label = new JLabel(scores[i]);
        li = files[i].lastIndexOf("_") + 1;
        if (files[i].substring(li,li+6).equals(file.substring(0,6)))
score_label.setForeground(Color.red);
       else if
(files[i].substring(li+3,li+6).equals(file.substring(3,6)))
score_label.setForeground(Color.blue);
             else score_label.setForeground(Color.black);
        containerx.add(score_label);
      }
      containerx.add(icon_label);
      containerx.add(score_label);
      containerx.setLayout(new GridLayout(0,6));
      containerx.repaint();
      containerx.invalidate();
      containerx.doLayout();
      iconPane.updateUI();
    }
  }

  private String parseFileName(String fileName) {
    java.util.StringTokenizer tok = new
java.util.StringTokenizer(fileName, "\\");
    int count = tok.countTokens();
    String token = null;
    for (int i = 0; i < (count-1); ++i) token = tok.nextToken();
    fileName = tok.nextToken();
    return fileName;
  }

  private void writeImageToFile(byte[] image, String filename) {
    try {
      System.out.println("writing image to file: " + filename);
      java.io.FileOutputStream file = new FileOutputStream(filename);
      file.write(image);
      file.close();
      System.out.println("image written to file: " + filename);
    } catch (java.io.FileNotFoundException e) {
      System.out.println("writeImageToFile: " + e);
    } catch (java.io.IOException e) {
      System.out.println("writeImageToFile: " + e);
    }
  }
```

```
  public void showError(JFrame f, String msg) {
    final JDialog d = new JDialog(f, "Exception Caught", true);
    d.setSize(400,150);
    JLabel l = new JLabel();
    JTextArea ta = new JTextArea();
    ta.setLineWrap(true);
    ta.setWrapStyleWord(true);
    ta.setEditable(false);
    ta.setBorder(BorderFactory.createEmptyBorder(5, 5, 5, 5));
    ta.getBorder();
    ta.append(msg);
    d.getContentPane().setLayout(new BorderLayout());
    d.getContentPane().add(ta, BorderLayout.CENTER);
    JButton b = new JButton("OK");

    b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent ev) {
        d.setVisible(false);
        d.dispose();
      }
    });
    JPanel p = new JPanel();
    p.add (b);
    d.getContentPane().add(p, BorderLayout.SOUTH);
    d.setLocationRelativeTo (f);
    d.setVisible(true);
  }

}
```

ADMINEXT.BAT

```
rem      a. Start a DB2 command line, by executing "db2cmd".
rem      b. Change directory to \db2extenders\MySamples
call dmbstart
@rem increase heap - needed for retrieving image data
db2 update db cfg for SAMPLE using app_ctl_heap_sz 4096
db2ext connect to SAMPLE user db2admin using db2admin
db2ext enable database for db2image
db2ext connect reset
db2 connect to SAMPLE user db2admin using db2admin
db2 set current function path = MMDBSYS, CURRENT FUNCTION PATH
db2 DROP TABLE POTPISI
db2 CREATE TABLE POTPISI (IME CHAR(12) NOT NULL, POTPIS DB2IMAGE NOT
NULL)
db2 DROP TABLE POTPISISTAT
db2 CREATE TABLE POTPISISTAT (PATTERN CHARACTER (15) NOT NULL, POTPIS
CHARACTER (15) NOT NULL, SCORE DOUBLE NOT NULL, DATUM TIMESTAMP NOT
NULL)
```

```
db2 connect reset
db2ext connect to SAMPLE user db2admin using db2admin
@rem db2ext enable database for db2image
db2ext enable table POTPISI for db2image
db2ext enable column POTPISI POTPIS for db2image
db2ext create qbic catalog POTPISI POTPIS on
db2ext open qbic catalog POTPISI POTPIS
db2ext add qbic feature QbDrawFeatureClass
db2ext add qbic feature QbColorFeatureClass
db2ext add qbic feature QbColorHistogramFeatureClass
db2ext add qbic feature QbTextureFeatureClass
db2ext set qbic autocatalog on
db2ext get qbic catalog info
@rem db2ext catalog qbic column for new
db2ext close qbic catalog
db2ext quit
@echo off
setlocal
set DB2_JDBC=C:\Program Files\SQLLIB\java\db2java.zip
set JAVA_HOME=C:\j2sdk1.4.2\bin
rem set JAVA_HOME=C:\Program Files\SQLLIB\java\jdk\bin
set PATH=.;%JAVA_HOME%;%PATH%;
set CLASSPATH=%DB2_JDBC%;%CLASSPATH%
javac Populate.java
java Populate SAMPLE db2admin db2admin
javac MyFileFilter.java
javac QbicQry.java
java QbicQry SAMPLE db2admin db2admin
endlocal
call dmbstop
```

QUERY.BAT

```
db2cmd
call dmbstart
@echo off
setlocal
set DB2_JDBC=C:\Program Files\SQLLIB\java\db2java.zip
set JAVA_HOME=C:\j2sdk1.4.2\bin
rem set JAVA_HOME=C:\Program Files\SQLLIB\java\jdk\bin
set PATH=.;%JAVA_HOME%;%PATH%;
set CLASSPATH=%DB2_JDBC%;%CLASSPATH%
java QbicQry SAMPLE db2admin db2admin
endlocal
call dmbstop
```

Nikola Lazovic and Vladan Pantovic
DB2 System Administrators
Postal Savings Bank (Serbia and Montenegro)

Project Cinnamon

IBM has recently revealed more details about its previously introduced Project Cinnamon, a set of features designed to provide graphical XML schema mapping at the administration level, ie customers will be able to automatically create the data model of a database based on the document type definitions or XML schemas they choose.

Cinnamon technology is now in beta and will ship with the next version of DB2 Content Manager. The technology is designed to help users define attributes based on XML schemas and import XML documents into the Content Manager.

In addition, Cinnamon can automate data modelling and provides a graphical XML schema mapping tool, which is designed to simplify the exporting of items from one content management system to another.

The thinking behind this is that while XML is an established standard, the schemas are still evolving. What Cinnamon offers is schema reporting and graphical modelling for users to see the schemas and establish mapping. No programming is now required.

IBM has also introduced a new graphical builder for delivering workflow based on the BPEL (Business Process Execution Language) standard. This tool is used by an administrator to describe flows of business tasks and how that information is to flow through the CM system.

Please note that the correct contact address for Xephon Inc is PO Box 550547, Dallas, TX 75355, USA. The phone number is (214) 340 5690, the fax number is (214) 341 7081, and the e-mail address to use is info@xephon.com.

Enhanced Web services support in Content Manager mean that it can now receive transaction requests in SOAP and other Web services standards.

The addition of Web services support, combined with enhanced XML capabilities, helps simplify development whether the APIs are written in .Net, Java, or use SOAP directly.

IBM also announced plans to support the emerging JSR (Java Specification Request) 170, which seeks to define a standard interface for content management systems, and increased Web services support. Support for JSR 170, which is still in draft form with the Java Community Process, is designed to simplify the process of adding content and connecting business applications to content management systems. The standard will define an element of J2EE for a content repository API.

Nick Nourse
Independent Consultant (UK)

Why not share your expertise and earn money at the same time? *DB2 Update* is looking for program code, JavaScript, REXX EXECs, JavaScript, etc, that experienced users of DB2 have written to make their life, or the lives of their users, easier. We are also looking for explanatory articles, and hints and tips, from experienced users. We would also like suggestions on how to improve DB2 performance.

We will publish your article (after vetting by our expert panel) and send you a cheque, as payment, and two copies of the issue containing the article once it has been published. Articles can be of any length and should be e-mailed to the editor, Trevor Eddolls, at trevore@xephon.com.

A free copy of our *Notes for Contributors* is available from our Web site at www.xephon.com/nfc.

Embarcadero Technologies has announced Version 3 of Embarcadero Change Manager, its cross-platform change management solution. The latest version integrates Embarcadero Change Manager with popular source code control systems.

This automation gives DBAs the control they need to ensure that their schemas are consistent with the database application code. Additionally, the product has enhancements in support of the DB2 UDB platform.

The source code control integration enables customers to store schema archives and associated DDL into Microsoft Visual SourceSafe, Serena PVCS, and IBM Rational ClearCase repositories. This integration streamlines development and deployment activities by linking the database schema to the application code accessing that same database.

Version 3.0 also offers a 'compare' facility to validate, audit, and report on database schema differences; a command-line utility that allows capture and compare jobs to be embedded in regular maintenance scripts; and job notification capabilities so DBAs can regularly receive information about their changes.

For further information contact:
Embarcadero Technologies, 100 California Street, 12th Floor, San Francisco, CA 94111, USA.
Tel: (415) 834 3131.
URL: www.embarcadero.com/products/changemanager/index.html.

* * *

Quest Software has announced new versions of Quest Spotlight on DB2 UDB and Quest Spotlight on DB2 OS/390, which help to pinpoint the source of DB2 performance problems so they can be resolved before end user service levels are affected.

The products display real-time graphical illustrations of all DB2 activity. With architecturally accurate visual representations of the DB2 environment, Spotlight enables DBAs to view all activity across the entire DB2 instance or subsystem analysing connections, wait events, locking, memory, and disk I/O to identify and alleviate problem areas as they occur. Metrics can be viewed in either real-time or historically.

For further information contact:
Quest Software, 8001 Irvine Center Drive, Irvine, CA 92618, USA.
Tel: (949) 754 8000.
URL: www.quest.com/db2/spotlight/index.asp.

* * *

Version 8.6 of the shareware product DB2CPI has recently been announced. DB2CPI is an ISPF-based DB2 Command processor interface with integrated catalog management facilities and many automated functions and utilities designed to make DBA's daily tasks easier.

DB2CPI was developed by DBAs to provide simple solutions to time-consuming tasks and provide rapid information required for problem determination in, what they describe as, an easy to interpret format.

Version 8.6 includes full DB2 Version 8 support. All utilities and screens have been changed to support the new features and limit increases in DB2 Version 8.

The product is available for download from www.db2cpi.com/default.asp.