115

DB2 update

May 2002

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 ($33.75) each including postage.

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

Editor
Trevor Eddolls

Contributions
When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £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 download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

Overview of the UDB DB2 7.2 XML Extender offering

This article introduces the UDB DB2 7.2 XML Extender offering. I will discuss the terms that you will hear/use when implementing DB2 XML Extender, and detail the different options available to you to store your XML documents. I have tried not to get bogged down in details, but to give you an overview. Once you have a grasp of the basics, then it is a lot easier to read up on the details in the *IBM DB2 Universal Database XML Extender Administration and Programming* guide (db2sxe70.pdf).

In a nutshell, the DB2 XML Extender allows you to store and retrieve XML documents in DB2 tables.

How does the DB2 XML Extender work? Well, it uses a combination of User Defined Functions (UDFs), User Defined Types (UDTs), and Stored Procedures (SP) to transfer the contents of the XML document to/from the DB2 tables. It knows how to store/retrieve data by using DTD and DAD files, which are associated with each XML document.

Now let's look at what the DTD and DAD files are.

A Data Type Definition (DTD) file is used to describe the structure of an XML document. This DTD file can be embedded in the XML document, or it can be an external file pointed to from the XML document. You can use a product such as XMLSpy (www.xmlspy.com) to generate/check a DTD file from an XML document.

You use a Data Access Definition (DAD) file to map out how the contents of the XML document are to be stored in the DB2 tables. This is called node mapping; two types are available – SQL node mapping and Remote Database (RDB) node mapping. Both of these terms are explained later when we talk about how an XML document can be stored in DB2.

You can use DB2 XML Extender to store any DTDs you may have in the table DTD_REF, and the DAD files in the XML_USAGE table. Both of these tables (and more) are created when you install the product. The advantage of storing your DTD/DAD files in this way is that it gives you a certain amount of control over your DTD/DAD files, especially as it gives an overview of which DTD/DAD files are being used by which XML document.

Other terms you will hear are ‘composing a document’, which means creating an XML document from existing DB2 data; and ‘decomposing a document’, which means transferring the contents of the XML documents to DB2 tables.

Now let’s talk about how you store XML documents in DB2. You can store the data in XMLColumn or XMLCollection format.

If you store your XML document in XMLColumn format, the entire document is stored in a single column in a DB2 table. This is the easiest way to implement DB2 XML Extender. If you want to break down the data in your XML document and store the results in separate columns, you need to use the XMLCollection format.

The XMLColumn format uses a concept called side tables. These side tables store frequently-accessed data, and can be used to speed up data retrieval. You need only a DAD file if you want to validate the data in your XML document, or if you want to build the side tables mentioned earlier. The side tables are ‘normal’ DB2 tables, so you can write standard SQL to query them.

If you store your XML document in XMLCollection format, then the contents of these document are broken down and stored in separate columns in DB2 tables. You can use either SQL node mapping or RDB node mapping to store this data. You use SQL node mapping to build XML documents from existing DB2 data. You can use standard SQL statements to specify criteria from which documents are actually composed (ie to compose documents only where the SAL column is greater than 10,000, you would have a clause such as WHERE SAL > 10,000). This is a one-way process – you cannot use SQL node mapping to decompose an XML document. You can use RDB node mapping to both compose and decompose XML documents. However, if you are composing an XML document from DB2 data and are using RDB node mapping, you cannot use SQL to filter your results.

To use the XMLCollection format you must provide a DAD file. There is a slight difference between the DAD files that you need for composing and for decomposing documents. This is detailed in the *Administration* manual.

You can tell if you are using an SQL_node DAD or an RDB_node DAD by looking in the section headed by the <Xcollection> tag. For an SQL_node DAD you will see an <SQL_stmt> tag, and for an RDB_node DAD you will see an <RDB_node> tag.

Don't forget, once you have stored your XML document in DB2 tables, you can use DB2's Text Information Extender product (free with UDB 7.2) to query the data. This is a huge feature, because it allows you to store data and run queries against it.

I have deliberately not included any examples, because the examples can get very complicated and the basic principles would be lost among the details. Once you have gone through the basic concepts, then you can look at the examples in the *Administration* manual.

To summarize, the DB2 XML Extender offering is an industrial-strength solution to the question of how to efficiently store/create XML documents from a relational database. With the addition of the text search facility (TIE) of DB2, you have a truly powerful tool, capable of meeting all your XML document storage and retrieval needs. Give it a go!

C Leonard
Freelance Consultant (UK)

DB2 level display via TSO

At times, especially when a site is running multiple DB2 releases, a user or systems programmer wants to make sure of what DB2 level his DB2 STEPLIBs are before going into SPUFI or other applications that are DB2 release dependent. The DB2LEVEL program will display the DB2 release and maintenance information that is stored in the DSN load module (DSNAA and DSNECP00, to be exact).

The program is assembled and linked into the DB2 SDSNLINK library in the LNKLST for each maintenance and accessed by users. The code provided below has been verified against DB2 V5.1 and DB2 V6.1.

```
//CSH JOB
//*
//ASM       EXEC PGM=ASMA9Ø,
//          REGION=4Ø96K,
//          PARM='NODECK,OBJECT,XREF(SHORT)'
//SYSLIB    DD DSN=SYS1.MACLIB,DISP=SHR
//          DD DISP=SHR,DSN=SYS1.AMODGEN
//SYSUT1    DD UNIT=SYSALLDA,SPACE=(17ØØ,(4ØØ,4ØØ))
//SYSUT2    DD UNIT=SYSALLDA,SPACE=(17ØØ,(4ØØ,4ØØ))
//SYSUT3    DD UNIT=SYSALLDA,SPACE=(17ØØ,(4ØØ,4ØØ))
//SYSLIN    DD DSN=&&LOADSET,
//          UNIT=SYSALLDA,DISP=(,PASS),
//          SPACE=(4ØØ,(1ØØ,1ØØ))
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
******************************************************************
*                                                                *
*   WRITTEN BY CHORNG S. (JACK) HWANG  2ØØ2 @ QUAKER OATS        *
*                                                                *
*   WRITTEN AND DISTRIBUTED AS IS, NO WARRANTIES EITHER          *
*     EXPRESSED OR IMPLIED                                       *
*                                                                *
*                        JACK HWANG    CSHWANG@HOTMAIL.COM       *
*                                                                *
******************************************************************
DB2LEVEL CSECT
DB2LEVEL AMODE 24
DB2LEVEL RMODE 24
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         USING DB2LEVEL,R15
         STM   R14,R12,12(R13)    STORE CALLER'S REGISTERS
         LR    R1,R13            SAVE CALLER'S SAVE AREA ADDRESS
         CNOP  Ø,4                FORCE FULLWORD ALIGNMENT
         BAL   R13,SPØØØ1         BRANCH AROUND SAVE AREA
SAØØØ1   DS    18F                SAVE AREA
SPØØØ1   DS    ØH
         ST    R1,4(R13)          STORE BACKWARD SA POINTER
         ST    R13,8(R1)          STORE FORWARD SA POINTER
         L     R1,24(R1)          RESTORE R1
         DROP  R15
         USING SAØØØ1,13
         TPUT  TITLE,72
         TPUT  BLANK,72
         LOAD  EP=DSNECPØØ,ERRET=DONODSN,LOADPT=LOADPT
         BAL   R6,INNARDS
         LOAD  EP=DSNAA,ERRET=DONODSN,LOADPT=LOADPT
         BAL   R6,INNARDS
*
         TPUT  D2LVL,72
*
EXIT     DS    ØH
         L     R1,4(13)           RESTORE CALLER'S SAVE AREA
         XC    16(4,1),16(1)      CLEAR R15
         L     13,4(13)           RESTORE CALLER'S SAVE AREA
         LM    14,12,12(13)       RESTORE CALLER'S REGISTERS
         BR    14                 RETURN TO CALLER
*
INNARDS  DS    ØH
         L     R3,LOADPT          ADDRESS IN R3, LENGTH IN R4
         LR    R4,R1              ADDRESS IN R3, LENGTH IN R4
         SLL   R4,8               SHIFT OFF AUTH CODE
         SRL   R4,5               SHIFT BACK TO GET LENGTH IN BYTES
         SH    R4,=H'32'          DON'T USE THAT THERE LAST 16 BYTES
PGMLOOP  DS    ØH
         CLC   Ø(2,R3),=CL2'UQ'   SEE IF PTF FOUND
         BNE   CONT1              NO, CONTINUE
         CLI   MAINTFND,C'Y'      MAINTENACE ALREADY PUT IN?
         BE    CONT1              YES, CONTINUE
         MVC   D2MAINT,Ø(R3)      MOVE MAINTENANCE LEVEL
         MVI   MAINTFND,C'Y'      SET MAINTENANCE FOUND
CONT1    DS    ØH
         CLC   Ø(15,R3),=CL15'PROPERTY OF IBM' FIND IBM HEADER
         BNE   CONT2              NO, CONTINUE
         LA    R3,15(R3)          BUMP TO PRODUCT CODE
         LA    R5,DB2V31          GET FIRST LITEREAL
PRODLOOP DS    ØH
         CLI   R5,C'*'            END?
         BE    CONT2
         CLC   Ø(8,R3),Ø(R5)      MATCH PRODUCT?
         BNE   NEXTPROD
         MVC   D2LEVEL,8(R5)      MOVE PROD LEVEL
         B     CONT2
NEXTPROD DS    ØH
         LA    R5,12(R5)          NEXT PROD
         B     PRODLOOP
CONT2    DS    ØH
         LA    R3,1(R3)           NEXT BYTE
         BCT   R4,PGMLOOP         DECREASE COUNTER BY 1
         BR    R6
         LTORG
*
DONODSN  DS    ØH
         TPUT  NODSN,72
         B     EXIT
*
TITLE    DS    ØF
         DC    CL72'********* DB2 LEVEL DISPLAY     CSH &SYSDATE *****X
               *****'
BLANK    DC    CL72' '
NODSN    DC    CL72'********* PROGRAM DSN NOT FOUND, NO DB2 LEVEL INFOX
                AVAILABLE'
D2LVL    DC    CL72'********** DB2 VERSION '
         ORG   D2LVL+23
D2LEVEL  DS    CL4
         DC    CL4' AT '
D2MAINT  DC    CL7'***BASE'
         ORG
DB2V31   DC    CL8'5685-DB2'
DB2V31L  DC    CL4'V3.1'
DB2V41   DC    CL8'5695-DB2'
DB2V41L  DC    CL4'V4.1'
DB2V51   DC    CL8'5655-DB2'
DB2V51L  DC    CL4'V5.1'
DB2V61   DC    CL8'5645-DB2'
DB2V61L  DC    CL4'V6.1'
DB2V71   DC    CL8'5675-DB2'
DB2V71L  DC    CL4'V7.1'
DB2VEND  DC    CL12'***********'
LOADPT   DS    F
MAINTFND DC    C'N'
         END
/*
//LKED      EXEC PGM=IEWL,REGION=4Ø96K,
//          PARM='LIST,XREF',COND=(7,LT,ASM)
//SYSLIB    DD DISP=SHR,DSN=SYS1.LINKLIB
//SYSLMOD   DD DISP=SHR,DSN=SYS1.DSN61Ø.SDSNLINK(DB2LEVEL)
//SYSUT1    DD UNIT=SYSALLDA,DCB=BLKSIZE=1Ø24,
//          SPACE=(1Ø24,(2ØØ,2Ø))
//SYSPRINT  DD SYSOUT=*
```

```
//SYSLIN    DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//*
//*
```

Chorng S (Jack) Hwang
Quaker Oats (USA)

A six-step strategy for deploying a Data Warehouse

SUMMARY

Decision support systems are key in enabling an enterprise to gain a competitive edge. However, in order to reap the returns on investment and to prevent costs spiralling, it is essential that the Data Warehouse or Data Marts behind every decision support system be implemented in a clearly definable and measurable way. This article outlines one approach.

THE DATA WAREHOUSE CONCEPT

Most business applications are based on the On-Line Transaction Processing (OLTP) concept. OLTP systems are designed to be application-specific. These systems process business transactions as quickly as possible, and are designed to have very low failure rates. User accessibility for the purpose of analysis is not a feature of most OLTP systems. Hence, an OLTP system designed for general ledger, order entry, or inventory control collects a veritable treasure of information, but that information is stored in a system that was never designed for access by business people.

The concept of data warehousing began more than a decade ago. The Data Warehouse (DW) is a separate repository of useful data extracted from the active operational databases populated by those OLTP business systems. The difference is that the DW is designed for easy user navigation and access. In short, the DW is designed for On-Line Analytical Processing (OLAP).

The DW solves two business problems:

- The inability of business decision-makers to gain timely access to corporate information – most OLTP systems are designed with hundreds of obscure table and field names. The design objective is one of volume and speed. Typically, end users find it difficult to navigate and locate data from an operational database. DW solves this problem by making access to, and navigation of, the data easy for the business user.
- The inconsistency of results – this arises because of the very nature of OLTP systems. These systems constantly change content as new data is added or existing data is manipulated. Database administrators will rightly tell you that accessing the active operational database is bad because it affects performance. The truth is that accessing the active operational database for reporting is as bad for the end user as it is for the DBA. The reason for this is that running the same query at different times will yield different results. These differing results defeat the main objective of Data Mining – that of everybody viewing the data from the same moment in time. DW solves this problem by providing an independent and stable point-in-time copy of the data for mining purposes.

THE DATA MART

While the DW is a repository for all the useful data within an enterprise that may be required by any OLAP business user, a Data Mart (DM) contains a subset of information specific to a business problem or function, eg all the data related to sales transactions. The DM will then be used as the basis for answering specific business queries. An enterprise will need to decide whether to opt for the Data Warehouse concept or that of the Data Mart. Data Marts are typically less expensive to implement because the data volumes and implementation time are greatly reduced. However, they limit the use to a predefined subset of data.

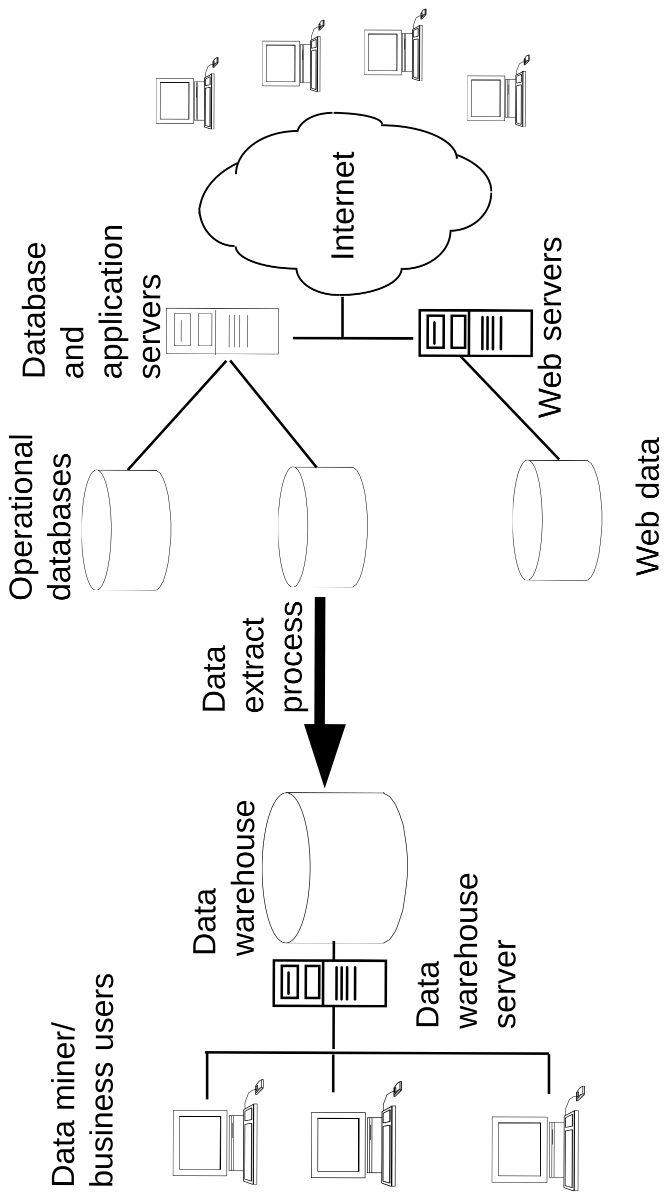


Figure 1: Sample architecture – centralized model

DATA WAREHOUSE ARCHITECTURE

The DW architecture can vary depending on whether a centralized model or distributed model is implemented. A centralized DW model architecture example is shown in Figure 1.

The distributed model would involve extracting data from a centralized Data Warehouse, and populating a Data Mart residing on the business user's personal computer.

SIX STEPS TO DEPLOYING A DATA WAREHOUSE

Once an enterprise has determinedwhether it needs a Data Warehouse (DW) or Data Mart (DM), the six steps for deployment are fairly straightforward and are essentially the same for both. The steps are as follows.

Step 1 – define the objectives and business problem

It is important to have very clearly defined and measurable objectives prior to deploying a DW/DM. Before one can define the objectives, one must first understand the business problem.

It is very important that the project sponsor understands exactly what a DW/DM can and cannot do. Problems and objectives that are too loosely defined, such as the objective to invent new services, are not sufficiently concrete to yield worthwhile results. A better objective is perhaps to improve understanding of our customers' requirements. By using the DW/DM to store customer-related data, one could employ data mining techniques to provide the initial deeper understanding of our customers' requirements. The results of this mining would be given to market researchers, who would analyse this information and, by recognizing customers' needs, develop new service offerings.

The objectives need to take into account the intended audience of the DW and/or scope of the DM. Furthermore, objectives need to be documented and agreed with the DW/DM project sponsor prior to commencing the deployment.

DW/DM deployment should be treated and managed as a separate project. This will ensure responsibility and accountability, and should ensure that costs are kept under control and remain within budget.

Step 2 – identify the data sources

Once our objectives are clearly defined, the data sources must be identified. Data is typically spread across multiple systems and/or servers, although this is not always the case. The data required may be only a subset of the active operational data. Identifying the data sources will aid in selecting the most appropriate toolset.

Step 3 – select the tools

There are many tools from which to choose. Selecting the most appropriate toolset is an important aspect of the project because it forms the backbone of the DW/DM. The most appropriate toolset is dependent on the objectives of the DW/DM and the data sources from where data is to be extracted. Needless to say, the budget will also play a major part in the selection process.

Step 4 – design and build the Data Warehouse/Data Mart

Designing the DW/DM will consume about 75% of the project's time and this is arguably the most important step of all. The more thought that goes into the design, the easier the DW/DM will be to use. A simple DW design may be just a complete copy of the operational database, with periodic data extracts and loads from the active operational database to the DW. A more complex design may involve a data model that only remotely resembles the operational data model. Special consideration must be given to the window of time available in which the DW/DM can be populated. Operational database systems typically have high availability requirements, and it is important that the process of extracting data from the operational database does not adversely impact the availability of these systems.

Step 5 – prepare the data

Data preparation is key to a successful result. As the cliché goes, garbage in, garbage out. It is at this stage that serious consideration must be given to data quality. Data must be extracted and then ‘cleaned’ to ensure data consistency. Moreover, some aspects of the data will need to be converted or reformatted in order to be more understandable to the business user. Once the data is prepared, the DW/DM can be populated. It is important that sufficient time be invested in this activity. Do not underestimate the time that may be required for this step.

Step 6 – mine the data/solve the business problem

The final step involves mining the DW/DM, with the purpose of solving the business problem identified in Step 1. When data mining, it is important for the data miner to concentrate on the aggregated results and not individual records. Also, the data miner should keep track of the mining procedure and results. This will enable the data miner to reproduce the same results at a later stage.

CONCLUSION

Implementing a Data Warehouse or Data Mart is a relatively straightforward task provided the project has both an owner and a sponsor, and is given sufficient resources and commitment.

The six-step strategy, outlined in this article, highlights the main aspects of deploying a Data Warehouse or Data Mart.

If, as suggested, the business problem is fully understood and the objectives of the system are measurable and clearly defined, the business should reap the benefits of the investment.

Return on investment can typically be expected within the first six months of deployment. By giving the project a budget and assigning both responsibility and accountability for its deployment, an organization will guard against spiralling costs.

TERMINOLOGY

Terminology used in this article includes:

- Business Intelligence (BI) – BI is the analytic process of transforming raw data into business information that can be used to increase a company's efficiency and profitability. It is an enabling process with the purpose of solving key business problems.

- Data Mart (DM) – a business-focused Data Warehouse designed for a particular business issue.
- Data Mining – a method for determining patterns in an enterprises business operation.
- Data Warehouse (DW) – a repository, separate from the active operational database, used to store business information for the purpose of OLAP systems and Data Mining.
- Decision Support System – a system that provides business users with the required information to assist them in making decisions for their business.
- On-Line Analytical Processing (OLAP) system – a system that provides a business user, or enterprise, with fast access to and unlimited analytical views of business information repositories for the purpose of solving key business problems. OLAP systems are designed to handle large or long *ad hoc*/dynamic queries that may result in large results tables.
- On-Line Transaction Processing (OLTP) system – a system designed to be application-specific, such as order entry, invoicing, inventory control or customer service. OLTP systems are designed to process large volumes of small transactions very fast.

Stephen B Forward
Consultant
Coveford Data Services (Switzerland)

Have you come across any undocumented features in DB2 Version 7? Why not share your discovery with others? Send your findings to us at any of the addresses shown on page 2.

A crash course for DB2 UDB for OS/390 certification

This article describes some recommendations for DB2 mainframe DBAs who plan to become IBM Certified Solutions Experts – DB2 UDB Database Administrator for OS/390. I passed Exam 516 at the recent IDUG 2001 Conference in Florence, Italy.

I have collected in this article some information for all six sections. Of course, you must have enough basic and practical experience with DB2.

CREATE AND MANAGE DB2 OBJECTS

Identify the characteristics of an index, view, or table

Tables are logical structures maintained by DB2. Tables are made up of columns and rows. Every table must have one or more columns, but the number of rows can be zero.

Types of table include:

- Base table – a table created with CREATE TABLE and used to hold persistent user data.
- Auxiliary table – a table created with CREATE AUXILIARY TABLE and used to hold the data for a column that is defined in a base table.
- Temporary table – a table defined with CREATE GLOBAL TEMPORARY TABLE and used to hold temporary data.
- Result table – a set of rows that DB2 selects or generates from one or more base tables.
- Empty table – a table with zero rows.
- Sample table.

An index is an ordered set of pointers to rows of a base table or an auxiliary table. Each index is based on the values of data in one or more columns. An index is an object that is separate from the data in the table. When you define an index using the CREATE INDEX statement, DB2 builds this structure and maintains it automatically. Indexes can be used by DB2 to improve performance and ensure uniqueness. In most cases, access to data is faster with an index. A table with a unique index cannot have columns with identical rows.

A view provides an alternative way of looking at the data in one or more tables. A view is a named specification of a result table. The specification is an SQL SELECT statement that is effectively executed in an SQL statement. The columns added to the base tables after the view is defined do not appear in the view. For retrieval, all views can be used like base tables.

Interpret the contents of system catalog and directory

The DB2 catalog consists of tables of data about everything defined to the DB2 system, including table spaces, indexes, tables, copies of table spaces and indexes, storage groups, etc. The system database, DSNDB06, contains the DB2 catalog. When you create, alter, or drop any structure, DB2 inserts, updates, or deletes rows of the catalog that describe the structure and how it relates to other structures. The SYSIBM.SYSTABLES is one catalog table that records information when a table is created.

The communications database (CDB) is part of the DB2 catalog.

The DB2 directory contains information that DB2 uses during normal operation. You cannot access the directory using SQL. The structures in the directory are not described in the DB2 catalog. The directory consists of a set of DB2 tables stored in five table spaces in system database, DSNDB01. The SCT02 skeleton cursor table space is created when you bind a plan, and the SPT01 skeleton package table space is created when you bind a package.

Create and alter tablespaces

The create tablespace statement defines a simple, segmented, or partitioned table space.

Sample create tablespace statement:

```
CREATE TABLESPACE TS1 IN DB1
       PRIQTY 8Ø
       SECQTY 4Ø
       ERASE NO
       FREEPAGE 1Ø
       PCTFREE 5
       BUFFERPOOL BP1
       LOCKSIZE ANY
       SEGSIZE 64
       CLOSE NO
```

The alter tablespace statement changes the description of a table space.

An example alter tablespace statement looks like:

```
ALTER  TABLESPACE DB1.TS1
       BUFFERPOOL BP2
       LOCKSIZE PAGE
```

Drop objects

The drop statement deletes an object on the current server. Except for storage groups, any objects that are directly dependent on that object are deleted. Whenever an object is deleted, its description is deleted from the catalog on the current server, and any plans or packages that refer to the objects are invalidated.

Add and drop columns and change characteristics in a table

The ALTER TABLE *table-name* ADD *column-definition* statement adds a new column to the table. Some changes to a table (drop table) cannot be made with an ALTER TABLE statement. For example, a column defined as SMALLINT must be redefined to INTEGER. To make such changes, you need to perform the following steps:

- Unload the table
- Drop the table
- Commit the changes
- Re-create the table

- Re-load the table.

The ALTER TABLE statement changes characteristics in a table. Some valid options in the ALTER TABLE statement are: add or drop unique constraint, referential integrity, check constraint, drop primary key, data capture, restrict on drop, etc.

Create aliases and synonyms for tables

The CREATE ALIAS statement defines an alias for a table or view. The definition is recorded in the DB2 catalog. The table or view does not have to be described in that catalog.

```
CREATE ALIAS MYPLAN FOR LOKNADI.SYSIBM.SYSPLAN
```

The CREATE SYNONYM statement defines a synonym for a table or view:

```
CREATE SYNONYM TABLES FOR SYSIBM.SYSTABLES
```

Activities associated with enabling stored procedures (Stored Procedure Builder)

The DB2 Stored Procedure Builder, an element of the DB2 Management Clients Package, provides an easy-to-use development environment for creating, installing, and testing stored procedures. With the DB2 Stored Procedure Builder, you can develop stored procedures on one operating system and deploy them on another. The DB2 Stored Procedure Builder supports two commonly-used languages for stored procedures: SQL procedures language and Java. The DB2 Stored Procedure Builder perform a variety of tasks that are associated with stored procedures, such as:

- Viewing existing stored procedures
- Modifying existing stored procedures
- Creating new stored procedures
- Running existing stored procedures.

The DB2 Stored Procedure Builder requires DB2 Connect.

Create all forms of referential integrity and determine the effects of each

Referential integrity is the state in which all values of all foreign keys for a given DB2 are valid. A referential constraint is the rule that the non-null values of a foreign key are valid only if they also appear as values of a parent key.

The following rules provide referential integrity:

- Insert rule or update rule – a non-null insert/update value of the foreign key must match some value of the parent key of the parent table.
- Delete rule – the choices when the referential constraint is defined are RESTRICT, NO ACTION, CASCADE, or SET NULL. SET NULL can be specified only if some column of the foreign key allows null values. The delete rule of the referential constraint applies when a row of the parent table is deleted. If the delete rule is:
  - RESTRICT or NO ACTION, an error occurs and no rows are deleted.
  - CASCADE, the delete operation is propagated to the dependent tables.
  - SET NULLS, each nullable column of the foreign key is set to null.

Given a DDL SQL statement, identify the results

The CREATE TABLE is a DDL SQL statement, and you need to identify the results. The definition must include its name and attributes of its column. The definition can include other attributes of the table, such as its primary key and its table space.

Example:

```
CREATE TABLE MYTAB1 (ID      INTEGER   NOT NULL
                    ,NAME    CHAR(25)  NOT NULL
                    ,SALARY  DECIMAL(7,2)) IN DATABASE DSNDBØ4
```

Manage storage allocation, for example using VSAM DELETE/ DEFINEs or STOGROUPs

To create a DB2 storage group, use the SQL statement CREATE STOGROUP.

Example:

```
CREATE STOGROUP MYSG1 VOLUMES(MVSDB7) VCAT(catalog-name)
```

Storage from the identified volumes can later be allocated for table spaces and index spaces. When you create table spaces and indexes, you name the storage from which you want space to be allocated.

To manage DB2 auxiliary storage yourself, you use access method service. To define the required datasets, use DEFINE CLUSTER, and to delete datasets, use DELETE CLUSTER. You can define simple, segmented, or partitioned table spaces.

Example:

```
DEFINE CLUSTER                                        -
       (NAME(catname.DSNDBC.dbname.tsname.IØØØ1.AØØ1)) -
        LINEAR                                        -
        REUSE                                         -
        VOLUMES(MVSDB7)                               -
        RECORDS(1ØØ 1ØØ)                              -
        SHAREOPTIONS(3 3)                             -
       DATA                                           -
       (NAME(catname.DSNDBD.dbname.tsname.IØØØ1.AØØ1)) -
        CATALOG(catname)
```

The use of DEFER DEFINE

DEFER DEFINE indicates whether the index is to be built during the execution of the CREATE INDEX statement. If the table is not empty and DEFER YES is specified, the index is placed in a rebuild pending status and a warning message is issued; the index must be rebuilt by the REBUILD INDEX utility.

Methods for data validation – check constraints, triggers, UDFs

VALIDPROC, an option in the CREATE TABLE statement, is a validation routine that receives an entire row of a base table as input, and can return an indication of whether or not to allow a following INSERT, UPDATE, or DELETE operation.

A check constraint is a rule that specifies the values allowed in one or more columns of every row of a table. DB2 enforces the constraints when a row is inserted, updated, or loaded with the LOAD utility (ENFORCE CONSTRAINTS).

A trigger defines a set of actions that are executed when a delete, insert, or update operation occurs on a specified table. When such an SQL operation is executed, the trigger is said to be activated.

A user-defined function (UDF) is an extension to the SQL language. A user-defined function is similar to a host language subprogram or function. A user-defined function is often the better choice for an SQL application because it can be invoked in an SQL statement.

DATA RECOVERY

Steps to prepare for DB2 disaster recovery

There are several levels of preparation for disaster recovery:

- Prepare the recovery site to recover to a fixed point in time. For example, you could copy everything weekly with a DFSMSdss volume dump (logical) and manually send it to the recovery site, then restore the data there.
- For recovery through the last archive, copy and send the following objects to the recovery site as you produce them:
  - Image copies of all catalog, directory, and user page sets.
  - Archive logs.
  - Integrated catalog facility catalog back-ups.
  - DSBS lists.
  - DB2 libraries.

The different object status (eg RECP, GRECP, LPL, RESTP)

The DB2 command DISPLAY DATABASE displays information about the status of DB2 databases, tablespaces etc. Some of these statuses are:

- RECP displays objects that are in RECOVER pending status.
- LPL displays logical page list entries.
- GRECP displays objects that are in group buffer pool RECOVER pending status.
- REORP displays objects that are in REORG pending status.
- STOP displays objects that are stopped.
- UT displays objects that are in utility access mode.

Back-up and recovery considerations

The back-up and recovery procedures are critical in order to avoid costly and time-consuming losses of data. You should develop procedures to:

- Create a point of consistency.
- Restore system and data objects to a point of consistency.
- Back up the DB2 catalog and directory and your data.

DB2 can recover a page set by using a back-up copy or the recovery log or both. The DB2 recovery log contains a record of all changes made to the page set. If DB2 fails, it can recover the page set by restoring the back-up copy and applying the log changes to it from the point of the back-up copy.

PERFORMANCE AND TUNING

Concurrency considerations

Concurrency is the ability of more than one application process to access the same data at essentially the same time. Concurrency must be controlled to prevent lost updates and such possibly undesirable effects as unrepeatable reads and access to uncommitted data. To prevent those situations from occurring, unless they are specifically allowed, DB2 might use locks to control concurrency. The effects of locks that you want to minimize are suspension, timeout, and deadlock:

- Suspension – an application process is suspended when it requests a lock that is already held by another application process and cannot be shared. The suspended process temporarily stops running
- Timeout – an application process is said to time out when it is terminated because it has been suspended for longer then a preset interval
- Deadlock – a deadlock occurs when two or more application processes each hold locks on resources that the other needs and without which they cannot proceed

The basic recommendations to promote concurrency are:

- Recommendations for DB2 system options.
- Recommendations for database design.
- Recommendations for application design.

The effect of COMMIT frequency

The COMMIT statement ends a unit of recovery and commits the relational database changes that were made in that unit of recovery. If relational databases are the only recoverable resources used by the application process, COMMIT also ends the unit of work. A commit point occurs when you issue a COMMIT statement or your program terminates normally. You should issue a COMMIT statement only when you are sure the data is in a consistent state.

Convert logical design to physical design

The conversion from logical design (data model) to physical design has the following steps:

- Creating a database, which is a logical collection of table spaces and index spaces.

- Creating table spaces, which are the physical spaces that hold tables.

It is possible to implement table spaces explicitly with the CREATE TABLESPACE statement or implicitly with the CREATE TABLE statement.

Analyse EXPLAIN/VISUAL EXPLAIN information

Explain is a monitoring tool that produces information about the following:

- A plan, package, or SQL statement when it is bound. The output appears in a plan table.
- An estimated cost of executing an SQL statement. The output appears in a DSN_STATEMENT_TABLE, which is also called a statement table.
- User-defined functions referred to in the statement, including the specific name and schema. The output appears in a DSN_FUNCTION_TABLE, which is also called a function table.

Visual explain is a graphical workstation feature of DB2 that provides:

- An easy-to-understand display of a selected access path.
- Suggestions for changing an SQL statement.
- An ability to invoke EXPLAIN for dynamic SQL statements.
- An ability to provide DB2 catalog statistics for referenced objects of an access path.

Capture and analyse DB2 trace data

You can use DATA CAPTURE CHANGES on the CREATE TABLE or ALTER TABLE statement to have data changes to that table written to the log in an expanded format. You can then retrieve the log by using a DB2 DataPropagator.

DB2 trace allows you to trace and record subsystem data and events. There are four different types of trace data:

- Statistics – data that allows you to conduct DB2 capacity planning and to tune the entire set of DB2 programs.
- Performance – data about subsystem events that can be used to do program, resource, user, and subsystem-related tuning.
- Audit – data that can be used to monitor DB2 security and access to data.
- Monitor – data that is available for use by DB2 monitor application programs.

The difference between dynamic and static SQL

Static SQL statements in a source program must be processed before the program is compiled. The DB2 precompiler checks the syntax of the SQL statements, turns them into host language comments, and generates host language statements to invoke DB2.

Programs that contain embedded dynamic SQL statements must be precompiled like those that contain static SQL; but unlike static SQL, the dynamic statements are constructed and prepared at run time. The source form of a dynamic statement is a character string that is passed to DB2 by the program using the static SQL statement PREPARE or EXECUTE IMMEDIATE.

Choose correct indexing columns and clustering index

Each index is based on the values of data in one or more columns of a table. After you create an index, DB2 maintains it, but you can perform necessary maintenance – such as reorganizing it or recovering the index. Indexes can provide efficient access to data. In fact that is the only purpose of non-unique indexes. Unique indexes have the additional purpose of ensuring that values are unique.

DB2 tries to store data rows in data pages in the sequence of the clustering index. A clustering index optimizes the performance of an SQL statement that accesses subsets of a table in the sequence of the clustering index. When defining the clustering index, the most important criteria are:

- The clustering index must reflect the sequence needed for batch processes that access the table.
- Most online applications are not affected by the choice of a clustering index.

Selecting the column sequence in the index could impact the ability to avoid sorts and the number of matching columns. The number of matching predicates is maximized if you choose all the columns that have '=' operator as the first character. When several columns are always used with an '=' operand, include the column with the highest cardinality first.

Determine correct index characteristics (freepage, pctfree, etc)

The PCTFREE clause specifies what percentage of each page in an index space is left free when loading or reorganizing the data. DB2 uses the free space later on when you insert or update data. When no free space is available, DB2 holds your additional data in another page. When several records are physically located out of sequence, performance suffers.

The FREEPAGE clause specifies how often DB2 leaves a full page of free space when loading data or reorganizing indexes.

Use PCTFREE if inserted rows are distributed evenly and densely across the key or page range. Use FREEPAGE if inserts are concentrated in small areas of the table spaces or indexes.

Determine whether compression should be used

You can compress data in a table space by specifying COMPRESS YES on CREATE TABLESPACE. When you compress data, bit strings that occur frequently are replaced by shorter strings. Before compressing data, you can use the DSN1COMP stand-alone utility to estimate how well it will compress. After data is compressed, use compression reports and catalog statistics to determine how effectively it was compressed (PAGESAVE column in SYSIBM.SYSTABLEPART and PCTROWCOMP column in SYSIBM.SYSTABLES).

Determine when to REORG

Execute the REORG utility periodically to maintain adequate performance for your applications. The REORG utility reorganizes a table space in the clustering sequence and reclaims fragmented space. You can determine when to run REORG for table spaces and indexes by using OFFPOSLIMIT, INDREFLIMIT catalog query options. If you specify the REPORTONLY option, REORG will produce a report if a REORG is recommended.

Return code 1 means no REORG recommended and return code 2 means that a REORG is recommended.

Alternatively, information from SYSTABLEPART and SYSINDEXPART tables (columns FAROFFPOS and NEAROFFPOS) can tell you which table spaces and indexes qualify for reorganization.

BIND options

Bind plan/package builds an application plan/package. DB2 records the description of the plan/package in the catalog tables and saves the prepared plan/package in the directory. Some BIND options are:

- The ACQUIRE and RELEASE options of bind operations determine when DB2 locks an object your application uses and when it releases the lock. Recommendation is ACQUIRE (USE) and RELEASE(DEALLOCATE).
- The ISOLATION option determines how far to isolate an application from the effects of other running applications. The valid options are RR, RS, CS, UR and NC.
- The VALIDATE (RUN) indicates that, if not all objects exist at bind time, the process issues warning messages, but the bind succeeds. The VALIDATE (BIND) indicates that DB2 checks all objects at bind time.
- The EXPLAIN (YES) option indicates what SQL explain information will be stored in a plan table.

Determine the benefits of data sharing

DB2 data sharing provides improved price performance and availability, and extends the processing capacity of your system. DB2 data sharing gives you a database solution that is powerful enough to handle complex business requirements, but can run on one or more smaller, less expensive System/390 microprocessors. These new types of microprocessor enable you to increase the number of available processors at a lower cost.

Describe considerations for distributed data including security, backup, and recovery

In a distributed data environment, DB2 applications can access data at many different DB2 sites and at remote relational systems. DB2 applications can use DDF to access data at other DB2 sites and at remote relational database systems that support DRDA. DRDA is a standard for distributed connectivity.

You must define a VTAM application, using the SECACPT option, which determines a level of conversation security. Use ALREADYV for the most flexibility in determining your security.

Considerations for recovering distributed data

Using distributed data, no matter where a unit of work originates, the unit is processed as a whole at the target system. On a DB2 server, the entire unit is either committed or rolled back. However, DB2 provides no special means to coordinate recovery between different subsystems even if an application accesses data in several systems. To guarantee consistency data between systems, write your applications, as usual, to do all related updates within one unit of work.

The features that enable 7x24 availability

7x24 availability allows two processes to update the same page simultaneously at any time of the day or night.

The uses of the REPAIR utility

The REPAIR utility repairs data. If the data is invalid, REPAIR replaces it with valid data. Be extremely careful using the REPAIR utility. You can use the REPAIR utility to:

- Test DBDs.
- Repair DBDs.
- Reset a pending status on table spaces or indexes.
- Verify the contents of data areas in table spaces and indexes.
- Replace the contents of data areas in table spaces and indexes.
- Delete a single row from a table space.
- Produce a hexadecimal dump of an area in a table space or index.
- Delete an entire LOB from a LOB table space.
- Dump LOB pages.
- Rebuild OBDs for a LOB table space.

Before starting to use REPAIR to change data, it may be useful to have a copy (full image copy or DSN1COPY) of the affected table space to enable fallback.

Use of various DB2 commands: display, start, stop, alter, recover, term utility

The DB2 DISPLAY command will display:

- Information about archive log processing.
- Information about the buffer pools.
- Information about the data sharing.
- Information about DB2 threads.
- Status information about DB2 databases.

- Status information about DDF.
- Status information about a DB2 utility.
- Statistics about user-defined functions and stored procedures.
- Log information.

The DB2 START command will:

- Make the database available for use.
- Start DB2 subsystem.
- Start DDF.
- Start DB2 trace.
- Start user-defined functions and stored procedures.

The DB2 STOP command will:

- Make the database available for applications.
- Stop a DB2 subsystem.
- Stop DDF.
- Stop DB2 trace.
- Stop user-defined functions and stored procedures.

The DB2 ALTER command will:

- Alter attributes for the buffer pools.
- Alter parameter values of the REORG utility.
- Close a current active log and open the next available log dataset.

The DB2 RECOVER command will:

- Re-establish dual bootstrap datasets.
- Recover threads left indoubt.

The DB2 TERM utility terminates execution of a utility.

Features of the UNLOAD utility

With the UNLOAD utility you can unload data from a table space or an image copy dataset. The UNLOAD utility is faster than the DSNTIAUL sample program and REORG UNLOAD EXTERNAL. The UNLOAD utility performs the following tasks:

- Unload data from an image copy dataset.
- Unload data from multiple partitions in parallel.
- Select data by using a syntax similar to the SQL SELECT statement.
- Use field selection, ordering, and formatting options.

Using stand-alone utilities

The stand-alone utilities can be run only in an MVS JCL environment. Most of the stand-alone utilities can be used while DB2 is running. The table spaces and index spaces must be stopped first because these utilities do not have access to the DB2 buffer pools. A stand-alone utility job stream requires that you code specific dataset names in the JCL. The stand-alone utilities are:

- DSN1CHKR verifies the integrity of DB2 catalog and directory.
- DSN1COMP estimates space savings (recommendation for compressing data).
- DSN1COPY copies VSAM or sequential datasets.
- DSN1LOGP formats the contents of the recovery log for display.
- DSN1PRNT prints DB2 VSAM datasets, image copy datasets, or sequential datasets.
- DSN1SDMP forces dump when DB2 trace events occur.
- DSNJU003 changes the bootstrap datasets.

The functions of the Control Center

The DB2 Control Center is a graphical interface for administering database objects for the Universal Database family. You can use the Control Center as your main point of administration to manage systems, DB2 instances, and database objects, such as tables, views, and user group. You can also use the Control Center to access DB2 for OS/390 subsystems.

The run-time considerations for utilities

Use the DB2 DISPLAY UTILITY command to check the current status of online utilities. A utility can have one of these statuses:

- Active – the utility has started execution.
- Stopped – the utility has abnormally stopped execution before completition.
- Terminated – the utility has been terminated by DB2 TERM UTILITY.

Restarting a failed utility

If a utility finishes abnormally, it is often possible to restart it. With the restart procedure, you avoid repeating much of the work that had already been done. Two different methods of restart are available:

- Phase restart can be done from the beginning of the phase that was being processed.
- Current restart can be done from the last checkpoint taken during the execution of the utility phase. If the utility phase does not take any checkpoints or has not reached the first checkpoint, current restart is equivalent to phase restart.

When it is necessary to run RUNSTATS

You must execute RUNSTATS to keep the correct statistics in the DB2 catalog tables, thereby enabling optimal performance for applications. A general recommendation is to run RUNSTATS after:

- LOAD, REORG TABLESPACE, or REORG INDEX.
- Creating a new index.

- Extensive insert, delete, or update activity.

The RUNSTATS utility collects the statistics by scanning the target table space or index and then updates the DB2 catalog tables.

Considerations for using online REORG

The REORG TABLESPACE utility reorganizes a table space to improve access performance. By specifying the SHRLEVEL CHANGE option you can access the data during most phases of the REORG utility. Before running the REORG utility with SHRLEVEL CHANGE, you must create a mapping table and an index for it.

The implications of the TERM UTILITY command

The DB2 command TERM UTILITY terminates execution of a DB2 utility job step and releases all resources associated with the step. In some cases, terminating a utility job can leave work in an undesirable state, requiring special processing before the job can be resubmitted. The following list describes some of the effects of the TERM utility:

- CHECK DATA – table spaces remain in CHECK-pending status.
- RECOVER – places the object being rebuilt in REBUILD-pending status.

The different LOAD options

The LOAD utility loads records into the tables and builds any indexes defined on them. If the table space already contains data, you can choose whether you want to add the new data to the existing data or replace the existing data. Some important load options are:

- RESUME YES loads records into a non-empty table space.
- SHRLEVEL CHANGE specifies that applications can concurrently read and write the table space or partition it is loading.
- REPLACE option replaces all existing rows of all tables in the table space, not just those of the table you are loading.

DB2I

The DB2I Primary Option Menu allows you the following DB2 functions:

- SPUFI executes SQL statements.
- DCLGEN generates SQL and source language declarations.
- PROGRAM PREPARATION prepares a DB2 application program to run.
- BIND/FREE performs BIND, REBIND, or FREE plans or packages.
- RUN runs an SQL program.
- DB2 COMMANDS issues DB2 commands.
- UTILITIES invokes DB2 utilities.

Use of the MODIFY utility

The MODIFY utility is a cleaning utility with the RECOVERY option; it deletes records from the SYSIBM.SYSCOPY catalog table, related log records from the SYSIBM.SYSLGRNX directory table, and entries from the DBD. The phases for MODIFY RECOVERY are initialization and set-up, deleting image copy rows from DB2 catalog, and clean-up. You can, with the DELETE AGE *integer-number* option, delete all SYSCOPY records older than a specified number of days, or you can use DELETE DATE *integer-date* option, which deletes all records written before a specified date.

Describe the function of the CHECK DATA/INDEX/LOB utility

The CHECK DATA utility checks table spaces for violations of referential and table check constrains, and reports information about violations that are detected. Run CHECK DATA periodically on all table spaces where parent and dependent tables might not be synchronized. CHECK DATA optionally deletes rows that violate referential or table check constraints. A row that violates one or more constraints is copied, once, to an exception table.

The CHECK INDEX utility tests whether indexes are consistent with the data they index, and issues warning messages when an inconsistency is found.

The CHECK LOB utility can be run against a LOB table space to identify any structural defects in the LOB table space and any invalid LOB values.

SECURITY

The use of GRANT and REVOKE

The GRANT statement grants privileges to authorization IDs. You can define privileges on a collection, database user-defined function, stored procedure, plan, package, schema, system, table, or view. The grants are recorded in the DB2 catalog.

The REVOKE statement revokes privileges from authorization IDs and deletes grants from the DB2 catalog.

Bernard Zver
DBA
Informatika (Slovenia)

The ‘top ten’ problem

A common problem faced by application developers is the requirement to retrieve a limited number of qualifying rows from the database – for example, to select the ten largest deposits or the five smallest balances. The first reaction of most programmers is to simply use the WHERE clause to eliminate non-qualifying rows. But this is simplistic, and often is not sufficient to produce the results desired in an optimal manner.

For example, what if the program requires that *only* the top ten results be returned? This can be a difficult request to formulate using SQL alone. Consider, for example, an application that needs to retrieve only the top ten highest paid employees from the EMP table in the DB2 sample database. You could simply issue a SQL request that retrieves all the salaries in order, but use only the first ten retrieved. That is easy, for example:

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN871Ø.EMP
ORDER BY SALARY DESC;
```

But that does not satisfy the requirements of retrieving *only* the top ten. It merely returns every row with the results sorted into descending sequence. So the results would still be all employees in the table, but in the correct order so you can view the ‘top ten’ salaries easily. When using this approach be sure to specify the ORDER BY clause with the DESC key word. This sorts the results into descending order, instead of the default, which is ascending. Without the DESC keyword, the top of the list would contain the lowest paid employees and the top ten would be at the very end of the results set.

For many users this approach of ordering the desired results to the top may be sufficient. But it is not a complete solution that meets the specifications of returning only the top ten using only SQL. The ideal solution should return *only* the top ten employees with the highest salaries. There are several DB2 solutions that can be used to produce this result. Of course, you could implement the SQL in a cursor and programmatically return only the first ten rows. But this would require host language programming skills and an application program would need to be run each time the results are required.

DB2 Version 7 provides an easy way to limit the results of a SELECT statement using a new clause – the FETCH FIRST *n* ROWS clause. When the FETCH FIRST *n* ROWS clause is specified, DB2 will limit the number of rows that are fetched and returned by a SELECT statement. This Version 7 approach requires SQL only and is quite simple and efficient. The FETCH FIRST *n* ROWS ONLY clause is appended right to the end of the SELECT statement. It is used as follows:

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN871Ø.EMP
ORDER BY SALARY DESC
FETCH FIRST 1Ø ROWS ONLY;
```

Of course, the value can be any number – not just 10. For example, to retrieve only the top four salaries you would code:

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN871Ø.EMP
ORDER BY SALARY DESC
FETCH FIRST 4 ROWS ONLY;
```

This is the simplest and probably the most elegant solution for limiting the number of rows returned by a DB2 query. This clause is different from the OPTIMIZE FOR *n* ROWS clause that has been available for several releases of DB2 now. The FETCH FIRST *n* ROWS ONLY clause will limit the number of rows returned to the specified number, *n*. Remember that the OPTIMIZE FOR *n* ROWS clause does not impact the number of rows returned, but is used only by the optimizer for optimizing SQL.

But the FETCH FIRST *n* ROWS ONLY clause requires you to be running at least DB2 Version 7 and you might not be running at that level. In that case, another approach is required. One approach is to use the COUNT function and some 'tricky' SQL. The following SQL will also return the top ten employees by salary:

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY

FROM DSN871Ø.EMP A
WHERE 1Ø > (SELECT COUNT(*)
FROM DSN871Ø.EMP B
WHERE A.SALARY < B.SALARY
AND B.SALARY IS NOT NULL)
ORDER BY SALARY DESC;
```

Let’s break this query down into components to understand how it works. This SQL statement uses a correlated subquery. A correlated subquery is an inner query that must be evaluated for *each* row of the outer query. The query uses aliases to identify the table references. The alias ‘A’ identifies the table in the outer query, so that in the subquery, the A.SALARY identifies that column as belonging to the outer query’s table. The alias ‘B’ is used for the subquery table (although it is not required).

So, for each row of the outer query, the subquery counts the number of rows with a larger score than that of the outer row under consideration. If there are fewer than 10 rows with a larger score, then the outer row satisfies the outer WHERE clause – in other words, it belongs in the top ten.

The ORDER BY clause is required to sort the results in the right order. If it is removed from the query, the results will still contain the top ten, but they may be in no particular order. Additionally, this query works for all DB2 versions and platforms (mainframe, Unix, and NT) and it is portable from DB2 to other database servers, such as SQL Server and Oracle. And, of course, you can change the constant 10 to any number you wish, thereby retrieving the top 20, or top five, as deemed necessary by the needs of your application.

That does not mean to suggest that this query is without problems – indeed, it can be quite inefficient. This particular SQL statement uses a correlated subquery. The number of rows counted will grow exponentially as the number of rows in the table grows. It can be quite inefficient when the number of rows grows to as little as a thousand. For very small amounts of data, though, this query usually performs quite well.

But there is another difference between this query and the previous one – and that is the way that ‘ties’ are handled. A tie occurs when more than one row contains the same value. This query may return more than 10 rows if there are multiple rows with the same value for salary within the top ten. The previous query that used the FETCH FIRST *n* ROWS

ONLY clause will always limit the number of rows returned to *n*, even if there are other rows with the same value for price as the number *n* row in the results set. The needs of your application will dictate whether ties are to be ignored or included in the result set. If ties should not be included in the results set, do not use the last SQL formulation because it will include ties.

One final approach to the top ten problem is the brute force method. This method relies on systematically identifying the largest value yet to be returned. It works like peeling back the layers of an onion. The following example uses the brute force method to return the top ten salaries from the EMP table:

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN8710.EMP
WHERE SALARY = (SELECT MAX(SALARY) FROM DSN8710.EMP)
UNION ALL
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN8710.EMP
WHERE SALARY =
(SELECT MAX(SALARY) FROM DSN8710.EMP
WHERE SALARY < (SELECT MAX(SALARY) FROM DSN8710.EMP))
UNION ALL
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN8710.EMP
WHERE SALARY =
(SELECT MAX(SALARY) FROM DSN8710.EMP
WHERE SALARY < SELECT MAX(SALARY) FROM DSN8710.EMP
WHERE SALARY <

(SELECT MAX(SALARY) FROM DSN8710.EMP))
UNION ALL
. . .
ORDER BY SALARY DESC;
```

You get the picture. The first query in the UNION ALL statement simply retrieves the largest salary. The second query retrieves the largest salary that is less than the first salary retrieved. The third query retrieves the largest salary that is less than the second salary retrieved, and so on until you have retrieved the *n* largest salaries. Be sure to include the ORDER BY clause to return the rows in descending order by salary.

Actually, this method of obtaining the top ten values is a little bit different from the other methods we have discussed, too. It actually returns the top ten *distinct* values – no duplicate salary values will be returned when using the brute force method. When multiple salary values fall within the requested range the results will show only one of the employees that qualify. This can be confusing because the query may return different employees each time it is run, depending on the access path chosen.

Additionally, the brute force method is not generally recommended because it can quickly become quite cumbersome to code; and making changes to such an unwieldy SQL statement can be very confusing. Furthermore, it is likely that the brute force method will not perform optimally because of all the MAX functions and subselects needed in the multiple UNION statements.

Want to return the bottom ten?

Any of these queries can be modified to return the bottom ten instead of the top ten. For the first query, simply remove the DESC from the ORDER BY clause. This will cause the rows to be sorted into ascending sequence, which is the default. Then the FETCH FIRST *n* ROWS ONLY will cause the bottom ten results to be returned.

For the middle query, using standard SQL alone, simply reverse the less than sign (<) to a greater than sign (>) in the subquery, and remove the DESC from the ORDER BY clause, as follows:

```
SELECT EMPNO, LASTNAME, FIRSTNME, SALARY
FROM DSN871Ø.EMP A
WHERE 1Ø > (SELECT COUNT(*)
FROM DSN871Ø.EMP B
WHERE A.SALARY > B.SALARY
AND B.SALARY IS NOT NULL)
ORDER BY SALARY;
```

And with the brute force method, you can simply deploy the same method but using the MIN function and greater-than predicates in place of the MAX function and less-than predicates.

BOTTOM LINE

The ‘top ten’ request is a common application requirement. Any application that needs to return an ordered subset of a given table can take advantage of one of these ‘top ten’ queries. Consider using SQL to return only the results you need instead of writing an application program that reads query results to limit the results set. SQL-only solutions can be easier to use than bulky application programs. But be aware that the SQL-only approach, depending on the method deployed, also may be less efficient.

Craig S Mullins
Director, Technology Planning
BMC Software (USA)

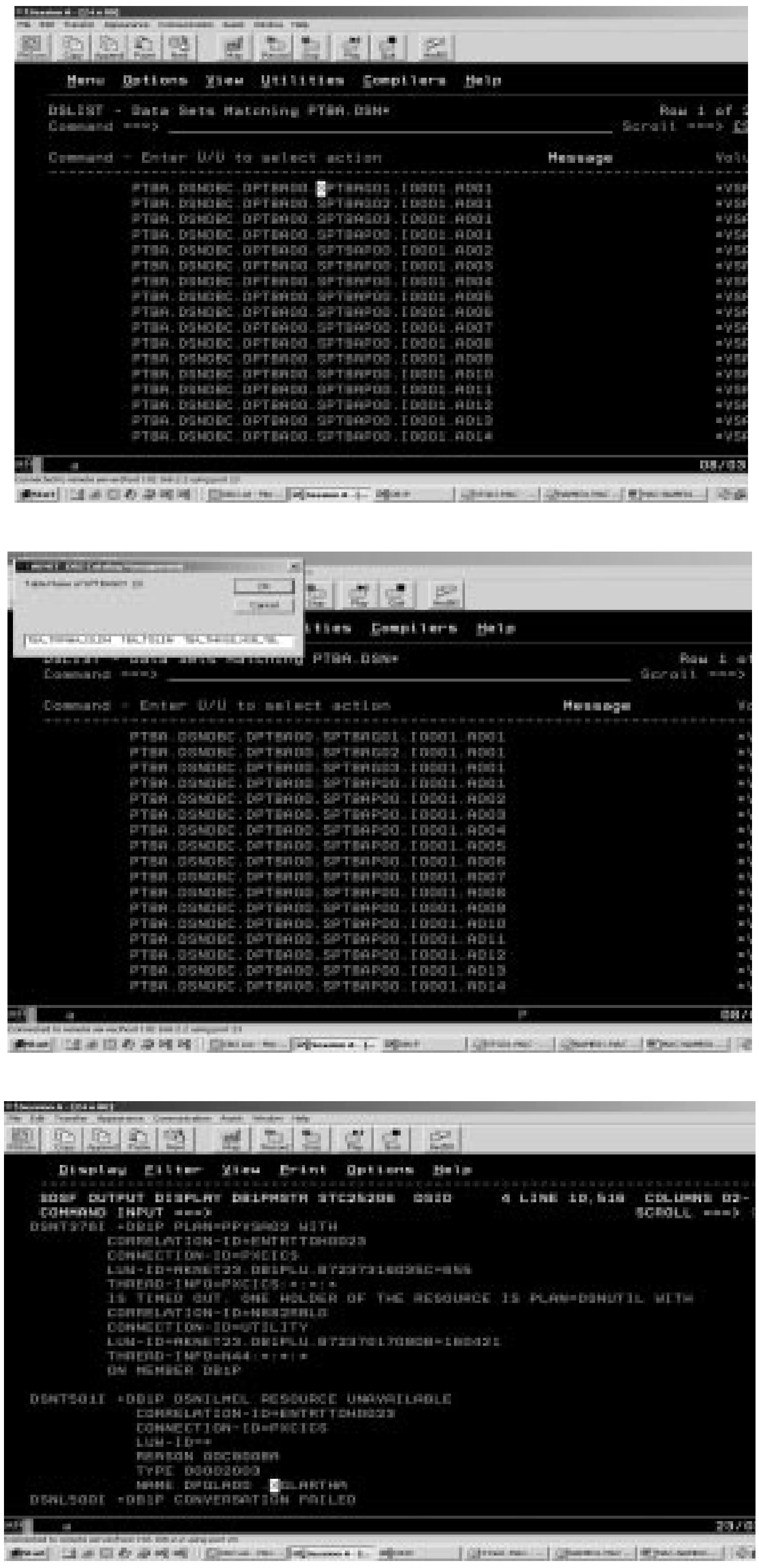
PC utilities to easily monitor/administer DB2

Most OS/390 DB2 utilities are written using REXX or COBOL, but with IBM Personnel Communications Version 4.3 it is possible to write macros in Visual Basic. Using macros, it is possible to capture a 24x80 screen in text format and evaluate it. In addition, with screen parsing, it is also possible to connect OS/390 DB2 across a DB2 Connect Gateway.

In OS/390 DB2, sometimes it is hard to derive the table or tables that a tablespace contains, or what is the index name of an indexspace and which table it belongs to.

There are two sample scripts/macros in this article. One of them is NAME01.MAC, which returns table name(s) if it parses a tablespace name, or it returns an index name and its table if it parses the indexspace name. The other is STG01.MAC, which returns STOGROUP and its VOLUMEs for a tablespace or indexspace.

You can activate NAME01.MAC by striking a key combination which you assign previously while the cursor is on the first character of a tablespace or indexspace on any screen (SDSF, ISPF etc). The macro parses the tablespace or indexspace name (in our shop index names always start with ‘X’) and gets the table name(s) or index name and its table from the DB2 catalog, and finally shows the information in a small window. STG01.MAC works like NAME01.MAC, but it is



```
  Menu  Options  View  Utilities  Compilers  Help

 DSLIST - Data Sets Matching PTBA.DSN*                          Row 1 of
 Command ===> ____________________________________________  Scroll ===>

 Command - Enter U/U to select action                    Message          Volu
 ------------------------------------------------------------------------------
          PTBA.DSNDBC.DPTBA00.SPTBAG01.I0001.A001                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAG02.I0001.A001                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAG03.I0001.A001                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A001                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A002                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A003                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A004                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A005                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A006                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A007                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A008                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A009                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A010                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A011                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A012                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A013                          *VSA
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A014                          *VSA
                                                                         08/03
```

```
                            tilities  Compilers  Help

                               PTBA.DSN*                          Row 1 of
 Command ===> ____________________________________________  Scroll ===>

 Command - Enter U/U to select action                    Message          Vo
 ------------------------------------------------------------------------------
          PTBA.DSNDBC.DPTBA00.SPTBAG01.I0001.A001                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAG02.I0001.A001                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAG03.I0001.A001                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A001                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A002                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A003                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A004                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A005                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A006                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A007                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A008                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A009                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A010                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A011                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A012                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A013                          *V
          PTBA.DSNDBC.DPTBA00.SPTBAP00.I0001.A014                          *V
                                                                         08/
```

```
   Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
  SDSF OUTPUT DISPLAY DB1PMSTR STC25208  DSID     4 LINE 10,518  COLUMNS 02-
  COMMAND INPUT ===>                                             SCROLL ===>
 DSNT376I -DB1P PLAN=PPYSA03 WITH
          CORRELATION-ID=ENTRTTOH0023
          CONNECTION-ID=PXCICS
          LUW-ID=RKNET23.DB1PLU.B72373160250C=855
          THREAD-INFO=PXCICS:*:*:*
          IS TIMED OUT. ONE HOLDER OF THE RESOURCE IS PLAN=DSNUTIL WITH
          CORRELATION-ID=NK82RBLD
          CONNECTION-ID=UTILITY
          LUW-ID=RKNET23.DB1PLU.B7237017080B=180421
          THREAD-INFO=A44:*:*:*
          ON MEMBER DB1P

 DSNT501I -DB1P DSNILMCL RESOURCE UNAVAILABLE
            CORRELATION-ID=ENTRTTOH0023
            CONNECTION-ID=PXCICS
            LUW-ID=*
            REASON 00C9008A
            TYPE 00002003
            NAME DPGLA00 .SGLARTHA
 DSNL500I -DB1P CONVERSATION FAILED
                                                                          23/0
```

Figure 1: Example usage

meaningful to use it from ISPF DSLIST because the macro also parses the partition number.

You must copy the macros to the *<Drive>:\...\Personal Communications\Private* folder.

Example output is shown in Figure 1.

STG01.MAC

```
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=
[PCOMM SCRIPT SOURCE]
OPTION EXPLICIT
'.......................................................................'
'.......................................................................'
'.... RETURNS STOGROUP and its VOLUMEs..............................'
'.......................................................................'
'............................Serdar Sabri ÖZKUBULAY - Ø1.Ø2.2ØØ2.......'
'Variables:
dim cnt
dim rst
dim title
dim header
dim msgtext
dim msgtext2
dim rc
dim colpos
dim rowpos
dim textØ1
dim partno
dim partno1
dim partno2
dim partno3
dim partst1
dim partst2
dim partst3
dim sqltextØ1
dim firstc
autECLSession.SetConnectionByName(ThisSessionName)
colpos = autECLSession.autECLPS.CursorPosCol
                                          'Get cursor's column position
rowpos = autECLSession.autECLPS.CursorPosRow
                                          'Get cursor's row position
textØ1 = autECLSession.autECLPS.GetText(rowpos, colpos, 8)
                                          'Parse object name
partno = autECLSession.autECLPS.GetText(rowpos, colpos + 16, 3)
```

```
                                           'Get partition number
partno1 = autECLSession.autECLPS.GetText(rowpos, colpos + 16, 1)
                                          'Get partition number digits
partno2 = autECLSession.autECLPS.GetText(rowpos, colpos + 17, 1)
partno3 = autECLSession.autECLPS.GetText(rowpos, colpos + 18, 1)
partst1 = InStr("Ø123456789",partno1)
'Check if it is numeric
partst2 = InStr("Ø123456789",partno2)
partst3 = InStr("Ø123456789",partno3)
if partst1 = Ø or partst2 = Ø or partst3 = Ø then
  partno = "Ø"
'Assign Ø if it is not numeric
end if
Set cnt = CreateObject("ADODB.Connection")
'Open connection to the database
cnt.Open "DSN=DB2DATA;UID=n682;PWD=serdarØ7"
'Conn.name, User, Password
Set rst = CreateObject("ADODB.Recordset")
'Open recordset
firstc = Mid(textØ1, 1, 1)    'Get the first charecter of TSNAME/ISNAME
if firstc = "X" Then
'Decide TSNAME or ISNAME and create SQL
  sqltextØ1 = "SELECT X.CREATOR, X.NAME, Y.PARTITION, Y.STORNAME,
Z.VOLID FROM SYSIBM.SYSINDEXES X, SYSIBM.SYSINDEXPART Y,
SYSIBM.SYSVOLUMES Z WHERE X.NAME=Y.IXNAME AND Y.STORNAME=Z.SGNAME  AND
X.CREATOR = Y.IXCREATOR AND X.indexspace ='" & textØ1 & "'"
Else
  sqltextØ1 = "SELECT Y.TSNAME, Y.PARTITION, Y.STORNAME, Z.VOLID FROM
SYSIBM.SYSTABLEPART Y, SYSIBM.SYSVOLUMES Z WHERE  Y.STORNAME=Z.SGNAME
AND Y.tsname ='" & textØ1 & "'"
end If
rst.open sqltextØ1, cnt                   'Get the information from DB2
Title = "AKNET   DB2 Catalog Management"          'Window title
Do
  if rst.EOF or rst.BOF Then exit do              'End of records
  if firstc = "X" Then                            'It is an INDEXSPACE
    if rst.Fields(2).Value = CInt(partno) or rst.Fields(2).Value = Ø
then
      msgtext = msgtext & rst.Fields(4).Value & "  "
      msgtext2 = "IS Name            : " & textØ1 & "        Part :   "
& rst.Fields(2).Value & (Chr(13)) & (Chr(13)) & "Storage Group  :  " &
rst.Fields(3).Value & (Chr(13))
    end if
    else                                           'It is a TABLESPACE
      if rst.Fields(1).Value = CInt(partno) or rst.Fields(1).Value = Ø
then
      msgtext = msgtext & rst.Fields(3).Value & "  "
      msgtext2 = "TS Name           : " & textØ1 & "        Part :   "
& rst.Fields(1).Value & (Chr(13)) & (Chr(13)) & "Storage Group  :  " &
rst.Fields(2).Value & (Chr(13))
```

```
    end if
  end If
  rst.MoveNext                                                     '
Locate next record.
Loop

header = msgtext2
rc = InputBox(header,Title, msgtext, 100, 100)
'Create information window
```

NAME01.MAC

```
[PCOMM SCRIPT HEADER]
LANGUAGE=VBSCRIPT
DESCRIPTION=
[PCOMM SCRIPT SOURCE]
OPTION EXPLICIT
'.......................................................................'
'.......................................................................'
'.... RETURNS Table Name(s) if it parse Tablespace name ..............'
'... RETURNS Index Name & It's Table if it parse Indexspace name .....'
'.......................................................................'
'...........................Serdar Sabri ÖZKUBULAY - 31.01.2002.......'
'Variables:
dim cnt
dim rst
dim msgtext
dim header
dim colpos
dim rowpos
dim text01
dim sqltext01
dim firstc
dim title
dim rc
dim i
autECLSession.SetConnectionByName(ThisSessionName)
colpos = autECLSession.autECLPS.CursorPosCol
                                          'Get cursor's column position
rowpos = autECLSession.autECLPS.CursorPosRow
                                          'Get cursor's row position
text01 = autECLSession.autECLPS.GetText(rowpos, colpos, 8)
                                          'Parse object name
Set cnt = CreateObject("ADODB.Connection")
                                      'Open connection to the database
cnt.Open "DSN=DB2DATA;UID=n682;PWD=serdar07"
                                           'Conn.name, User, Password
Set rst = CreateObject("ADODB.Recordset")
                                           'Open recordset
```

```
firstc = Mid(textØ1, 1, 1)
                             'Get the first charecter of TSNAME/ISNAME
If firstc = "X" Then            'Decide TSNAME or ISNAME and create SQL
   sqltextØ1 = "Select name, tbname from sysibm.sysindexes where
indexspace ='" & textØ1 & "'"
else
   sqltextØ1 = "Select name from sysibm.systables where tsname ='" &
textØ1 & "'"
end If
rst.open sqltextØ1, cnt                  'Get the information from DB2
Title = "AKNET   DB2 Catalog Management"          'Window title
i=1
Do
  If rst.EOF Or rst.BOF then exit do              'End of records
  If firstc = "X" Then                            'It is an INDEXSPACE
     header = "Index Name   &   Table name of    " & textØ1
     msgtext = msgtext & rst.Fields(Ø).Value & "   &   " &
rst.Fields(1).Value
  else                                            'It is a TABLESPACE
     header = "Table Name of " & textØ1 & "  (" & i & ")"
     msgtext =  msgtext & rst.Fields(Ø).Value & "    "
  end If
  rst.MoveNext                                     'Locate next record.
  i = i + 1                           'Number of tables in tablespace
Loop
rc = InputBox(header,Title, msgtext, 1ØØ, 1ØØ)
                                                'Create information window
```

Serdar Sabri Özkubulay
OS/390 DB2 Systems Programmer
Aknet (Turkey)

DB2 news

IBM has announced new and enhanced data management tools for z/OS, for DB2 and IMS, covering administration, recovery and replication, and performance and application management.

Enhancements include cloning of work statement lists, more support for generating work statement lists, and restartable execution of utilities, DSN, and DB2 commands in addition to the support of SQL statements currently provided.

Another enhancement is that ALTER of primary key will also change foreign keys, DB2 subsystem selection on DB2 Tools Launchpad, and DROP and REVOKE impact analysis.

DB2 High Performance Unload for z/OS, provides sequential reading and the capability to access DB2 data at high speed. It can unload data from image copies as well as active tables and create multiple output files during a single unload. It can also store the output in multiple formats, including user-tailored formats.

New capabilities include DB2 Admin Line command support, multiple unload statements in one SYSIN, tablespace name becoming an optional parameter, support of FastUnload from Computer Associates and Unload Plus from BMC, expressions supported in the SELECT clause of the select statement, and SQL-Case statements supported in the SELECT clause and in the WHERE clause.

DB2 Object Comparison Tool for z/OS, helps keep test and development system as a mirror image of the production system. It lets users compare objects (and dependent objects) from one source to another and, once a difference file is generated, the product can be used to generate the DB2 commands needed to bring the catalogues back into synchronization.

Enhancements include target changes, which can be generated into a work statement list, and there are improved reports.

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

* * *

BMC has announced the first in a series of SmartDBA enterprise data management products for DB2, providing DB2 UDB Unix and NT sites with administration, performance, and recovery tools.

The Web-based SmartDBA integrated tools are geared to managing production databases from anywhere at any time and embedded tools within each product are able to pinpoint and recommend solutions to problems.

SmartDBA can work independently of an enterprise management system or it can be configured to take advantage of an existing PATROL operation.

It includes SmartDBA Cockpit, DBXray for DB2 UDB, Space Expert for DB2 UDB, SQL-Explorer for DB2 UDB, and SQL analysis.

For further information contact:
BMC Software, 2101 City West Blvd, Houston, TX 77042-2827, USA.
Tel: (800) 841 2031.
URL: http://www.bmc.com/db2udb.