123

DB2 update

January 2003

In this issue

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

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

DB2 Update on-line

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

Editor

Trevor Eddolls

Contributions

When Xephon is given copyright, articles published in *DB2 Update* are paid for at the rate of £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.

Script for getting a list of all tables in the database in the order of referential integrity

Often, in a project for administration/development purposes, you need to get a list of tables in the order of referential integrity – ie you need to know in which tables you must enter records first and which tables should follow. For example, to perform a clean-up routine we need to find the master-child relationships between tables so that we can delete first the child records and then the master records. In large projects these relationships often tend to become intricate and cumbersome to find. This stored procedure lists all the tables in the database in the order of referential integrity. Each record has a table name and a level. The higher the level number the lower it is in the hierarchy.

The temp table in the procedure tmp_tree contains all the tables in the database along with the parent table. This procedure can also be modified to solve various hierarchical problems.

In this procedure we depend on the system records that exist in SYSCAT.TABLES and SYSCAT.REFERENCES. SYSCAT.TABLES contains a list of all the tables in the database, while SYSCAT.REFERENCES contains a list of all the tables that reference other tables along with the parents' names.

The first step is to get a list of all tables and insert it into a temp table. A LEFT OUTER join on SYSCAT.TABLES and SYSCAT.REFERENCES ensures that we get a list of all tables including those which do not reference other tables.

The next step is to set to 1 the tree_level of all records in the temp table that do not have any parents.

The next step is to set to the current level the tree_level of all records in the temp table that refer to the records in the previous level, ie update tree_level of records which have refers_name = (objname of records which have tree_level = 1) to 2. Next, update the tree_level of records which have refers_name = (objname of records which have tree_level = 2) to 3 and so on until you have no more records to update.

Select the max(tree_level) and objname from the temp table. Note that, in the SYSCAT.REFERENCES table, we have multiple records for a table that refers to multiple tables. So if a table *Emp* refers to two tables *Dept* and *Division*, and if *Dept* refers to *Division*, then *Division* will have tree_level 1. *Dept* will have tree_level 2. But *Emp* will have two records. The one that indicates *Division* as the parent will have tree_level as 2, and the one that indicates *Dept* as parent will have tree_level as 3. Hence we need to select the max(tree_level).

USP_GET_DB_TABLE_SEQ_DB2.SQL

```
CREATE PROCEDURE usp_get_db_table_seq (IN p_schema_name VARCHAR(128))
RESULT SETS 1
LANGUAGE SQL
/*******************************************************************
     NAME:          usp_get_db_table_seq.sql
     CREATED BY:    Ramanath Shanbhag
     CREATED ON:    Oct 22 2001
     DESCRIPTION:   This script generates table in sequence of
               referential integrity.
     CALLS:         NONE
*******************************************************************/
P1: BEGIN
     DECLARE v_level    INT;
     DECLARE  v_rowcount    INT;
        DECLARE stmt  VARCHAR(300);
     DECLARE c1 CURSOR WITH RETURN TO CALLER FOR s1;
     — Temp table
        DECLARE GLOBAL TEMPORARY TABLE tmp_tree
     (
          tree_level    INT,
          constraint_name    VARCHAR(18),
          objschema VARCHAR(128),
          objname  VARCHAR(128) NOT NULL,
          refkeyname    VARCHAR(18),
          refers_schema VARCHAR(128),
          refers_name   VARCHAR(128),
          parents       SMALLINT,
          children  SMALLINT,
          selfrefs  SMALLINT
     ) WITH REPLACE NOT LOGGED;
     SET  v_level = 0;
     /* Select all the user tables */
     INSERT INTO SESSION.tmp_tree (constraint_name, objschema, objname,
refkeyname, refers_schema, refers_name, parents, children, selfrefs)
     SELECT constname, TAB.tabschema, TAB.tabname, REF.refkeyname,
REF.reftabschema, REF.reftabname, TAB.parents, TAB.children,
TAB.selfrefs
     FROM SYSCAT.TABLES TAB LEFT OUTER JOIN SYSCAT.REFERENCES REF
     ON (TAB.tabschema = REF.tabschema AND TAB.tabname = REF.tabname)
     WHERE type = 'T' AND TAB.tabschema = p_schema_name;
     /* Update the v_level to 1 for all tables who do not refer any
other objects. */
     UPDATE SESSION.tmp_tree SET tree_level = v_level
     WHERE    (objschema = refers_schema
          AND  objname = refers_name)
     OR parents = Ø;
     SET v_level = v_level + 1;
     SET v_rowcount = 1;
     WHILE (v_rowcount <> Ø)
     DO
     /* Update the level to the next level for tables who have parents
             in previous level other than the tables which have self
             referential integrity. */
          UPDATE SESSION.tmp_tree SET tree_level = v_level
          WHERE    (refers_name IN (SELECT objname FROM SESSION.tmp_tree
                         WHERE tree_level = v_level - 1
                        )
                AND refers_name <> objname
               );
          GET DIAGNOSTICS v_rowcount = ROW_COUNT;
          SET v_level = v_level + 1;
     END WHILE;
     /* Select the output in a cursor and pass it as a resultset to the
calling proc. */
 SET stmt = 'SELECT MAX(tree_level) as tree_level, objschema, objname ';
 SET stmt = stmt || ' FROM SESSION.tmp_tree GROUP BY objschema, objname
ORDER BY 1';
     PREPARE s1 FROM stmt;
     OPEN c1;

END P1
```

USP_CREATEINSERTS

The stored procedure *CreateInsertsForDB* is used to generate insert statements for all the tables in the database schema. This procedure is a wrapper procedure, which utilizes the functionality provided by *usp_get_db_table_seq* and *usp_CreateInserts* to generate insert scripts for all tables in the database in the order of referential integrity.

The stored procedure internally calls *usp_get_db_table_seq* to get a list of all tables in order of referential integrity. It stores this result set in a temporary table. Then it loops through each record and calls *usp_CreateInserts* to generate the insert scripts for each table. Here we pass the parameter p_called_from = 'D'. As a result the *usp_CreateInserts* will insert the records in a temp table temp_*inserts*. All records from this temp table are stored into a physical table named *Insert_Scripts_For_DB*. Lastly the stored procedure returns all the records from the *Insert_Scripts_For_DB.* Thus you can have the insert scripts for all the tables in any database schema in SQL format which can be fired on any other database with the same schema.

Note that you can also select the records from *Insert_Scripts_For_DB* at a later stage.

```
CREATE PROCEDURE DB2ADMIN.CreateInsertsForDB (IN p_schema_name
VARCHAR(128) )
RESULT SETS 1
LANGUAGE SQL
/***********************************************************************
     NAME:         CreateInsertsForDB.sql
     CREATED BY:   Ramanath Shanbhag
     CREATED ON:   Oct 22 2001
     DESCRIPTION:  This script internally calls usp_CreateInserts
              stored procedure which creates a SELECT Statement.
              This Select query when executed generates INSERT
              statements for the existing data in the given table.
              This is done for all tables in the database.
     CALLS:        usp_get_db_table_seq, usp_CreateInserts
***********************************************************************/
P1: BEGIN
     DECLARE strSelect  VARCHAR(8000);
     DECLARE v_col_id   INT;
     DECLARE v_table_name    VARCHAR(128);
     DECLARE v_table_schema  VARCHAR(128);
     DECLARE v_str      VARCHAR(300);
     DECLARE fetchsts   INT;
        DECLARE v_called_from   CHAR(1) DEFAULT 'D';
     DECLARE loc1       RESULT_SET_LOCATOR VARYING;
     DECLARE not_found CONDITION FOR SQLSTATE '02000';
     DECLARE c1 CURSOR WITH RETURN TO CALLER FOR s1;
        DECLARE CONTINUE HANDLER FOR not_found SET fetchsts = 1;
     — Temp table which will contain the final insert scripts.
     DECLARE GLOBAL TEMPORARY TABLE temp_inserts
     (
       col_id  INTEGER NOT NULL GENERATED ALWAYS AS IDENTITY(START WITH
1, INCREMENT BY 1),
       schema_name VARCHAR(128),
       table_name  VARCHAR(128),
          strInsert   VARCHAR(8ØØØ)
        ) WITH REPLACE NOT LOGGED;
        /* Execute the usp_get_db_table_seq proc which generates the list
           of tables in order which respects referential integrity.
           ie master tables will be listed first and child tables last. */
        CALL usp_get_db_table_seq (p_schema_name);
        /* Get the result set returned by usp_get_db_table_seq into a
           cursor. */
ASSOCIATE RESULT SET LOCATOR (loc1) WITH PROCEDURE usp_get_db_table_seq;
           ALLOCATE c3 CURSOR FOR RESULT SET loc1;
           SET fetchsts = Ø;
        FETCH c3 INTO v_col_id, v_table_schema, v_table_name;
        /* Call the proc which generates insert stmts in a loop for every
           table. */
        WHILE fetchsts  = Ø
        DO
             /* Call the proc which generates the Insert stmts. */
      CALL usp_CreateInserts (v_table_schema, v_table_name, v_called_from);
        FETCH c3 INTO v_col_id, v_table_schema, v_table_name;
        END WHILE;
        CREATE TABLE Insert_Scripts_For_DB LIKE SESSION.temp_inserts
       INSERT INTO Insert_Scripts_For_DB SELECT * FROM SESSION.temp_inserts
        /*
        — Return the output as a resultset.
        SET v_str = 'SELECT * FROM SESSION.temp_inserts';
        PREPARE s1 FROM v_str;
        OPEN c1;
        */
END P1
```

Ramanath N Shanbhag (India) © Xephon 2003

DB2 UDB V8.1 beta

My concluding sentence in *DB2 UDB Version 7.2 enhancements* (*DB2 Update*, issue 114, April 2002) was "I wonder what IBM has in store when it announces V8!" Now the world knows, with the beta release of V8.1 containing an expiration date and an IBM warning not to use it in production.

The product changes include:

- IBM combined UDB Enterprise Edition (EE) and UDB

Enterprise-Extended Edition (EEE) into UDB Enterprise Server Edition (ESE).

- UDB Workgroup Edition is now UDB Workgroup Server Edition.
- IBM replaced UDB Satellite Edition V6 with UDB Personal Edition V8.
- UDB OLAP Starter Kit is not available.

Terminology changes include:

- Long tablespace = large tablespace.
- Nodegroup = database partition group.
- Online index reorganization = online index defragmentation of leaf pages.
- Country code = territory code.

A new term is *materialized query table.* This is a table whose definition is query-defined, with the data comprising precomputed results from one or more tables. It is also known as a *summary table.*

IBM classified the enhancements into manageability, performance, scalability, Data Warehouse and Warehouse Manager, application development, federated systems, business intelligence, and DB2 family. This article describes the more important enhancements.

MANAGEABILITY

LOAD

New functionality simplifies loading data into single- and multi-partition tables. Load operates at the table level, meaning exclusive access to the entire tablespace is not required and concurrent access is possible. Tablespaces are not quiesced.

The READ ACCESS option permits querying of pre-existing data. LOCK WITH FORCE compels applications to release their locks, permitting load to proceed smoothly.

A partitioned database can be loaded using single partition commands of LOAD and db2load, and APIs db2load and db2LoadQuery. A new CURSOR file type can load SQL query results into a database without needing to export to a data file.

LOAD generates column values, so the table does not need to be in check pending state. The LOAD QUERY command can query table states even if a load is in progress.

Storage management tool

Control Center has a new storage management tool that displays storage snapshots for a database, database partition group, or tablespace. It displays statistical information for:

- Tablespace – systems catalog and database monitor for tables, indexes, and containers defined under the tablespace group.
- Database or database partition group – all tablespaces defined in the given database or database partition group.
- Databases – all database partition groups within the database.

DBAs can monitor tablespace usage and data skew for database partition groups and cluster index ratio for database partition groups and tablespaces. DBAs can set thresholds for data skew, space usage, and index cluster ratio. Exceeding a threshold causes a warning or alarm flag.

FLUSH PACKAGE cache

A new function, FLUSH PACKAGE, removes cached dynamic SQL statements from the package cache by invalidating them. Current users can continue, but new requests require a compilation to create a new cache entry.

Logging

Dual logging is possible on all platforms supported by UDB.

Back-up and recovery

Back-up can use storage vendors' solutions if they have implemented the XSBA industry standard interface. DBAs can restore databases to different code pages. Tablespace recovery needs only the log files necessary. DBAs can specify point-in-time (PIT) rollforward recovery using local time rather than GMT.

Administration Notification Log

The new Administration Notification Log stores log information in two distinct logs, depending on intended usage:

- Administration notification log – V8.1 writes information about significant events. Many messages provide additional data to supplement SQLCODE. NOTIFYLEVEL specifies event type and detail. It does not contain diagnostic information.
- db2diag.log – contains diagnostic information.

Tivoli

V8.1 is Tivoli ready. Any installation creates the needed signature files, so Tivoli Inventory and Discovery can investigate a computer and find DB2. Tivoli is used for manageability with such functions as start and stop tasks, recover tasks.

Type-2 indexes

V8.1 provides support for Type-2 indexes, whose advantages include:

- Reduced next-key locking, by marking key deleted instead of physically removed from the index page, improves concurrency.
- Index can be created on columns > 255 bytes.
- Supports multi-dimensional clustering facility.

V8.1 creates all new indexes as type-2.

Miscellaneous

RENAME index allows renaming instead of the previous procedure of creating a new index, removing the old index, and changing the name of the new index. There is no user performance degradation.

Null and default compression saves disk space for tables containing many NULLS and SYSTEM DEFAULT values.

The AUTOCONFIGURE command is new. It recommends and optionally applies values for buffer pool sizes, database configuration, and database manager configuration. AUTOCONFIGURE provides initial tuning. AUTOCONFIGURE combined with CREATE DATABASE creates a tuned database.

PERFORMANCE

Multi-dimensional clustering (MDC)

MDC provides continuous automatic data clustering over multiple dimensions. Its use results in noteworthy performance improvements in queries (IBM estimates up to 100%) and other activities such as reorganization and index management during insert, update, and delete. IBM built MDC for data warehouses and VLDBs, although OLTP environments can benefit.

MDC enables physical clustering of a table on more than one key. Clustering indexes improve performance of range queries having predicates containing one or more keys of the clustering index. Good clustering reads only that portion of the table needed. More efficient prefetching is possible when pages are sequential.

An MDC table maintains its clustering over all dimensions automatically and continuously, eliminating table reorganization.

Prefetching

Using block based buffer pools further enhances prefetching. Prefetching code knows when a block buffer pool is available and will use block I/Os to read multiple pages in a single I/O, considerably improving performance. BLOCKSIZE parameter of CREATE and ALTER BUFFERPOOL SQL statement defines block size, which determines quantity of pages read. Buffer pools, by default, are page-based, causing read of prefetched contiguous pages into non-contiguous memory pages. Sequential prefetching of contiguous pages into contiguous buffer pool pages further enhances performance.

DBAs can create block based buffer pools by specifying a page area and a block area. Non-sequential prefetching workloads require page area.

Page cleaner I/O

V8.1 uses asynchronous I/O facilities to improve I/O performance, including page cleaning.

Materialized Query Tables (MQT)

MQT is a table defined by a query result containing precomputed results. V7 supported summary tables (called automatic summary tables – ASTs). V8.1 considers ASTs an MQT special type consisting of a fullselect containing a GROUP BY clause summarizing data from tables referenced in the fullselect.

SQL can route queries to MQTs if they contain a non-aggregated join. IBM's example is:

```
CREATE TABLE  bad_account   AS
     (SELECT  customer_name, customer_id, a.balance
        FROM  account a, customers c
        WHERE status IN ('delinquent', 'problematic', 'hot')     ...AND
a.customer_id = c.customer_id
        DATA INITIALLY DEFERRED REFRESH DEFERRED
```

UDB uses the MQT cached BAD_ACCOUNT results instead of the base table, providing better response time.

Many applications maintain and load tables that are MQTs. Improvement of dynamic query performance is possible, by identifying a user-maintained MQT. Insert, update, and delete operations are possible against user-maintained MQTs. Setting appropriate special registers lets query optimizer take advantage of the MQT.

MQT can reference nicknames and local tables. DBAs can cache remote data on a DB2 instance supported by Oracle, Sybase, and others. Routing a query to MQT yields better performance than getting results from the remote table. REFRESH TABLE performs maintenance locally.

REFRESH DEFERRED uses a staging table built by CREATE TABLE SQL to automatically refresh MQT. Any insert, update, delete modification of MQT results in changes being immediately appended to the staging table, enhancing performance. REFRESH DEFERRED eliminates high lock contention if MQT data does not need to be current.

SCALABILITY

COMPRESSION of NULLS and DEFAULTS

V8.1 VALUE COMPRESSION creates a new data row format, providing better storage of NULL and SYSTEM DEFAULT.

INSERT into UNION ALL views

V8.1 allows INSERT into UNION ALL views.

Logging

V8.1 provides an infinite log, allowing a single transaction to span both primary and archive logs with a maximum log space of 256GB.

Satellite administration

DBAs can create DB2CTLSRV and SATCTKDB into partitioned databases for exploiting scalability and additional processing power.

DATA WAREHOUSE CENTER AND WAREHOUSE MANAGER

Warehouse server

V8.1 supports warehouse server and associated logger, initialization, external trigger, and mail notification programs on AIX. The author suspects the programming team got a bonus because it eliminates the Windows dependency.

Warehouse agent

V8.1 supports Linux warehouse agent, programs, and transformers on 32-bit Intel processors for levels kernel 2.4.7 and glibc 2.2.4.

Clean transformer

Microsoft rewrote clean transformer to improve performance and functionality. Clean rules provided are carry over, find and replace, separate into ranges, clip, convert case, and encode invalid values. New enhancements are better null support, more data types, generate all, first, and last matches, error on multiple or no matches, differentiator and ordering columns for rules, and use of automatic summary tables.

Common Warehouse Metamodel (CWM) XML support

CWM is a model-based architecture containing features effective in and mappable to a broad range of representative warehouse configurations. OMG published CWM 1.0 in February 2001. CWM allows interchange of metadata with other warehousing tools conforming to CWM standard. CWM utility imports or exports CWM objects from an XML file and converts objects to DWC objects.

DWC column mapping

DWC provides enhanced support for column mapping between sources and targets:

- Source columns and target columns can scroll separately or together.
- Map all columns by column name and data type, by column name only, or by position.

- Permits single removal of all column mapping.
- Edit names and types of new target columns in column mapping page.
- The *Find Columns* feature can search for a column in source or target.

DWC cascading processes

V8.1 allows triggering based on success, failure, or completion of multiple steps by defining and enabling a schedule and task flow for the processes containing the steps. Progress can be monitored in the *Work In Process* window.

Multiple wait

Users can specify via the user interface that a step should wait on multiple steps to complete before running the dependent step. The specification mechanism is a process cascade tree.

SQL SELECT and UPDATE step

SQL can update a target table in DW without replacing the entire table or writing additional code by using key column mapping. The compiler automatically generates the UPDATE statement. It is editable in place and can run on demand.

Information Catalog Manager (ICM)

V8.1 has a new user interface, Information Catalog Center, which manages business metadata by providing capacity to organize, navigate, and search metadata objects. ICM has Java APIs usable by third party applications to manipulate metadata in the information catalog. ICM is invokable from Windows and Unix platforms either directly or from other DB2 centres. Object-level security allows only authorized users to view metadata objects.

Routines (stored procedures, user-defined functions, and methods)

V8 parameter styles, data type mapping, and system catalogs are the same for all three routine types. V8.1 merged catalog views for functions, methods, and stored procedures. SYSCAT.ROUTINES describes all routines defined in the database and SYSCAT. ROUTINEPARMS describes the parameters for routines, including return information.

V8.1 redefined the EXECUTE privilege to explicitly control who can invoke routines. Routine definer must have EXECUTE privilege on any packages used by an SQL statement.

V8.1 defined new authorities for registering external routines by using CREATE_ EXTERNAL_ ROUTINE. FENCED THREADSAFE routines run as threads inside a shared process. Each thread can read memory used by other threads in the threaded process.

Warning: a thread with flawed memory management can corrupt other threads or crash the entire thread process. CREATE_NOT_FENCED_ROUTINE is used to register NOT FENCED routines that run in the same process as the database manager.

Warning: it can corrupt databases and crash the database manager.

V8.1 provides each routine type with an ALTER statement, allowing the routine's EXTERNAL NAME to reference a new routine body. This permits the routine to use a new Java method rather than having to drop and re-register a Java stored procedure.

NOT FENCED routines support nesting and recursion. Stored procedures can return result sets. V8.1 automatically registers SQL procedures as NOT FENCED.

Java routines support nesting and recursion improving performance and scalability. DB2 library manager dynamically adjusts its library caching according to workload.

External user-defined functions and methods can contain read-only SQL statements in either static or dynamic mode.

The CALL statement is fully compiled, allowing it to be dynamically prepared in CLI, ODBC, embedded SQL, JFBC, and SQLj. Input arguments can be expressions subject to data type checking and promotion.

Development Center (DC)

V8.1 DC replaces the Stored Procedure Builder, providing more functions and procedures, including wizards for development tasks. Functions provided include:

- Create, build, and deploy Java and SQL stored procedures.
- Create, build, and deploy user-defined functions such as:
  - SQL tables and scalar UDFs
  - UDFs that read MQSeries messages
  - UDFs that access OLE DB data sources
  - UDFs that extract data from XML documents.
- Debug SQL stored procedures using an integrated debugger.
- See server contents for each database connection within a project or explicitly added to Server View.
- Provide export and import routines and project information.

DC provides add-ins for MS Visual C++, Visual Basic, and Visual InterDev.

SQL Assist

SQL Assist uses outline and details panels to organize information needed to create an SQL statement. Enhancements include:

- A redefined user interface.
- Assistance for creating table joins.

- SQL syntax checking.
- The option to copy and paste an existing SQL statement into SQL Assist for modifications.

Miscellaneous SQL

V8.1 supports INSTEAD OF triggers to do inserts, updates, deletes, and selects transparently against views even if a view is too complex to provide for native update operations.

V8.1 introduces new constraint types (informational constraint) that are rules usable to rewrite queries to improve performance. Their advantage is that the database manager does not enforce them.

Dynamic dispatch

V8.1 allows overwriting methods by reimplementing them in subtypes to provide functionality that is more specific. V8.1 executes the most specific method at run time, depending on the subject dynamic type.

Call Level Interface (CLI) LOAD functionality

V8.1 provides a new interface to DB2 LOAD utility from CLI, allowing data insert using LOAD instead of array inserts. This can yield major performance improvements when there are large amounts of data.

Declared Global Temporary Table (DGTT)

Enhancements include:

- The ability to create indexes using the CREATE INDEX statement.
- The ability to undo logging, to support rollback of data changes to DGTTs.
- Improved RUNSTATS command, which provides better performance.

IBM OLE DB Provider for DB2

IBM OLE DB Provider for DB2 allows DB2 to be a resource manager for the OLE DB provider, permitting OLE DB-based applications to extract or query DB2 data using the OLE interface. Features include:

- Support level 0 of OLE DB provider specification, including some level 1 interfaces.
- Enables applications created in one thread for use in any other thread.
- There is an error lookup service that returns DB2 error messages.

Web Services applications

Web services are a powerful new programming paradigm enabling development and deployment of loosely-coupled applications within a company or across industries. Web services use emerging technologies, including Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).

Programmers can create a Web service by wrapping an application, allowing its access using XML messages that are similarly wrapped to mask the underlying transport protocol. Publication of service in a standard-format registry permits people and applications to find and use it over the Web.

IBM WebSphere combined with DB2 allows DB2 to be a Web service provider to supply a complete Web services framework.

DB2 provides Document Access Definition extensions (DADx) for developing data-intensive Web services applications. DADx are XML documents containing stored procedure references, DB2 XML Extender references, or traditional SQL. Java programming is not required.

References: DB2 Web service provider demo is at http://www.ibm.com/software/data/developer/samples/video/.

JDBC driver

V8.1 provides new features and enhancements including an architectural improvement shortening the code path between JDBC driver and DB2 servers, providing better performance and stability.

V8.1 adds a new client layer communicating with the DB2 server via DRDA, replacing CLI and many layers beneath it. The new JDBC type 4 drivers reside on top of this Java client layer. It is a two-tier pure Java driver allowing direct communication via DRDA.

FEDERATED SYSTEMS

A DB2 *federated system* is a special type of distributed DBMS consisting of a DB2 instance operating on a server as the federated database, one or more data sources, and clients (users and applications) who access the database and data sources. It has the capability to distribute requests to multiple data sources within a single SQL statement. Advantages include:

- Ability to join tables and remote data sources as if all data is local.
- Compensate for SQL limitations at the data source by processing parts of a distributed request at the federated server.

V8.1 enhancements include:

- Support for Unix, HP-UX, and Windows.
- Perform insert, update, delete actions on data sources.
- Create remote tables on relational data sources.

BUSINESS INTELLIGENCE

OLAP Server and OLAP Integration Server

DB2 OLAP Server and its add-on features such as OLAP Server and OLAP Integration Server build OLAP applications that are production-ready and Web-ready. Two major enhancements are:

- OLAP Server Miner is a no-cost add-on of the OLAP Server feature that automatically mines large volumes of data, divulging opportunities and problems previously hidden, for immediate action.
- OLAP Integration Server Hybrid analysis builds a virtual extension mapping OLAP hierarchy lowest members to a relational database. This allows access to more data than the OLAP database can hold.

References: a Web site describing other enhancements is at http://www.ibm.com/software/data/db2/db2olap/.

DB2 Spatial extender

Spatial extender allows attributes, such as how far customers live from their office, to be used for business analysis. Major enhancements include:

- An Index Advisor assists in designing spatial indexes quicker.
- More functions allowing more extensive information scope.
- Integrate user- and vendor-geocoders via a simpler interface and greater ability to export spatial data to geobrowsers.
- More sample code for pasting into applications.
- Export SDE transfer data to files for loading to external data sources.

Spatial Extender conforms to OpenGISConsortium (OGS) and ISO standards.

References: *IBM DB2 Spatial Extender User's Guide and Reference,* db2sbx80.

Multiplatform tools

DB2 Recovery Expert for Multiplatforms V1

DB2 Recovery Expert is a targeted flexible automated recovery of database assets, letting systems remain on-line. Built-in SMART provides intelligent analysis of altered, corrupted, incorrect, or missing database assets, including tables, indexes, or data, and automates the process of rebuilding those assets to a correct point-in-time. Benefits include:

- Recover database objects without resorting to resource-intensive disaster recovery.
- Accurately roll back unwanted data changes throughout a database.
- Obtain intelligent assistance for determining the most efficient technique for a recovery situation.
- Recreate databases or selected objects to new environments.
- Support multiple databases on multiple operating systems and hardware platforms.

DB2 Recovery Expert is usable with V7.

DB2 Performance Expert for Multiplatforms V1

DB2 Performance Expert offers a comprehensive view, consolidating reports, doing analysis, and recommending changes on DB2 performance-related information. DB2 Performance Expert selectively employs and integrates views from all trace, snapshot, event, and DB2 Health Monitor outputs. It provides its own online snapshot reports, buffer pool analyzer, and reporting facility, selectively storable in its performance data warehouse for later analysis. DB2 Performance Expert uses SMART to recommend system-tuning changes to gain optimum throughput.

DB2 Performance Expert has a state-of-the-art workstationbased user interface, providing a common interface across DB2 server platforms.

DB2 High Performance Unload for Multiplatforms V2.1

DB2 High Performance Unload provides a fast efficient tool for unloading and extracting data for movement across enterprise systems or for reorganization in-place. It delivers high levels of parallelism when either unloading or extracting in partitioned database environments. DB2 High Performance Unload can process multiple SELECT statements simultaneously, allowing DB2 to finish the task with a single table scan. Other capabilities include:

- Quickly unload tablespaces of any DB2 supported type.
- Format output file-set, tape, disk, or named pipe for optional reorganization or reload.
- Simultaneously do several unloads accessing the same tablespace.
- Do unload against a full back-up to eliminate interference with DB2 production database.
- Use SELECT to unload selected rows and columns.
- Unload every n^{th} row.
- Generate load control statements for subsequent reload.
- Optionally unload data and move it from a partitioned table to a single file-set for use in another database environment.
- Unload DB2 UDB table data from multiple database partitions in a single request.

DB2 High Performance Update is usable in V7.

DB2 Table Editor for Multiplatforms V4.3

DB2 Table Editor offers direct database access to anyone creating, reviewing, or updating data. It offers constructible task-specific forms and an expert interface for browsing existing databases and *ad hoc* actions. DB2 Table Editor is usable with V7.

DB2 Web Query tool for Multiplatforms V1.3

DB2 Web Query tool provides an authoring environment using advanced SQL functionality to achieve maximum performance from any DB2 database. It is usable in V7.

DB2 Data Links Manager

DB2 Data Links Manager enables applications to manipulate data in both unstructured and structured files and in RDBMS. Files outside the database are accessible over a network and managed as if they were stored in the RDBMS. DB2 UDB extensions provide the integration. Enhancements include:

- There is additional availability for other vendors' operating systems.
- DB2 Data Links Manager can update linked files while they remain under its control without having to unlink, make changes, and relink.
- There is better performance for archive, data recovery, and replication on linked files.
- New security features restricts file linking to authorized users.

Reference: *IBM DB2 Universal Database Data Links Manager Administrative Guide and Reference,* db2z0x80.

DB2 XML Extender

DB2 XML Extender is an end-to-end solution for storing and retrieving XML documents from DB2 databases. New features include:

- DB2 XML Extender supports Web services with WORF (Web Object Runtime Framework). It can start Web services from the Internet.

- DB2 XML Extender supports MQSeries applications allowing users to send to or retrieve from MQSeries message queues.

DB2 Net Search Extender

DB2 Net Search Extender merges V7.2 in-memory search capabilities with the text search capabilities of V7.2 DB2 Text Information Extender into a single DB2 option. Search options include word, phrase, Boolean, fuzzy, wildcard, proximity, free-text, or thesaurus. The merger of these two extenders is an integrated solution conforming to SQL Multimedia (SQL/MM) standards. It can be used to search DB2 federated databases, other DB2 databases, or Informix Dynamic (IDS) databases.

Host and iSERIES applications

Host and iSERIES applications can use two-phase commit when connecting to DB2 UDB over TCP/IP.

Eric Garrigue Vesely
Principal/Analyst
Workbench Consulting (Malaysia)

UDB – restrictions on renaming a table

This article looks at the facility to be able to rename a table. There has long been a requirement to rename a table rather than have to define a new table and then perform an unload/reload or do a select into. DB2 now has this facility. I ran all of the SQL below on Windows 2000 running DB2 UDB 7.2 FP7 and used the sample database.

The command syntax is shown below:

```
>db2 RENAME <source-table-name> TO <target-table-name>
```

Unfortunately, it is not as simple as this! There are various restrictions on the ‘source’ table, and if any of these restrictions exist, you cannot rename the table. The *SQL Reference Manual* details these restrictions (quoting directly from the manual):

The source table must not:

- *Be referenced in any existing view definitions or summary table definitions.*
- *Be referenced in any triggered SQL statements in existing triggers or be the subject table of an existing trigger.*
- *Be referenced in an SQL function.*
- *Have any check constraints.*
- *Have any generated columns other than the identity column.*
- *Be a parent or dependent table in any referential integrity constraints.*
- *Be the scope of any existing reference column.*

Let me show you the queries you can run to check for each of the above restrictions. Suppose we have a table called FRED and want to check that we can in fact use the rename command to rename it.

Before going through the SQL, bear in mind that the catalog will have to be updated if the table name changes, so if you have lots of grants etc on the table, there will be a proportionate amount of catalog update activity.

To check whether the table is referenced in any existing view definitions or summary table definitions:

```
select
substr(viewschema,1,1Ø),substr(viewname,1,1Ø),dtype
from syscat.viewdep
where btype = 'T' and bschema = 'DB2ADMIN' and bname = 'FRED';
```

Suppose you had created a view based on table FRED:

```
>db2 create view hmv as select * from fred
```

If you now try to rename table FRED (>db2 rename fred to sue) you will get a ‘SQL0750N The source table cannot be renamed because it is referenced in a view, summary table, trigger, SQL function, SQL method, check constraint, or referential constraint. SQLSTATE=42986' message.

To check whether the table is referenced in any triggered SQL statements in existing triggers or is the subject table of an existing trigger:

```
select
substr(trigschema,1,10),substr(trigname,1,10)
from syscat.trigdep
where btype = 'T' and bschema = 'DB2ADMIN' and bname = 'FRED';
```

An example of creating a trigger is:

```
create table fred (id int, item char(10));
create table harr (ct int);
–
drop trigger hmt;
CREATE TRIGGER hmt
AFTER INSERT ON fred
FOR EACH ROW MODE DB2SQL
UPDATE harr SET ct = ct + 1;
```

If you try to rename table FRED you will get the 'SQL0750N' message.

If the table FRED was the 'target' of the trigger, as in:

```
CREATE TRIGGER hmt
AFTER INSERT ON harr
FOR EACH ROW MODE DB2SQL
UPDATE fred SET ct = ct + 1;
```

then, if you try to rename table FRED, you still get the 'SQL0750N' message.

Therefore it doesn't matter whether the table FRED is the 'source' or 'target' table in a trigger, either way you cannot rename it whilst it is referenced in a trigger statement.

To check whether the table is referenced in an SQL function:

```
select
substr(funcschema,1,10),substr(funcname,1,10)
from syscat.funcdep
where btype = 'T' and bschema = 'DB2ADMIN' and bname = 'FRED';
```

You could create an SQL function as follows:

```
drop function hmf;
–
CREATE FUNCTION HMF ()
RETURNS DOUBLE
LANGUAGE SQL
READS SQL DATA
DETERMINISTIC
RETURN
SELECT COUNT(*) FROM FRED;
–
select hmf() from sysibm.sysdummy1
```

Then, if you try to rename table FRED, you will still get the 'SQL0750N' message.

To check whether the table has any check constraints:

```
select
substr(definer,1,1Ø),substr(constname,1,1Ø)
from syscat.checks
where tabschema = 'DB2ADMIN' and tabname = 'FRED';
```

An example of a check constraint is:

```
create table fred (id int constraint hmc check (id > 5));
insert into fred values(1Ø);
insert into fred values(3);
select * from fred;
```

Then, if you try to rename table FRED, you will still get the 'SQL0750N' message.

To check whether the table has any generated columns other than the identity column:

```
select
colname
from syscat.columns
where generated in ('A','D') and identity = 'N' and
tabschema = 'DB2ADMIN' and tabname = 'FRED';
```

An example is:

```
create table fred (id int, tot int generated always as (id + 3));
insert into fred(id) values(1Ø);
select * from fred;
```

Then, if you try to rename table FRED, you will still get the 'SQL0750N' message.

If FRED is defined with an identity column:

```
CREATE TABLE FRED
(INV_NUM INT NOT NULL unique GENERATED ALWAYS AS IDENTITY
(START WITH 1ØØØ,INCREMENT BY 1), ITEM CHAR (1Ø));
```

you *can* rename the table.

To check whether the table is a parent or dependent table in any referential integrity constraints:

```
select
substr(tabschema,1,1Ø), substr(tabname,1,1Ø),parents,children
from syscat.tables
where tabschema = 'DB2ADMIN' and tabname = 'FRED'
and (parents >Ø or children >Ø);
```

An example would be:

```
CREATE TABLE fred(ID INT NOT NULL PRIMARY KEY, item char(1Ø));
—
CREATE TABLE harr
(hid INT,desc CHAR(2Ø), CONSTRAINT FRED_FK
FOREIGN KEY (hID) REFERENCES fred (ID));
```

Then, if you try to rename table FRED, you will still get the ‘SQL0750N’ message.

To check whether the table is the scope of any existing reference column:

```
Select
Substr(tabschema,1,1Ø), substr(tabname,1,1Ø)
from syscat.references
where reftabschema = 'DB2ADMIN' and reftabname = 'FRED';
```

An example of defining a reference column is shown below:

```
create table fred (id2 int not null unique);
Create table fred1 (id1 int
constraint hm1 References fred (id2) on delete no action);
—
insert into fred values(1Ø);
—
insert into fred1 values(1Ø);
select * from fred1;
```

You cannot rename FRED, because it is referenced in table fred1. If you try, you will get the ‘SQL0750N’ message.

I haven't run any tests to see what the impact is on the catalog if you try and rename a table with lots of authorities defined on it.

I hope I have given you all the SQL you will need to run before you can rename a table.

C Leonard
Freelance Consultant (UK)

DB2 for z/OS performance tuning

Assuring optimal performance is one of the biggest challenges for DB2 database administrators (DBAs). The loudest complaints about poor performance usually come from those users who must wait longer than normal for their applications to respond. No-one likes waiting, especially if they are not accustomed to waiting.

But what causes those once fast applications to stall and deliver sub-par performance? If this question had an easy answer, many DBAs would be out of work. In truth, there is no easy, or short, answer to this question. As a starting point, though, let's examine the fundamentals of performance management and optimization and develop a roadmap for tuning DB2 for z/OS databases and applications.

Every database application, at its core, requires three components in order to operate: the system, the database, and the application. To deliver performance, the DBA must be able to monitor and tune each of these components. This is easier said than done.

TUNING THE SYSTEM

The *system* consists of the hardware and system software required for database applications to provide service to end users. This includes the computer itself and its disk subsystems, network connections, and all peripherals. From a software perspective, the system includes the operating system, the file system, the database management system (DBMS) itself, networking protocols and any related middleware such as transaction processors or message queues.

To deliver consistent system performance, the DBA must have the resources to monitor, manage, and optimize the performance of these disparate pieces of hardware and software. Some of the tasks required for system tuning include the proper allocation and management of memory structures (eg buffer pools, program cache area, etc), storage management, integration of the DBMS with other system software, proper usage of database logs, and coordination of the operating system resources used by the DBMS. Additionally, the DBA must control the installation, configuration, and migration of the DBMS software. If the system isn’t performing properly, everything that uses the system will perform poorly. In other words, a poorly performing system impacts every database application.

When managing DB2 system performance, the DBA’s first job is to ensure that all of the allied agent address spaces are tuned and connected properly. This includes, but is not limited to, CICS, IMS/TM, TSO, WebSphere, MQSeries, and DB2 Connect. All of these system software components interact with, and can affect the performance of, DB2 applications. If not configured and tuned properly, performance degradation can occur.

For example, when DB2 data is accessed using CICS, multiple threads can be active simultaneously, giving multiple users concurrent access to a DB2 subsystem of a single CICS region. Using the CICS Attach Facility, a mechanism that connects CICS with DB2, you can connect each CICS region to only one DB2 subsystem at a time. You can connect each DB2 subsystem, however, to multiple CICS regions simultaneously. DB2 provides services to CICS via MVS TCBs. All of these TCBs reside in the CICS address space and perform cross-memory instructions to execute the SQL code in the DB2 database services address space (DSNDBM1).

Furthermore, the Resource Control Table (RCT) must be configured for each DB2 program that will run under CICS. The

RCT applies only to CICS transactions that access DB2 data; it defines the manner in which DB2 resources will be used by CICS transactions. In particular, the RCT defines a plan for each transaction that can access DB2. Additionally, it defines parameters detailing the number and type of threads available for application plans and the DB2 command processor.

The DB2 DBA also must ensure that appropriate DB2 system parameters are set using DB2 commands and DSNZPARMs. One of the most important areas for tuning here is memory usage. DB2 uses memory for buffer, EDM, RID, and sort pools to cache data and structures in memory. The more efficiently memory is allocated to these structures, the better the overall DB2 subsystem will perform – and therefore, DB2 application performance can be enhanced.

When allocating DB2 buffer pools, keep these rules of thumb in mind:

- Don’t allocate everything to a single buffer pool (eg BP0). Use a multiple buffer pool strategy.
- Explicitly specify a buffer pool for every table space and index. Don’t code DDL that just lets your DB2 objects default to a buffer pool.
- Isolate the DB2 Catalog in BP0; put user and application DB2 objects into other buffer pools.
- Consider separating indexes from table spaces with each in their own dedicated buffer pools.
- Consider isolating heavily hit data into its own buffer pool to better control performance.
- Consider isolating sorts into a single buffer pool and tuning for mostly sequential access (eg BP7).
- Consider separating DB2 objects into separate buffer pools that have been configured for sequential versus random access.

Forget about trying to follow a cookie-cutter approach to buffer pool management. Every shop must create and optimize a buffer pool strategy for its own data and application mix. DB2 offers buffer pool-tuning ‘knobs’ that can be used to configure virutal buffer pools to the type of processing they support. They are:

- Deferred write threshold (DWQT) – this value is expressed as a percentage of the virtual buffer pool that might be occupied by unavailable pages. When this threshold is reached, DB2 will start to schedule write I/Os to externalize data. The default is 50%, which is probably too high for most shops.
- Vertical deferred write threshold (VDWQT) – this value is basically the same as DWQT, but for individual data sets. The default is 10%, which is also likely to be too high for most shops.
- Sequential steal threshold (VPSEQT) – this value is expressed as a percentage of the virtual buffer pool that can be occupied by sequentially accessed pages. Tune buffer pools for sequential access such as scans and sorting by modifying VPSEQT to a larger value. The default is 80%.
- Sequential steal threshold for parallel operations (VPPSEQT) – the default value is 50%.
- Assisting parallel sequential threshold (VPXPSEQT) – this value is basically the VPPSEQT for operations from another DB2 subsystem in the data sharing group.

These parameters can be changed using the ALTER BUFFERPOOL command. Additionally, hiperpools can be created to back up DB2 virtual buffer pools with additional memory. DB2 provides several tuning knobs for hiperpools, too, including HPSIZE to adjust the size of hiperpools and HPSEQT to adjust the hiperpool sequential steal threshold. DB2 cannot directly access data in a hiperpool, though. Hiperpools act as sort of a secondary cache for DB2’s virtual buffer pools. When data needs to be removed from a virtual pool, it can be moved to a hiperpool if one exists. The next time that data is needed by DB2 it can be moved from the hiperpool to the virtual pool. This is faster than re-reading that data from disk, as would be necessary if no hiperpool was defined.

The EDM pool is used for cacheing internal structures used by DB2 programs. This includes DBDs, SKCTs, CTs, SKPTs, and PTs. It also includes the authorization cache for plans and packages, as well as the cache for dynamic SQL mini-plans. As a general rule, you should shoot for an 80% hit rate with the EDM pool; this means that only one out every five times should a structure need to be loaded from disk into the EDM pool.

Finally, remember that buffer and EDM pool tuning are in-depth subjects that cannot be adequately covered in a high-level article such as this. Additionally, there is much more to proper DB2 system performance tuning than allied agent and memory tuning. Other system elements requiring attention include locking, logging, and Parallel Sysplex configuration and management for DB2 data-sharing shops.

TUNING THE DATABASES

The second component of DB2 performance tuning is making sure the *database* is optimally created and maintained. The database stores the data that is used by the application. When the application needs to access data, it does so through DB2 to the database of choice. If the database isn't optimally organized or stored, the data it contains will be difficult or slow to access. The performance of every application that requires this data will be negatively impacted.

The first step of database performance optimization is that of assuring an optimal database design. Database design is the process of transforming a logical data model into a physical database design and then implementing the physical model as an actual database. Proper database design requires up-front data modeling and normalization – in other words, a logical data model is required before you can even begin to design a physical database. Assuming a logical model exists, it should be used as the basis for creating the physical database. The physical database is created by transforming the logical data model into a physical implementation based on an understanding of the

DBMS to be used for deployment. But a simple one-to-one mapping of logical entity to physical table is unlikely to result in an optimal physical database design.

Successfully developing a physical database design requires a good working knowledge of DB2's features. More precisely, the DBA must possess in-depth knowledge of physical database objects supported by DB2 as well as the physical structures and files required to support those objects. This must include knowledge of the manner in which DB2 supports indexing, referential integrity, constraints, data types, and other features that augment the functionality of database objects. Armed with the correct information, the DBA can create an effective and efficient database from a logical data model.

The DBA creating the physical DB2 database implementation should keep the following in mind when building the databases:

- Keep the physical database as normalized as possible. However, performance should win out over aesthetics. In other words, don't let data modellers dictate 'physical' design. For example, if you need to reorder columns in the table to optimize logging, do it.
- In general, put each table in its own table space. Exceptions can be made for very small tables such as code and reference tables.
- In general, favour partitioned and segmented table spaces over simple table spaces. And don't specify a DSSIZE of 4GB or greater unless you really need a large table space (doing so will waste space).
- Don't create base table views.
- Use NULL sparingly.
- Use appropriate DB2 data types (eg use DATE for dates instead of CHAR or numeric data types).
- Consider using DB2 compression instead of using VARCHAR columns. With compression, there is less overhead and no programmatic handling is required.

- Favour using DB2 declarative RI instead of programmatic RI. It will usually perform better and is easier to administer. However, it's generally a bad idea to use RI for lookup and reference tables.
- Avoid the DDL defaults – they are usually wrong. Explicitly code every single DDL option when creating DB2 objects.
- Calculate the appropriate amount of free space for both PCTFREE and FREEPAGE based on the frequency of modification. Do not just default every table space to 10% free space. DB2 DBAs should always keep in mind that a proper indexing strategy can be the primary factor to ensure optimal performance of DB2 applications.

To create appropriate indexes, the DBA must have the SQL that is to be used against the DB2 objects in order to understand the access patterns. But many times the DBA is called upon to create indexes well in advance of the creation of any SQL. Creating indexes to optimize SQL performance without any knowledge of the SQL can be a challenging dilemma. DBAs can develop some basic rules of thumb, though. These steps can be used as a basic roadmap for DB2 index creation:

- First, take care of unique and primary key constraints by creating unique indexes on those columns. Then consider creating indexes for each foreign key to optimize RI.
- The next step is to examine the most heavily used queries. Look at the predicates and build indexes to support these queries. Consider overloading indexes by adding columns to encourage index only access.
- Next, examine queries that invoke sorts. Look for ORDER BY, GROUP BY, UNION, and SELECT DISTINCT. Consider building indexes to support these clauses so DB2 can avoid initiating an expensive sort. Look at your indexes and make sure you've chosen the first column wisely. In general, the first column of the index should have a high cardinality.
- In general, consider avoiding indexes on variable columns because DB2 will expand the VARCHAR column to its full

length in the index.

- After coming up with a 'first stab' indexing strategy consider the insert, update, and delete implications of those indexes. If the columns of those indexes must be modified in any way,

Column	*Description*	*Catalog Table*	*Impacts*
NEAROFFPOSF	Rows located out of position but near where they belong	SYSIBM.SYSINDEXPART	Table spaces
FAROFFPOSF	Rows located out of position and far from where they belong	SYSIBM.SYSINDEXPART	Table spaces
PERCDROP	Percentage showing the amount of space utilized by dropped tables.	SYSIBM.SYSTABLEPART	Table spaces
CLUSTERRATIOF	Ratio of rows that are in sequence by the clustering index key	SYSIBM.SYSINDEXES	Indexes
NEARINDREF	Rows that have near indirect references	SYSIBM.SYSTABLEPART	Indexes
FARINDREF	Rows that have far indirect references	SYSIBM.SYSTABLEPART	Indexes
LEAFDIST	The average number of pages between consecutive index leaf pages	SYSIBM.SYSINDEXPART	Indexes

Figure 1: DB2 Catalog statistics

DB2 will incur additional overhead keeping the indexes up to date.

Over time, as data is modified and updated, DB2 will have to move the data around within the database. Such activity causes the data to become fragmented and inefficiently ordered. The longer the database remains online and the more changes are made to the data, the more inefficient database access can become. To overcome disorganized and fragmented databases, the DBA can run a reorganization utility to refresh the data and make the database efficient once again. But the key to successful reorganization is to reorganize only when the database requires it. This is done by keeping RUNSTATS updated and then analysing the appropriate statistics that show table space and index organization. Consult Figure 1 for a listing of the pertinent statistics.

As PERCDROP, FAROFFPOS, and NEAROFFPOS increase, the need to reorganize the table space increases. As CLUSTERRATIO decreases, and as NEARINDREF, FARINDREF, and LEAFDIST increase, the need to reorganize the index increases. Some companies reorganize excessively, regularly scheduling database reorganization jobs whether the database is fragmented or not. This wastes valuable CPU cycles.

But reorganization is only one of many database performance tasks performed by the DBA. Others include dataset placement, partitioning for parallel access, managing free space, and assuring optimal compression.

TUNING THE APPLICATIONS

The third, and final, component of database performance tuning involves analysing and optimizing the *application* itself. Indeed, as much as 80% of all database performance problems are caused by inefficient application code. The application code nominally consists of two parts: the SQL code and the host language code in which the SQL is embedded.

SQL is simple to learn and easy to start using. But SQL tuning and optimization is an art that takes years to master. Some general rules of thumb for creating efficient SQL statements include:

- Let SQL do the work instead of the application program. For example, code an SQL join instead of two cursors and a programmatic join.
- Simpler is generally better, but complex SQL can be very efficient.
- Retrieve only the columns required, never more.
- Retrieve the absolute minimum number of rows by specifying every WHERE clause that is appropriate.
- When joining tables, always provide join predicates. In other words, avoid Cartesian products.
- Favour using Stage 1 and indexable predicates.
- If possible, avoid sorting by creating indexes for ORDER BY, GROUP BY, and DISTINCT operations.
- Avoid 'black boxes' – that is, avoid I/O routines that are called by programs instead of using embedded SQL.
- Avoid deadlocks by updating tables in the same sequence in every program.
- Issue data-modification statements (INSERT, UPDATE, DELETE) as close to the COMMIT statement as possible.
- Be sure to build a COMMIT strategy into every batch program that changes data. Failing to COMMIT can cause locking problems.

Every DBMS provides a method of inspecting the actual access paths that will be used to satisfy SQL requests. The DBA must thoroughly understand the different types of access path, and know which ones are best in a given situation. For DB2, the DBA must be able to interpret the output of the access path explanation produced by EXPLAIN. This information is encoded in the PLAN_TABLE and must be interpreted. To make matters more difficult, the PLAN_TABLE doesn't contain 100% of the information required to determine whether the SQL will perform efficiently. The DBA (and programmers too) must be able to read the PLAN_TABLE in conjunction with the SQL code, host language code, and information from the DB2 catalog to judge the efficiency and effectiveness of each SQL statement.

Host language code refers to the application programs written in C, COBOL, Java, Visual Basic, or the programming language *du jour*. SQL statements are usually embedded into host language code and it's quite possible to have finely tuned SQL inside inefficient host language code. And this, of course, would cause a performance problem.

THE BOTTOM LINE

DBAs must understand all three aspects of database performance management. The system, databases, and applications all must be monitored and tuned to assure efficient DB2 performance.

Furthermore, the DBA must be able to identify performance problems as they occur. Once identified, each problem must be analysed to determine its cause. Only then can a proper tuning strategy be deployed to rectify the problem. The guidelines in this article can help, but remember: each performance management situation is different, and general guidelines may not apply.

This article only touches the tip of the DB2 performance tuning and management iceberg. Database performance management is a complex and ever-changing discipline that requires rigorous attention and focus to master. And just when you seem to understand things, that new version of DB2 comes out and changes everything! But this is good. It might make life more difficult as you learn the latest DB2 nuances, but new releases invariably help to make DB2 applications more efficient.

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

Automatic placement of user-managed datasets – part 2

This month we conclude the code to improve disk I/O performance by automatically placing user-managed datasets.

```
third_phase:

ADDRESS "ISPEXEC" "TBSORT TBGRPA "||,
        "FIELDS(TGRPTXT1,C,A)"
ADDRESS "ISPEXEC" "TBTOP TBGRPA"
ADDRESS TSO
w_sp_name = ''
w_tsp_name = ''
w_wsp_name = ''
wsp_name = ''
sp_name = ''
tsp_name = ''
max_cyl = Ø
rows_cnt = Ø
rec1 = ''
rec2 = ''
inp_rows = Ø
oup_rows = Ø
tot_cyls = Ø
i = Ø
j = Ø
k = Ø
call thrd_rtn
max_cyl = 999999 - max_cyl
   do rowsvar = 1 to rows_cnt
       rec4 = rows.rowsvar
       rec2 = ''
       rec2 = rec2||substr(word(rec4,1),1,8)||'  '
       w_cyl_val = value(word(rec4,3))
       wn_cyl_val = 999999 - w_cyl_val
       rec2 = rec2||right(max_cyl,6,'Ø')||w_wsp_name
       rec2 = rec2||right(wn_cyl_val,6,'Ø')||'   '
       rec2 = rec2||substr(word(rec4,2),1,8)||'  '
       rec2 = rec2||substr(word(rec4,4),1,8)||'  '
       rec2 = rec2||substr(word(rec4,5),1,2)||'  '
       rec2 = rec2||substr(word(rec4,6),1,2)||'  '
       rec2 = rec2||subword(rec4,7)||'  '
       call write_tbgrpb
       oup_rows = oup_rows + 1
   end
return
```

```
 thrd_rtn:
/*********/
eof = 'NO'
rows. = ''
rows.Ø = Ø
rowsvar = Ø
rec1 = ' '
rec2 = ' '
max_cyl = Ø
first_rec = Ø
j = Ø
k = Ø
INLIST.    =  ""
INLIST.Ø   =  Ø
OUTLIST.    =  ""
OUTLIST.Ø   =  Ø
do while eof = 'NO'
      ADDRESS "ISPEXEC" "TBSKIP TBGRPA"
      if rc > 4  then
         do
            ADDRESS TSO
            eof = 'YES'
         end
      else
         do
            rec1 = TGRPTXT1
            call process_rtn_thrd
         end

end
ADDRESS "ISPEXEC" "TBCLOSE TBGRPA"
return

 process_rtn_thrd:
/*****************/
ADDRESS TSO
inp_rows = inp_rows  +  1
sp_name = substr(word(rec1,4),1,8)
wsp_name = word(rec1,2)
cyls = value(word(rec1,3))
if first_rec = Ø then
  do
    first_rec = 1
    max_cyl = cyls
    rows_cnt = 1
    rows.rows_cnt = rec1
    w_wsp_name = wsp_name
  end
else if w_wsp_name <> wsp_name then
  do
     max_cyl = 999999 - max_cyl
```

```
      do rowsvar = 1 to rows_cnt
        rec4 = rows.rowsvar
        rec2 = ''
        rec2 = rec2||substr(word(rec4,1),1,8)||'  '
        w_cyl_val = value(word(rec4,3))
        wn_cyl_val = 999999 - w_cyl_val
        rec2 = rec2||right(max_cyl,6,'Ø')||w_wsp_name
        rec2 = rec2||right(wn_cyl_val,6,'Ø')||'   '
        rec2 = rec2||substr(word(rec4,2),1,8)||'  '
        rec2 = rec2||substr(word(rec4,4),1,8)||'  '
        rec2 = rec2||substr(word(rec4,5),1,2)||'  '
        rec2 = rec2||substr(word(rec4,6),1,2)||'  '
        rec2 = rec2||subword(rec4,7)||'  '
        call write_tbgrpb
        oup_rows = oup_rows + 1
      end
      w_wsp_name = wsp_name
      max_cyl = cyls
      rows_cnt = 1
      rows.rows_cnt = rec1
   end
else
    do
       rows_cnt = rows_cnt  +  1
       rows.rows_cnt = rec1
       if cyls > max_cyl then
         max_cyl = cyls
    end
return Ø

 write_tbgrpb:
/*************/
  TGRPTXT2 = rec2
  ADDRESS "ISPEXEC" "TBADD TBGRPB"
  if rc > 4  then
     say 'Write Error in Table TBGRPB'
  ADDRESS TSO
return Ø

 fourth_phase:
/*************/
ADDRESS "ISPEXEC" "TBSORT TBGRPB "||,
        "FIELDS(TGRPTXT2,C,A)"
ADDRESS "ISPEXEC" "TBTOP TBGRPB"
ADDRESS TSO
dbname = ''
run_no = Ø
cyls_bef = Ø
cyls_aft = Ø
inp_rows = Ø
oup_rows = Ø
```

```
cyls_alloc = Ø
txt_ind = Ø
txt_cnt = Ø
vol_cnt = Ø
txt.    =  ""
txt.Ø   =  Ø
volname.    =  ""
volname.Ø   =  Ø
volptn.    =  ""
volptn.Ø   =  Ø
volavl.   =  Ø
volavl.Ø  =  Ø
partnvol.   =  ""
partnvol.Ø  =  Ø
last_start = Ø
w_wsp_name = ''
cdc_val = Ø
rec1 = ''
rec2 = ''
i1 = Ø
i2 = Ø
i1_next = Ø
i1_start = Ø
if empty_vol = 'YES' then
  do
    run_no = 1
    do i1  =  start_vol to (start_vol  +  max_vol - 1)
       volconv = vol_name_pfx||right(i1,4,'Ø')
       volname.i1 = volconv
       volptn.i1 = ''
       volavl.i1 = vol_cyl_limit
       cyls_bef = cyls_bef  +  volavl.i1
    end
  end
else
    call space_init
say "Total number of volumes  =  "||max_vol
do i1  =  1 to 254
   partnvol.i1 = ''
end
ind = Ø
first_ind = Ø
first_rec = Ø
sp = ''
cyl = ''
vol = ''
prt = ''
i = Ø
j = Ø
k = Ø
if sysdsn("'"fileo"'") ¬= "OK" then
```

```
do
  say 'OUTPUT FILE ' fileo ' DOES NOT EXIST.'
  say
  exit
end
"ALLOC DA('"||fileo||"')    F(DATAOUT) shr"
"EXECIO Ø DISKW DATAOUT (OPEN"
call frth_rtn
rec2 = ''
"EXECIO Ø DISKW DATAOUT (FINIS"
"FREE F(DATAOUT)"
call space_left
return

 frth_rtn:
/*********/
eof = 'NO'
rec1 = ' '
rec2 = ' '
first_ind = Ø
j = Ø
k = Ø
INLIST.    =  ""
INLIST.Ø   =  Ø
OUTLIST.    =  ""
OUTLIST.Ø   =  Ø
filename = ''
do while eof = 'NO'
      ADDRESS "ISPEXEC" "TBSKIP TBGRPB"
      if rc > 4  then
         do
            ADDRESS TSO
            eof = 'YES'
         end
      else
         do
            rec1 = TGRPTXT2
            call process_rtn_frth
         end
end
ADDRESS "ISPEXEC" "TBCLOSE TBGRPB"
return

 process_rtn_frth:
/*****************/

rec2 = ''
cdc_1 = ''
cdc_2 = ''
cdc_3 = ''
create_ind = ''
```

```
dbname = word(rec1,1)
wcntre = word(rec1,2)
inp_rows = inp_rows  +  1
wsp_name = word(rec1,3)
if first_ind = Ø then
   do
      first_ind = 1
         do volptn_ctr =  start_vol to (start_vol  +  max_vol - 1)
            volptn.volptn_ctr = ''
         end
      w_wsp_name = wsp_name
   end
else
   if w_wsp_name <> wsp_name then
      do
         do volptn_ctr =  start_vol to (start_vol  +  max_vol - 1)
            volptn.volptn_ctr = ''
         end
         w_wsp_name = wsp_name
      end
space_name = word(rec1,4)
tc_ind = strip(word(rec1,5))
tix = strip(word(rec1,6))
rec1 = subword(rec1,7)
space_name = strip(space_name)
tot_ptns = words(rec1)
i2_max = tot_ptns
if first_rec = Ø then
  do
    first_rec = 1
    i1_start = start_vol - 1
    last_start = start_vol - 1
  end
if i2_max>1 then
  i1_start = i1_start + 1
if i1_start > (start_vol + max_vol - 1) then
  i1_start = start_vol
i1_next = i1_start
if i2_max = 1 then
  do
    i1_next = last_start + 1
    if i1_next > (start_vol + max_vol - 1) then
      i1_next = start_vol
  end
reca = substr(dbname,1,1Ø)
gen_db = dbname
reca = reca||substr(space_name,1,1Ø)
gen_sp = space_name
rec2  =  ''
do i2  =  1 to i2_max
    cdc_1 = word(rec1,i2)
```

```
parse value cdc_1 with cdc_2 '.' cdc_3 '.' tc_ind
cdc_val  =  value(cdc_2)
if tc_ind = 'T' then
   cdc_val = trunc(cdc_val/15) + 1
do  loop  =  1 to max_vol
    if cdc_val < =  volavl.i1_next then
        do
            if i2 < 1Ø then
                ptn_nr = 'AØØ'||substr(i2,1,1)
            else
                if i2 < 1ØØ  then
                    ptn_nr = 'AØ'||substr(i2,1,2)
                else
                    ptn_nr = 'A'||substr(i2,1,3)
            create_ind = 'N'
            if (si_ind = 'Y' & tix  = 'SI' & volptn.i1_next='') then
               create_ind = 'Y'
            if (si_ind<>'Y'  & tix  = 'SI' ,
                             & pos(ptn_nr,volptn.i1_next) = Ø) then
               create_ind = 'Y'
            if (si_ind =  'Y' & tix<>'SI' ,
                             & pos('A999',volptn.i1_next) = Ø) then
               create_ind = 'Y'
            if (si_ind<>'Y'  & tix<>'SI'  ,
                             & pos(ptn_nr,volptn.i1_next) = Ø) then
               create_ind = 'Y'
            if create_ind = 'Y' then
                do   /* process this partition */
                    partnvol.i2 = volname.i1_next
                    volavl.i1_next   =  volavl.i1_next - cdc_val
                    gen_tc_ind = tc_ind
                    gen_prtp = ptn_nr
                    gen_pqty = cdc_2
                    gen_sqty = cdc_3
                    gen_vol = volname.i1_next
                    oup_rows = oup_rows + 1
                    cyls_alloc = cyls_alloc + cdc_val
                    call gen_ams
                    if (si_ind =  'Y' & tix  = 'SI' ) then
                       volptn.i1_next = 'A999'
                    else
                       volptn.i1_next = volptn.i1_next||ptn_nr
                    last_start = i1_next
                    if ( i2_max>1 & i2 = 1) then
                        i1_start = i1_next
                    i1_next  =  i1_next  +  1
                    if i1_next > (start_vol + max_vol - 1) then
                        call next_line
                    leave
                end  /* process this partition */
            else   /* pos(volptn.i1_next,ptn_nr) <> Ø  */
```

```
                    do
                        i1_next  =  i1_next + 1
                        if i1_next > (start_vol + max_vol - 1) then
                            call next_line
                    end
            end
        else   /* cdc_val > volavl.i1_next   */
            do
               i1_next  =  i1_next + 1
               if i1_next > (start_vol + max_vol - 1) then
               call next_line
            end
        if loop  =  max_vol then
            do
               say 'ALL VOLUMES FULL'
               say 'Space name  '||space_name
               say 'Part  '||i2
               say 'Space '||cdc_val
               exit
            end
      end   /*  loop  =  1 to max_vol */
end  /*  i2  =  1 to i2_max*/
return 0

 next_line:
/**********/
i1_next  = start_vol
return 0

 space_left:
/***********/
"ALLOC DA('"||fvolout||"')    F(VOLOUT) shr"
"EXECIO 0 DISKW VOLOUT (OPEN"
do i1  =  start_vol to (start_vol + max_vol - 1)
   rec2 = ' '
   rec2 = rec2||substr(volname.i1,1,10)
   rec2 = rec2||substr(volavl.i1,1,10)
   cyls_aft = cyls_aft + volavl.i1
   call vols_write
end
rec2 = '**    '||'Run Number:  '||run_no||'                 '
rec2 = rec2||'    Date: '||date()
rec2 = rec2||'  Time:  '||time()
call vols_write
rec2 = '**    '||'Last Database run:  '||dbname
call vols_write
rec2 = '**    '||'Total Cylinders available Before:  '||cyls_bef
call vols_write
rec2 = '**    '||'Total Cylinders available After:  '||cyls_aft
call vols_write
rec2 = '**    '||'Total Input rows processed:  '||inp_rows
```

```
call vols_write
rec2 = '**    '||'Total Output rows processed:  '||oup_rows
call vols_write
rec2 = '**    '||'Total Cylinders allocated:  '||cyls_alloc
call vols_write
do i = 1 to txt_cnt
  rec2 = txt.i
  call vols_write
end
"EXECIO Ø DISKW VOLOUT  (FINIS"
"FREE F(VOLOUT)"
return Ø

 vols_write:
/***********/
outlist.1  =  rec2
"EXECIO 1 DISKW VOLOUT (STEM OUTLIST."
return Ø

 space_init:
/***********/
start_vol = 1
rec3 = ''
nvol = Ø
volinfo.  =  ""
volinfo.Ø  =  Ø
"ALLOC FI(volin) DA('"||fvolin||"') SHR"
IF RC ¬= Ø THEN DO
    SAY 'ERROR IN fvolin ALLOC *' || RC
    EXIT
END
"EXECIO Ø DISKR volin   (OPEN"
"EXECIO * DISKR volin   (STEM volinfo."
"EXECIO Ø DISKR volin   (FINIS"
do nvol  =  1 to volinfo.Ø
   rec3 = volinfo.nvol
   if words(rec3) = 2 then
      do
        vol_cnt = vol_cnt + 1
        volname.nvol = word(rec3,1)
        volavl.nvol = value(word(rec3,2))
        cyls_bef = cyls_bef + volavl.nvol
        say "Volume name  =  "||volname.nvol
        say "Volume space  =  "||volavl.nvol
      end
   else
      do
         txt_cnt = txt_cnt + 1
         txt.txt_cnt = rec3
         if txt_ind = Ø then
            if subword(rec3,1,3) = '**    Run Number:' then
```

```
               do
                  txt_ind = 1
                  run_no = value(word(rec3,4))  +  1
               end
      end
end
max_vol = vol_cnt
return

 gen_ams:
/********/
if gen_tc_ind = 'T' then
   trk_cyl = 'TRACKS'
else
   trk_cyl = 'CYLINDERS'
gen_vol = strip(gen_vol)
prt = gen_prtp
cyl = gen_pqty
if gen_tc_ind = 'C' then
   cyls = cyls + value(cyl)
else
   cyls = cyls + trunc(value(cyl)/15) + 1
call write_vsam

return

 write_vsam:
/***********/
rec2 = '  DEFINE CLUSTER - '
call process_write
rec2 = '  (NAME('||vcat||'.DSNDBC.'||gen_db
rec2 = rec2||'.'||gen_sp||'.IØØØ1.'||prt||') - '
call process_write
rec2 = '  LINEAR - '
call process_write
rec2 = '  '||substr(trk_cyl,1,1Ø)||'('||cyl||'  '||gen_sqty||') - '
call process_write
rec2 = '  VOLUMES ( '||gen_vol||' ) - '
call process_write
rec2 = '  REUSE SHAREOPTIONS (3 3)) - '
call process_write
rec2 = '  DATA - '
call process_write
rec2 = '  (NAME('||vcat||'.DSNDBD.'||gen_db
rec2 = rec2||'.'||gen_sp||'.IØØØ1.'||prt||') - '
call process_write
rec2 = '  ) '
call process_write
rec2 = ' '
call process_write
return Ø
```

```
 process_write:
/**************/
outlist.1  =  rec2
"EXECIO 1 DISKW DATAOUT (STEM OUTLIST."
return Ø

 process_put:
/**************/
ADDRESS TSO
"EXECIO Ø DISKW DATADS (OPEN"

  ADDRESS "ISPEXEC" "TBSORT TBGRPS "||,
          "FIELDS(TDBNAME,C,A,TTSREF,C,A,TTSNAME,C,A,TGRP,C,A)"
  ADDRESS "ISPEXEC" "TBTOP TBGRPS"
  ADDRESS TSO
  puteof = 'NO'
  do while puteof = 'NO'
      ADDRESS "ISPEXEC" "TBSKIP TBGRPS"
      if rc > 4  then
         do
            ADDRESS TSO
            puteof = 'YES'
         end
      else
         do
            ADDRESS TSO
            rec2  =  TDBNAME
            rec2  =  rec2||'  '||TTSNAME
            rec2  =  rec2||'  '||TGRP
            rec2  =  rec2||'  '||TTSREF
            rec2  =  rec2||'  '||TTIX
            rec2  =  rec2||'  '||TNUMPRTS
            rec2  =  rec2||'  '||TSIZPRTS
            rec2  =  rec2||'  '||TSECQ
            rec2  =  rec2||'  '||TCYLS
            call process_writeds
         end
  end
"EXECIO Ø DISKW DATADS  (FINIS"
return

 process_writeds:
/****************/
outlist.1  =  rec2
"EXECIO 1 DISKW DATADS (STEM OUTLIST."
return Ø

 process_get:
/************/
ADDRESS TSO
```

```
"EXECIO Ø DISKR DATADS (OPEN"

  geteof = 'NO'
  do while geteof = 'NO'
      "EXECIO 1 DISKR DATADS (STEM INLIST."
      if rc  =  2 then
         do
            geteof = 'YES'
         end
      else
         do
            rec2 = inlist.1
            TDBNAME   =  word(rec2,1)
            TTSNAME   =  word(rec2,2)
            TGRP      =  word(rec2,3)
            TTSREF    =  word(rec2,4)
            TTIX      =  word(rec2,5)
            TNUMPRTS  =  word(rec2,6)
            TSIZPRTS  =  word(rec2,7)
            TSECQ     =  word(rec2,8)
            TCYLS     =  word(rec2,9)
            ADDRESS "ISPEXEC" "TBADD TBGRPS"
            if rc > 4  then
               msg = 'Duplicate Entry'
            else
               do
                  tbl_entries = tbl_entries + 1
                  tot_tbl_entries = tot_tbl_entries + 1
                  dbarr.tot_tbl_entries = TDBNAME
                  tsarr.tot_tbl_entries = TTSNAME
               end
         end
  end
"EXECIO Ø DISKR DATADS  (FINIS"
return
```

PANEL PDSDTL

```
)ATTR
/*********************************************************************/
/*                                                                   */
/*        Tablespace Dataset Entries Panel                           */
/*                                                                   */
/*********************************************************************/
   @ TYPE(INPUT)  INTENS(LOW)   COLOR(YELLOW)       CAPS(OFF)
   * TYPE(INPUT)  INTENS(LOW)   COLOR(RED)          CAPS(OFF)
   _ TYPE(INPUT)  INTENS(LOW)   COLOR(TURQUOISE)    CAPS(OFF)
   + TYPE(TEXT)   INTENS(LOW)   COLOR(BLUE)      SKIP(ON)
   % TYPE(TEXT)   INTENS(HIGH)  COLOR(WHITE)     SKIP(ON)
   $ TYPE(TEXT)   INTENS(HIGH)  COLOR(RED)       SKIP(ON)
   # TYPE(INPUT)  INTENS(HIGH)  COLOR(TURQUOISE) CAPS(ON)
   ! TYPE(INPUT)  INTENS(HIGH)  COLOR(GREEN)     CAPS(ON) HILITE(USCORE)
)BODY CMD(C)
%--------------------+    Panel for Entries     %-------------------
%OPTION ===_C                                              %SCR-_AMT +  +
+                                                                       +
+                                                                       +
+ %Enter Action (GET, ADD, MOD, DEL, DIS, NXT, PRV, SAVE, GEN &  END)  +
+                                                                       +
+                                                                       +
+ %                    Action      :!z  +                               +
+                                                                       +
+ %   Database name .............. :!z       +                         +
+ %   Tablespace / Index name .... :!z       +                         +
+ %   Group Number ............... :!z  +                               +
+                                                                       +
+ %   Tablespace / Index Indn .... :!z + %(TS, PI, SI)                 +
+ %   Reference Tablespace ....... :!z       + %(for Index Entry)      +
+ %   Partitions in the Group .... :!z  +                               +
+ %   Primary Quantity ........... :!z   +                              +
+ %   Secondary Quantity ......... :!z   +                              +
+ %   Cylinders / Tracks Indn .... :!z+                                 +
+
+ #msg                                                              +   +
+
)INIT
.ZVARS='(acn,dbname,tsname,grp,tix,tsref,numparts,szparts,secprts,cyls)'
)END
```

Sharad K Pande
Senior DBA (USA)

What’s new in DB2 UDB V8 – revisited

The article *What’s new in DB2 UDB V8* in the October 2002 issue of *DB2 Update* suggested that renaming a table is a new function with DB2 UDB Version 8. We have been using this feature since Version 7.1 came on the market!

Daniel Derron
dderron@dis.ch

DB2 news

Syncsort has announced that its Backup Express products for both DB2-UDB and SQL Server now support SAP environments following Integration Assessment of its enterprise data back-up and restore product by SAP’s ICC-Lab.

Backup Express for DB2 supports DB2 back-up and restore in Solaris 2.7 (32/64-bit), AIX 4.3, SIX 5L (32/64-bit), Linux2.2+, HP 11 (32/64-bit), Irix (32/64-bit), True64 5+, and Windows NT/2000.

It uses DB2-supported back-up methods, APIs, and utilities to perform and confirm offline, online non-disruptive back-up and recovery of DB2 databases, tablespaces, and archive logs.
Web address:

For further information contact:
Syncsort, 50 Tice Boulevard, Woodcliff Lake, NJ 0767, USA.
Tel: (201) 93 8200.
URL: http://www.syncsort.com/bexpress/bexpress39.htm.

* * *

Lazy Software has announced its LazyView database aggregration product, which promises users a coherent unified real-time view, without the need for data warehousing and ETL tools.

It uses the capabilities of its associative architecture to present data drawn simultaneously in real time from multiple relational databases such as DB2, Oracle, and SQL Server. Data from target databases is aggregated to combine similar types of data and to resolve semantic conflicts, and can be referenced, searched, and queried via the Sentences associative database.

Data can also be augmented locally to extend the functionality of target databases and to support ‘what if’ scenarios. Data added in this way persists in Sentences’ own data store.

The ability to operate in real time means, says the vendor, that users are always assured that what they’re seeing via LazyView is current, and it can operate on very large databases without requiring additional storage capacity.

It also resolves semantic conflicts, recognizing that what one database calls a ‘customer’ is the same type of thing that another calls a ‘client’.

For further information contact:
Lazy Software, Gemini House, Mercury Park, Wycombe Lane, Wooburn Green, Bucks HP10 0HH, UK.
Tel: (1628) 642300.
URL: http://www.lazysoft.com.

* * *

IBM has announced DB2 OLAP Server for iSeries, V8.1, based on Hyperion Essbase 6.5, which incorporates improvements in the areas of scalability, performance, ease of use, and administration.

A logical cube now can be defined to have a lower portion of the cube reside in the relational database. The OLAP Server retrieves the cells from the relational database as if they physically reside on the cube storage.

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

* * *