93 DB2

July 2000

update

In this issue

© Xephon plc 2000

DB2 Update

Published by

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

North American office

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

Subscriptions and back-issues

A year's subscription to *DB2 Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1997 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

DB2 Update on-line

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

Editor

Trevor Eddolls

Disclaimer

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

Contributions

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

© Xephon plc 2000. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.

Optimizing Dynamic SQL

My article *Dynamic SQL for fuzzy SELECT, DB2 Update*, Issue 89, March 2000, recommended using Dynamic SQL for processing user fuzzy SELECT queries. Its recommendations included using EXPLAIN and RUNSTATS. This article expands on those suggestions and provides additional ideas.

DYNAMIC SQL

A varying-list SELECT statement was recommended because the columns needed by user fuzzy queries are *unknown.* A user could ask for best students or worse instructors or most generous alumni. Best student is defined by GPA (Grade Point Average) and attendance; instructor by individual course ratings, average course ratings, and attendance; alumni by total contribution and contributions to specific appeals. Simplified logical tables are:

```
STUDENT(STUDENT_ID, surname, gpa, total_absence_days)

    STUDENT_COURSE(STUDENT_ID, COURSE_ID, grade, absence_days)

INSTRUCTOR(INSTRUCTOR_ID), surname, avg_rating, total_absence_days)

  INSTRUCTOR_COURSE(INSTRUCTOR_ID, COURSE_ID, rating, absence_days)

ALUMNUS(ALUMNUS_ID, surname, total_contribution)

    ALUMNUS_APPEAL(ALUMNUS_ID, APPEAL_ID, contribution)
```

Best, worse, and most generous require different column names (identical column names require table qualification when used in single SELECT) and a different quantity of columns, making varying-list SELECT mandatory.

Users must define all fuzzy predicate parameters before they can be used. Best student requires a GPA > 91.69 *and* total_absence_days <8.88 (see *Fuzzy SELECT, DB2 Update*, Issue 87, January 2000 for an explanation of how 91.69 and 8.88 were calculated). The SELECT statement is:

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
SELECT   student_id, surname, gpa, total_absence_days
FROM     student
WHERE    gpa > 91.69
AND      total_absence_days < 8.88
```

A good student is defined as GPA > 86.33 and < 91.69 with total_absence_days > 8.88 and < 13.13. This SELECT needs a BETWEEN which is normal for fuzzy SELECT statements:

```
SELECT   student_id, surname, gpa, total_absence_days
FROM     student
WHERE    gpa                 BETWEEN 86.34 AND 91.68
AND      total_absence_days  BETWEEN 8.89 AND 13.12
```

INDEXES AND DESCRIBE

Fuzzy predicate columns such as student.gpa, student.total_absence_days, instructor.avg_rating, instructor.total_absence_days, instructor_course.rating, alumnus.total_contribution, and alumnus_appeal.contribution should be indexed for efficiency. DESCRIBE provides an easy way to identify table indexes.

```
DESCRIBE
INDEXES FOR TABLE student  --must be fully qualified name or alias
                           --in form of schema.table.name; schema
                           --is user name under which table or view
                           --was created
SHOW DETAIL                --includes column name in output
```

CREATE INDEX for any column involved with a fuzzy predicate and then run RUNSTATS.

RUNSTATS

RUNSTATS should be run on all tables containing columns required by the translated fuzzy predicate, ie best student needs student.gpa and student.total_absence_days. The recommended statement is:

```
RUNSTATS ON TABLE student --must be fully-qualified name or alias
WITH DISTRIBUTION         --number of most frequent values
                          --collected is defined by num_freqvalues
                          --number of quantiles collected is
                          --defined by num_quantiles; both should
                          --be set with user assistance
AND INDEXES ALL           --updates stats on table and all indexes
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
DETAILED                --calculates extended index stats
SHRLEVEL REFERENCE      --restricts other users to read only
```

COMMIT should be issued after the execution of release locks.

(You may want to create new static access paths after RUNSTATS by rebinding the packages using BIND.)

DB2LOOK

Prudence dictates that newly-translated fuzzy predicates be tested for correctness and efficiency on a test database before running against the production database. Use db2look to match the test database catalog statistics to the production so that the optimizer uses the same catalog statistics while evaluating test SQL. The recommended basic statement is:

```
db2look    --has many parameters that must be evaluated and set
           --by the DBA.
-d DBname  --alias name of production database
-a         --ignores -u Creator parameter; generates stats for all
           --database users
```

RANDOM

You can use the random nested table expression to populate the test database. SQL template (this requires customization based on your specific environment to provide a proper population).

```
INSERT INTO   student_test
SELECT        *
FROM          (SELECT INTEGER(RAND()* "n")   --n is row quantity
              FROM  student)
```

You now have a test database with the current statistics from the production database and sample SQL statements such as:

```
SELECT    student_id, surname, gpa, total_absence_days
FROM      student
WHERE     gpa                BETWEEN 86.34 AND 91.68
AND       total_absence_days BETWEEN 8.89 AND 13.12
```

The SELECT above is very simple. Many fuzzy SELECT statements require complex objects such as CASE, compound SQL, CORRELATION, COVARIANCE, GROUPING operands of grouping-sets or super-groups including CUBE and ROLLUP, join

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

and joined-table operands of left outer join or right outer join or full outer join, REGRESSION, STDDEV (standard deviation), subselects including correlated, UNION, and many more. The sequence of objects or operations can drastically affect efficiency. It is essential to EXPLAIN each fuzzy SELECT.

EXPLAIN

The author recommends using DB2 Visual Explain (DB2VE) as your SQL performance tool, if it is available. The main reasons include:

- DB2VE offers suggestions for improving performance of your SQL queries or statements.
- You can change the SQL to fit the suggestions and then dynamically EXPLAIN again to see if the access path is improved by the change.
- DB2VE uses graphs to show access paths, eliminating the need to look up the values in the plan table, which most people find difficult and time-consuming. Comparing graphs of the same SQL statement in different formats quickly reveals which is best. Version 6 graphs can provide statement costs in milliseconds and service units.
- You can browse the current subsystem parameters' values and get their meaning.

In any case, use EXPLAIN to get the optimum performance for any SQL statement but particularly for dynamic SQL.

SUMMARY

The following points summarize the best way to proceed:

- Translate user fuzzy predicates into column names.
- Determine whether there is an index for each column name (DESCRIBE is recommended). If not, CREATE INDEX. EXPLAIN will show whether the index is useful and, if it is not needed, it can be DROPped without affecting the database.
- CREATE a test database:

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

  - Use db2look to provide production database catalog statistics.
  - Populate the test database with production rows. Random is a possible population tool.
- Write a fuzzy SELECT statement:
  - Use EXPLAIN (DB2 Visual Explain is recommended) for optimization.
- Implement the optimized fuzzy SELECT on the production database.

WHY

Implementing fuzzy SELECT provides users with more relevant answers, allowing them to make better decisions that can *dramatically* and *positively* affect the bottom line.

It allows DBAs to become organizational heroes since they are satisfying important user needs that have been neglected for decades.

Fuzzy logic and fuzzy SELECT should be in every enterprise toolbox!

Eric Garrigue Vesely
Principal
Analyst Workbench Consulting (Malaysia)
© Xephon 2000

Reorganizing the DB2 catalog and directory

Since Version 4 of DB2 it has been possible for DBAs to expediently reorganize the DB2 catalog and DB2 directory in a systematic manner using the native IBM REORG utility. This article will describe how to reorganize the DB2 catalog and provide implementation tips and advice.

WHAT ARE THE DB2 CATALOG AND DB2 DIRECTORY?

The DB2 catalog is the central repository for DB2 object and user meta data. DB2 is constantly referring to that meta data as it processes

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

applications and queries. The physical condition of the tablespaces and indexes that comprise the DB2 catalog is therefore a major component in overall DB2 subsystem performance.

Likewise, the DB2 directory contains internal control structures such as DBDs, skeleton cursor tables, and skeleton package tables that can be accessed only by DB2 itself. The information in the DB2 directory is critical for database access, utility processing, plan and package execution, and logging. It is quite obvious that efficient access to this critical information should be of paramount importance.

Prior to DB2 Version 4, reorganization of the DB2 catalog and DB2 directory was not possible using the REORG utility. The only option for any type of 'reorganization' activity was to run the RECOVER INDEX utility on DB2 catalog indexes. This rebuilt the indexes, but had no impact on the underlying data housed in actual physical tablespaces.

As of DB2 Version 4 it is permitted to execute the REORG utility on tablespaces and indexes in the DB2 catalog database (DSNDB06) and on specific tablespaces (SCT02, SPT01, and DBD01) in the DB2 directory database (DSNDB01).

WHEN SHOULD THE DB2 CATALOG AND DIRECTORY BE REORGANIZED?

To determine when to reorganize the system catalog, DBAs can use the same basic indicators used to determine whether application tablespaces should be reorganized. Although it always has been a wise course of action to execute RUNSTATS on the DB2 catalog tablespaces, it becomes even more important now that these tablespaces can be reorganized. The RUNSTATS utility collects statistical information that is used by the optimizer to generate access paths. Additionally, these statistics can be analysed to determine when a REORG should be run. When RUNSTATS is run for a catalog tablespace, the statistics about that system catalog tablespace are gathered and then stored in the DB2 catalog tables themselves! The table contained in Figure 1 should serve as a basic guide to help in determining when to reorganize system catalog tablespaces and indexes.

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

Note that the data in Figure 1 can be used for data contained in application tablespaces and indexes as well as DB2 catalog tablespaces and indexes. For application tablespaces, though, you may also wish to use the PERCDROP column in SYSIBM.SYSTABLEPART to determine when to reorganize application tablespaces. PERCDROP does not apply to DB2 catalog tablespaces because tables cannot be dropped from the DB2 catalog.

Column	*Catalog table*	*Object*	*Impact*
NEAROFFPOS	SYSIBM.SYSINDEXPART	TABLESPACE	+
FAROFFPOS	SYSIBM.SYSINDEXPART	TABLESPACE	++++
CLUSTERRATIO	SYSIBM.SYSINDEXES	INDEX	- - - - -
NEARINDREF	SYSIBM.SYSTABLEPART	INDEX	+
FARINDREF	SYSIBM.SYSTABLEPART	INDEX	++++
LEAFDIST	SYSIBM.SYSINDEXPART	INDEX	+++

Figure 1: Reorganization indicators

The column and table name where the statistic can be found is given in the first two columns of the chart. The third column indicates whether the statistic is applicable for a tablespace or an index. The fourth column gives an indication of the impact of the statistic. A plus (+) sign indicates that you should REORG more frequently as the value in that column gets larger. A minus (-) sign indicates that you should REORG more frequently as the value gets smaller. As the number of '+' or '-' signs increases, the need to REORG becomes more urgent. For example, as FAROFFPOS gets larger, the need to REORG is more urgent, as indicated by the five plus (+) signs.

For the SYSDBASE, SYSVIEWS, and SYSPLAN catalog tablespaces, the value for the FAROFFPOS and NEAROFFPOS columns of SYSINDEXPART can be higher than for other tablespaces before they need to be reorganized.

In addition to the guidelines in Figure 1, consider DB2 catalog and DB2 directory reorganization in the following situations:

- To reclaim space and size tablespaces appropriately when DB2

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

catalog and directory datasets are not using a significant portion of their allocated disk space (PRIQTY).

- When it is necessary to move the DB2 catalog and directory to a different storage device.
- When the DB2 catalog and directory datasets contain a large number of secondary extents.

The bottom line, however, with regard to catalog reorganization is not to go overboard. Let's face it, we could not REORG the DB2 catalog at all for over a decade, and we somehow managed to get by. Although reorganizing the DB2 catalog and directory can bring performance gains to a DB2 environment, you do not need to schedule a weekly or monthly catalog reorganization job. Instead, monitor the key statistics and REORG only as needed.

SYNCHRONIZING SYSTEM CATALOG REORGANIZATION

It is a more difficult prospect to determine when the DB2 directory tablespaces should be reorganized. The RUNSTATS utility does not maintain statistics for these 'tablespaces' like it does for the DB2 catalog.

However, it is possible to base the reorganization of the DB2 directory tablespaces on the reorganization schedule of the DB2 catalog tablespaces. In fact, in certain situations, it is imperative that specific DB2 directory tablespaces are reorganized when a 'companion' DB2 catalog tablespace is reorganized. The chart contained in Figure 2 provides information on keeping the DB2 catalog and DB2 directory tablespaces 'in sync'.

When you REORG:	*Be sure to also REORG:*
DSNDB06.SYSDBASE	DSNDB01.DBD01
DSNDB06.SYSPLAN	DSNDB01.SCT02
DSNDB06.SYSPKAGE	DSNDB01.SPT01

Figure 2: Reorganization indicators

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

These tablespaces are logically related and DB2 requires that you reorganize them at the same time to keep them synchronized.

DB2 CATALOG AND DIRECTORY REORGANIZATION DETAILS

The DB2 catalog is composed of 13 tablespaces and 63 tables all in a single database, DSNDB06 (refer to Figure 3). There are six DB2

Database name: DSNDB06
Tablespaces:

SYSCOPY	Contains image copy information
SYSDBASE	Contains database object information
SYSDBAUT	Contains database and database authority information
SYSDDF	Contains data distribution details
SYSGPAUT	Contains resource authority information
SYSGROUP	Contains storage group information
SYSOBJ	Contains object/relational information
SYSPLAN	Contains plan information
SYSPKAGE	Contains package information
SYSSTATS	Contains optimization statistics
SYSSTR	Contains translation and check constraint information
SYSUSER	Contains user authority information
SYSVIEWS	Contains view information

Figure 3: DB2 catalog tablespaces

directory tablespaces in a separate database, DSNDB01 (refer to Figure 4). DB2 has different rules for different sets of these tablespaces. There are three groupings of tablespaces, those that:

- Cannot be reorganized at all.
- Can be reorganized using normal REORG procedures.
- Can be reorganized using special REORG procedures.

There are only two tablespaces in the first grouping of tablespaces that cannot be reorganized at all – DSNDB01.SYSUTILX and DSNDB01.SYSLGRNX. Do not attempt to reorganize these tablespaces because DB2 will not permit it.

The second grouping of tablespaces are those that the REORG utility

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

Database name: DSNDB01
Tablespaces:

DBD01	Contains database descriptor information
SCT01	Contains skeleton cursor table information
SPT02	Contains skeleton package table information
SYSLGRNX	Contains recovery log range information
SYSUTILX	Contains utility processing information

Figure 4: DB2 directory tablespaces

processes as it would any other tablespace:

- DSNSB06.SYSCOPY
- DSNDB06.SYSDDF
- DSNSB06.SYSGPAUT
- DSNDB06.SYSOBJ
- DSNSB06.SYSPKAGE
- DSNSB06.SYSSTATS
- DSNSB06.SYSSTR
- DSNSB06.SYSUSER
- DSNSB01.SCT02
- DSNSB01.SPT01.

The third, and final, grouping of tablespaces must be processed differently from other tablespaces:

- DSNDB06.SYSDBASE
- DSNDB06.SYSDBAUT
- DSNDB06.SYSGROUP
- DSNDB06.SYSPLAN
- DSNDB06.SYSVIEWS
- DSNDB01.DBD01.

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

These six tablespaces require special ‘handling and care’. Because they have a different internal configuration from most other tablespaces, a different calculation is required for the size of the unload dataset (SYSREC) used during the REORG utility. These tablespaces contain internal links. Links are internal pointers that tie the information in their tables together hierarchically. A link can be thought of as a type of parent-child relationship. Because of these links, the BUILD and SORT phases of the REORG utility are not executed.

The WORKDDN, SORTDATA, SORTDEVT, and SORTNUM options are ignored when reorganizing these tablespaces.

Also, the REORG utility cannot be restarted from the last checkpoint when used against these six tablespaces. Instead, it must be restarted from the beginning of the PHASE.

Also, as mentioned before, a different set of steps must be executed during reorganization for these tablespaces.

STEPS TO REORG THE SIX ‘SPECIAL’ TABLESPACES

There are special requirements for reorganizing the six ‘different’ tablespaces, namely:

- DSNDB06.SYSDBASE
- DSNDB06.SYSDBAUT
- DSNDB06.SYSGROUP
- DSNDB06.SYSPLAN
- DSNDB06.SYSVIEWS
- DSNDB01.DBD01.

For these tablespaces, REORG will reload the indexes in addition to the tablespace during the reload phase. Therefore, REORG does not need to store the index keys in a work dataset for sorting. The following steps should be used when reorganizing these six ‘different’ tablespaces:

1 Calculate the size of the unload dataset (SYSREC).

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

The SYSREC dataset for the 'special' tablespaces has a different format from the other tablespaces. This causes a special calculation to be required to determine its size. The equation to use is:

```
DATASET SIZE IN BYTES = (28 + LONGROW) * NUMROWS
```

NUMROWS is the number of rows to be contained in the dataset and LONGROW is the length of the longest row in the tablespace. The value for LONGROW can be determined by running the following SQL statement:

```
SELECT    MAX(RECLENGTH)
FROM SYSIBM.SYSTABLES
WHERE     DBNAME = 'DSNDBØ6'
AND  TSNAME = 'name of tablespace to REORG'
AND  CREATOR = 'SYSIBM';
```

2 Ensure incompatible operations are not executing.

3 Start database DSNDB01 and DSNDB06 for read-only access.

4 Run QUIESCE and DSN1CHKR utilities.

5 Take a full image copy of the entire DB2 catalog and directory tablespaces.

6 Start DSNDB01 and DSNDB06 for utility access.

7 Execute the REORG utility.

8 Take a full image copy of the entire DB2 catalog and directory tablespaces.

9 Start tablespace and associated indexes for read/write access.

Additionally, keep in mind that the WORKDDN, SORTDATA, SORTDEVT, SORTNUM, SORTKEYS, COPYDDN, and RECOVERYDDN options are ignored for these six 'special' DB2 catalog and directory tablespaces.

STEPS TO REORG REGULAR TABLESPACES

The following steps should be used when reorganizing the remaining 'regular' system catalog and directory tablespaces:

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

- Calculate the size of the unload dataset (SYSREC) using the normal calculation:

  ```
  DATASET SIZE IN BYTES = LONGROW * NUMROWS
  ```

  In this case it is unnecessary to add the additional 28 bytes to the length of the longest row. This is because these system catalog tablespaces do not utilize links.

- Ensure that incompatible operations are not concurrently executing (see the next section for an explanation of incompatible operations).
- Start the tablespace and its associated indexes for read-only access.
- Run CHECK INDEX on all indexes associated with the tablespace that is being reorganized.
- Take a full image copy of the entire DB2 catalog and directory tablespaces.
- Start the tablespace and its associated indexes for utility access.
- Execute the REORG utility.
- Take a full image copy of the entire DB2 catalog and directory tablespaces.
- Start the tablespace and any associated indexes for read/write access.

These steps should be familiar to you because they closely follow the steps executed during the reorganization of an application data tablespace. There are several additional steps required as an added precaution because of the critical nature of the DB2 catalog and directory.

CATALOG REORGANIZATION RESTRICTIONS

In addition to the procedures outlined previously, there are several restrictions on the manner in which the REORG TABLESPACE utility can be used with system catalog tablespaces. Firstly, recall that the SYSLGRNX and SYSUTILX tablespaces in the DB2 directory cannot be reorganized at all.

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

Furthermore, when reorganizing the DB2 catalog (DSNDB06) and DB2 directory (DSNDB01) tablespaces, the following options cannot be used:

- The UNLOAD ONLY option is not permitted.
- The LOG YES option is not permitted because image copies are explicitly required following a DB2 catalog and/or DB2 directory reorganization.

Also, DB2 tracks the reorganization of two specific tablespaces differently from any other. Generally, DB2 will record the reorganization of any tablespace in the SYSIBM.SYSCOPY system catalog table. However, DB2 records the reorganization of the DSNSB06.SYSCOPY and DSNDB01.DBD01 tablespaces in the log instead. Therefore, the REORG utility will scan logs to verify that an image copy is available. If an image copy is not found, the REORG will request archive logs.

Finally, in many 24 x 7 environments, it may be necessary to reorganize the system catalog and dictionary while it is being accessed. However, because of the central nature of the system catalog and directory to the operation of DB2, there are restrictions on concurrent activity during catalog reorganization. These restrictions on concurrent activity are listed below:

- ALTER, DROP, and CREATE statements cannot be executed during the reorganization of any DB2 catalog or DB2 directory tablespace with the exception of SYSIBM.SYSSTR and SYSIBM.SYSCOPY.
- The BIND and FREE commands cannot be issued when the following tablespaces are being reorganized: SYSIBM.SYSDBAUT, SYSIBM.SYSDBASE, SYSIBM. SYSGPAUT, SYSIBM.SYSPKAGE, SYSIBM.SYSPLAN, SYSIBM.SYSSTATS, SYSIBM.SYSUSER, and SYSIBM. SYSVIEWS.
- No DB2 utility can be running while SYSIBM.SYSCOPY, SYSIBM.SYSDBASE, SYSIBM.SYSDBAUT, SYSIBM. SYSSTATS, and/or SYSIBM.SYSUSER are being reorganized.

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

- No plan or package may be executed during the reorganization of SYSIBM.SYSPLAN and SYSIBM.SYSPKAGE.
- The GRANT and REVOKE statements cannot be issued when REORG is being run on SYSIBM.SYSDBASE, SYSIBM.SYSDBAUT, SYSIBM.SYSGPAUT, SYSIBM.SYSPKAGE, SYSIBM.SYSPLAN, and/or SYSIBM.SYSUSER.

SYNOPSIS

The ability to reorganize the DB2 catalog and directory tablespaces provides the DBA with a potent tool for system tuning. If you have not yet started to run RUNSTATS on the system catalog tablespaces, begin to do so immediately. This will enable you to determine when your DB2 catalog and directory will need to be reorganized. Good luck and happy tuning.

Craig S Mullins
Director, DB2 Technology Planning
BMC Software (USA)
© Craig S Mullins 2000

Call for papers

Why not share your expertise and earn money at the same time? *DB2 Update* is looking to swell the number of contributors who send in technical articles, hints and tips, and utility programs, etc. We would also be interested in articles about performance and tuning, and information and tips for DB2 DBAs. If you have an idea for an article, or you would like a copy of our *Notes for Contributors*, contact the editor, Trevor Eddolls, at any of the addresses shown on page 2.

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

Distributing index pieces to different volumes

You can decrease I/O contention on tablespaces by defining them as partitioned and spreading the partitions across various I/O paths. Also you can do the same thing for partitioned indexes, but you could have contention on non-partitioned indexes (NPI).

In Version 5, IBM announced the keyword PIECESIZE. By using this keyword you can divide a huge index into small pieces.

With partitioned indexes, you can assign different storage groups to different partitions. With this definition, you can split them up on different volumes. But you can't do this for NPIs. For NPIs you can assign only one storage group to the index.

You can solve this problem by moving the pieces of your index onto different volumes manually. But when you REORG or recover this index, all the pieces go to the same volume and you have to move index pieces again.

To circumvent this problem, I developed three REXX programs. The first program, NPICHECK, works every day and checks for the volumes used that have a PIECE of the index and which are still in the storage group. If it finds a volume like this, it creates a DDL to remove it from the storage group. The second program, NPISTART, works before a REORG or recover of the index. It creates DDL to add volumes that are physically used by the index but not in the storage group of the NPI. The last program, NPIMOVE, does the main process. It QUIESCEs necessary tablespaces to write the pages that are in the bufferpool. Then it stops related DB2 objects to prevent update activity. Thirdly, it finds the necessary PIECES, which have to be moved, and moves them. It removes physically-used volumes from the storage group to prevent usage of the same volume again. And finally it starts stopped objects. The program does not do the process internally. It just creates the necessary statements.

NPICHECK

The program NPICHECK executes a query on the catalog tables and

 © 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

it finds all NPIs with VCAT, storage group, and volumes. Then, for all NPIs, it finds cluster names by executing the LISTC LEVEL command; it then finds the physical volume names of the clusters by executing the LISTC ENT VOLUMES command. If the storage group volume is used physically, the program creates a DDL to REMOVE it using the command ALTER STOGROUP REMOVE VOLUMES. The program writes all REMOVE DDLs to DDLOUT DD. If the program doesn't find any volumes to remove from the storage group, DDLOUT will include just a COMMIT command.

```
/***REXX*************************************************************/
/*                                                                   */
/*  NPICHECK: Used to REMOVE physically-used volumes                 */
/*  ========= from a stogroup of a non-partitioned index             */
/*            with a PIECESIZE.                                      */
/*                                                                   */
/* Description:  S1 - Get the names of indexspaces that have a       */
/* ============       PIECESIZE value.                               */
/*               S2 - For all Indexspaces                            */
/*                    S2 - i.  Get the physical volume names that are*/
/*                             used by the indexspace PIECE.         */
/*                    S2 - ii. If the volume name is EXIST in STOGRP */
/*                             and the name of the volume is not     */
/*                             DB2EX1 (default volume for all        */
/*                             storage groups).                      */
/*               S3 - Write all DDL into DD called DDLOUT.           */
/*                                                                   */
/*********************************************************************/
ADDRESS TSO                   /* PGM will work in TSO environment,    */
"PROFILE NOPREFIX"            /* without TSO prefix.                  */

SYSINCnt = 0                  /* DDL line counter                     */

OldIXClust = ''               /* Old index cluster name               */
                              /* Get the necessary info about the non-*/
                              /* partitioned indexes with a PIECESIZE */
                              /* value                                */
SQLQUERY = ,
  "SELECT C.VCATNAME, A.DBNAME, A.INDEXSPACE, B.SGNAME, B.VOLID" ||,
  "  FROM SYSIBM.SYSINDEXES A, SYSIBM.SYSVOLUMES B,            " ||,
  "       SYSIBM.SYSINDEXPART C                                " ||,
  "  WHERE                                                     " ||,
  "    C.PARTITION = 0 AND A.NAME = C.IXNAME                   " ||,
  "    AND A.CREATOR = C.IXCREATOR                             " ||,
  "    AND C.STORNAME = B.SGNAME                               " ||,
  "    AND A.PIECESIZE > 0 AND A.PIECESIZE < 2097152           " ||,
  "ORDER BY 1 ASC, 2 ASC, 3 ASC                                "
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
ADDRESS LINK "REXXSQL"
If RC = Ø & SQLSTATE = Ø & _NROWS > Ø Then
  Do                          /* SQL executed successfully           */
    Do RowCnt = 1 to _NROWS
      IxClust = Strip(VCATNAME.RowCnt) || ".DSNDBC." ||,
                Strip(DBNAME.RowCnt) || '.' ||,
                Strip(INDEXSPACE.RowCnt) /* The cluster name of NPI  */
      If IxClust ¬= OldIXClust Then      /* Is it new cluster        */
        Do
          VolStr = ''                    /* To keep physical volume  */
                                         /* names of the Cluster     */
          OldIXClust = IXClust
          Call GetVolumes                /* Get the volume names     */
        End
      If Pos(VolID.RowCnt,VolStr) ¬= Ø Then /* Is Stogroup Volume    */
                                         /* used physically          */
        Do
          SYSINCnt = SYSINCnt + 1        /* Yes, prepare a DDL       */
          SYSINLine.SYSINCnt = " "       /* to REMOVE it from the */
          SYSINCnt = SYSINCnt + 1        /* Stogroup                 */
          SYSINLine.SYSINCnt = "  ALTER STOGROUP " ||,
            Strip(SGNAME.RowCnt) || " REMOVE VOLUMES(" ||,
            Strip(VOLID.RowCnt) || ") ;"
          SYSINCnt = SYSINCnt + 1
          SYSINLine.SYSINCnt = " "
        End
    End
  End

SYSINCnt = SYSINCnt + 1                  /* To COMMIT the unit of */
SYSINLine.SYSINCnt = " "                 /* work                  */
SYSINCnt = SYSINCnt + 1
SYSINLine.SYSINCnt = "  COMMIT ; "
SYSINCnt = SYSINCnt + 1
SYSINLine.SYSINCnt = " "

"EXECIO * DISKW DDLOUT (FINIS STEM SYSINLine." /* Write down the     */
                          /* prepared DDL into a DD called DDLOUT */
Exit(ØØ)                  /* Leave the program                    */

/*****************-------------------------------------*****************/
/* GETVolumes: Get the volume names of the NPI that are physically   */
/*             used.                                                 */
/*****************-------------------------------------*****************/
GetVolumes: Procedure,
  EXPOSE VolStr IXClust

ReturnCode = OutTrap('Mesaj.')  /* Execute LISTC LVL to get the PIECE*/
"LISTC LVL(" || IXClust || ")"  /* names of the NPI                  */
Dummy = OutTrap('OFF')
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
Do I = 1 to Mesaj.Ø
  If SubStr(Mesaj.I,1,7) = "CLUSTER" Then /* For each PIECE get the  */
    Do                                     /* volume name            */
      IndexDSet = Strip(SubStr(Mesaj.I,17,44)) /* PIECE name         */

      ReturnCode = OutTrap('ListCOut.')
      "LISTC ENT(" || IndexDSet || ") VOLUME"
      Dummy = OutTrap('OFF')

      Do J = 1 to ListCOut.Ø
        If SubStr(ListCOut.J,8,6) = "VOLSER" Then
          Do
                                       /* DB2EX1 is the default    */
                                       /* volume for all stogroups */
            Volume = SubStr(ListCOut.J,26,6)
            If Pos(Volume,VolStr) = Ø & Volume ¬= "DB2EX1" Then
              VolStr = VolStr || Volume || ' '
          End
      End
    End
End
Return                                        /* Go back to caller */
```

JCL FOR NPICHECK

```
//JNPICHCK JOB ,NPICHECK,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//NPICHECK EXEC PGM=IKJEFTØ1,PARM='%NPICHECK'
//SYSEXEC  DD DISP=SHR,DSN=SØØØ.COMM.REXX
//DDLOUT   DD DISP=(NEW,PASS),SPACE=(TRK,(1,1)),UNIT=SYSALLDA,
//         DCB=(LRECL=8Ø,RECFM=FB,DSORG=PS,BLKSIZE=27200)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//REMOVE   EXEC DYNSQL
//SYSIN    DD DISP=(OLD,DELETE),DSN=*.NPICHECK.DDLOUT
/*
```

NPISTART

The program NPISTART reads the DD called INDEX. In this DD there are NPI cluster names (four qualifiers; the format is high_level_qualifier.DSNDBC.database_name.indexspace_name; at each row there is only one name). Then it finds the storage group of the NPI. By executing another query, it finds the volumes of the storage group. By executing the LISTC LVL command, the program finds the cluster names of the NPI; and by executing the LISTC ENT VOLUMES command on each cluster, it finds the physical volume

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

names. Then it compares the physical volume names with the storage group volume names. If a volume does not exist in a storage group, the program creates DDL to ADD this volume to the storage group. The created DDL will be written to a DD called SQLOUT.

```
/***REXX*************************************************************/
/*                                                                  */
/* NPISTART:  Used to add the physical volume names of non-part.    */
/* =========  index with PIECESIZE to the stogroup.                 */
/*                                                                  */
/* Description: S1 - Read DD called INDEX to get the names of NPIs. */
/* ============      Format of the input is cluster name of the NPI */
/*                   till INDEXSPACE name.                          */
/*              S2 - Get the physical volume names of the different */
/*                   datasets (PIECE) of the same NPI.              */
/*              S3 - Prepare necessary DDL to add the volumes into  */
/*                   Stogroup of the NPI.                           */
/*              S4 - If there is another NPI go to S1.              */
/*              S5 - Write prepared DDL into DD called SQLOUT.      */
/*                                                                  */
/* Return Codes:                                                    */
/* ==============                                                   */
/* RC   Description                                                 */
/* --   ------------------------------------------------------------ */
/* ØØ   Normal completion.                                          */
/* Ø8   No STOGROUP name found for NPI.                             */
/* 12   No NPI name supplied.                                       */
/*******************************************************************/
ADDRESS TSO
"PROFILE NOPREF"

"EXECIO * DISKR INDEX (FINIS STEM IndexLine." /* Get the names of NPI*/
If RC ¬= Ø | IndexLine.Ø = Ø Then
  Do                                  /* No NPI name found          */
    Say "ERROR 1: At least one NPI should be supplied"
    Exit(12)                          /* Leave the program RC = 12  */
  End

OutCnt = Ø                            /* Output line counter ( DDL ) */

Do LineCnt = 1 to IndexLine.Ø         /* Continue for all NPIs       */

  Obje = Strip(SubStr(Strip(IndexLine.LineCnt),1,44)) /* Name of NPI */
  ObjeII = Translate(Obje,' ','.')    /* To get DBName & Indexspace  */


  SQLQUERY = "SELECT STORNAME " ||,
             "  FROM SYSIBM.SYSINDEXPART, SYSIBM.SYSINDEXES " ||,
             "  WHERE NAME = IXNAME AND CREATOR = IXCREATOR AND " ||,
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
           "    DBNAME = '" || Word(ObjeII,3) || "' AND " ||,
           "    INDEXSPACE = '" || Word(ObjeII,4) || "'"
ADDRESS LINK "REXXSQL"
If RC = Ø & SQLSTATE = Ø & _NROWS > Ø Then
  Stogroup = Strip(STORNAME.1)      /* Get the STOGROUP of NPI     */
Else
  Do
    Say "ERROR 2: Couldn't get the STOGROUP for " || Obje
    Say "         SQLSTATE = " || SQLSTATE
    Exit(Ø8)
  End
                                    /* Get the volumes of Stogroup */
SQLQUERY = "SELECT VOLID " ||,
           "  FROM SYSIBM.SYSVOLUMES " ||,
           "  WHERE SGNAME = '" || Stogroup || "'"
ADDRESS LINK "REXXSQL"
STGStr = ''
Do Cnt = 1 to _NROWS          /* Create volume string for STOGROUP */
  STGStr = STGStr || VolID.Cnt || ' '
End

ReturnCode = OutTrap('LVLMsg.')     /* Execute LISTC LVL for NPI   */
"LISTC LVL(" || Obje || ")"
Dummy = OutTrap('OFF')

VolStr = ''                         /* To keep physical Volsers    */

Do I = 1 to LVLMsg.Ø
  If SubStr(LVLMsg.I,1,7) = "CLUSTER" Then
    Do
      IndexDSet = Strip(SubStr(LVLMsg.I,17,44)) /* Index Dset name */

                                      /* Execute LISTC ENT to get*/
      ReturnCode = OutTrap('ListCOut.') /* names of volsers        */
      "LISTC ENT(" || IndexDSet || ") VOLUME"
      Dummy = OutTrap('OFF')

      Do J = 1 to ListCOut.Ø          /* Trap volsers            */
        If SubStr(ListCOut.J,8,6) = "VOLSER" Then
          Do
            Volume = SubStr(ListCOut.J,26,6)

               /* If volume is not added to physical volume list or*/
               /* not added to STOGROUP volume list then add       */

            If Pos(Volume,VolStr) = Ø &,
               Pos(Volume,STGStr) = Ø Then
              VolStr = VolStr || Volume || ' '
          End
      End                             /* EOF ListCOut trap       */
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
      End
    End                                    /* EOF LVLMsg trap         */

                                           /* Add new volumes to DDL  */
    OutCnt = OutCnt + 1
    OutLine.OutCnt = ' '
    OutCnt = OutCnt + 1
    OutLine.OutCnt = "  ALTER STOGROUP " || Stogroup
    OutCnt = OutCnt + 1
    OutLine.OutCnt = "     ADD VOLUMES(" || SubStr(VolStr,1,6) || ","
    VolStr = DelStr(VolStr,1,7)
    NumOfVol = Length(VolStr) % 7

    Do Cnt = 1 to NumOfVol
      Volume = SubStr(VolStr,1,6)
      VolStr = DelStr(VolStr,1,7)
      OutCnt = OutCnt + 1
      OutLine.OutCnt = "                 " || Volume || ","
    End
    Str = OutLine.OutCnt
    OutLine.OutCnt = DelStr(Str,Length(Str),1) || ") ;"
    OutCnt = OutCnt + 1
    OutLine.OutCnt = ' '
    OutCnt = OutCnt + 1
    OutLine.OutCnt = "  COMMIT ;"
End                                        /* Continue with the next NPI */

"EXECIO * DISKW SQLOUT (FINIS STEM OutLine." /* Write DDL             */
Exit(ØØ)                                   /* EOF program execution      */
```

JCL FOR NPISTART

```
//PBSANP1R JOB ,PBSANP1R,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//BSANPIØ1 EXEC PGM=IKJEFTØ1,PARM='%NPISTART'
//SYSEXEC  DD DISP=SHR,DSN=SØØØ.COMM.REXX
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*
//*************************************************************
//* Prepared ALTER STOGROUP Statements will be written to DD **
//* called SQLOUT                                             **
//*************************************************************
//SQLOUT   DD DISP=(NEW,PASS),DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB),
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//*************************************************************
//* First 4 qualifiers of the NPI cluster                      **
//*************************************************************
//INDEX    DD *
PBSA.DSNDBC.DPBSAØ1.XBSA1KVK
/*
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
//ALTSTGRP EXEC DYNSQL,COND=(Ø,NE)
//SYSIN    DD DISP=(OLD,DELETE),DSN=*.BSANPIØ1.SQLOUT
/*
//RECINDEX EXEC DSNUPROC,SYSTEM=DB1P,UID='PBSANP1R',COND=(Ø,NE)
//SYSPRINT DD SYSOUT=*
//DSNUPROC.SYSIN    DD  *
  RECOVER INDEX (PBSA.XBSA_THAREKETØ3)
/*
```

NPIMOVE

The program NPIMOVE reads the DD called INDEX. At this DD there are NPI cluster names (four qualifiers; the format is high_level_qualifier.DSNDBC.database_name.indexspace_name; at each row there is only one name). Then it finds the storage group and the tablespace name of the NPI and creates QUIESCE and STOP commands. It writes QUIESCE commands to QUIESCE DD, and STOP commands to STOP DD. Then it finds the volumes of the storage group and physical volumes of the NPI. It compares these volumes and find PIECEs that should be moved. It writes the necessary MOVE commands to DMS DD. At our site we use DMS, from Sterling Software (now CA), for data management. Then the program creates the necessary DDL to remove physically-used volumes from the storage group. It writes these statements to SQL DD. Finally it sends START control statements to the START DD.

```
/***REXX*************************************************************/
/*                                                                   */
/*  NPIMOVE : Used for distribution of non-partitioned indexes with  */
/*  --------- PIECESIZE onto different volumes.                       */
/*                                                                   */
/* Description:    - Start execution                                 */
/* ============    - Reads DD called INDEX to get the names of NPIs. */
/*                   Format of the input is cluster name of the NPI  */
/*                   till INDEXSPACE name.                           */
/*                 - Get the Storage Group name and TS name of the   */
/*                   NPI (Step 1).                                   */
/*                 - Create QUIESCE and STOP commands.               */
/*                 - Get the volumes of the storage group.           */
/*                 - Get the physical volume names of the NPI.       */
/*                 - Find used volumes of the storage group.         */
/*                 - Create MOVE control statements.                 */
/*                 - Remove physically used volumes from STOGROUP.   */
/*                 - Go to Step 1 (Continue with the next NPI).      */
/*                                                                   */
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
/* Return Codes:                                                     */
/* =============                                                     */
/* RC  Description                                                   */
/* --  ------------------------------------------------------------- */
/* ØØ  Normal completion                                             */
/* Ø8  No STOGROUP name found for NPI.                               */
/* 1Ø  There is not enough volume for Stogroup.                      */
/* 12  No NPI name supplied.                                         */
/*********************************************************************/
ADDRESS TSO                  /* PGM will work in TSO environment     */
"PROFILE NOPREFIX"           /* without TSO prefix.                  */

"EXECIO * DISKR INDEX (FINIS STEM IndexLine." /* Get the names of NPI*/
If RC ¬= Ø | IndexLine.Ø = Ø Then
  Do
    Say "ERROR 1: At least one NPI should be supplied"
    Exit(12)                          /* Leave the program RC = 12 */
  End

STOSTACnt = Ø                /* STOP-START command line counter      */

QUICnt = Ø                   /* QUIESCE line counter                 */
QUIStr = ''                  /* DB2 Object names for QUIESCE         */

DMSCnt = Ø                   /* DMS line counter                     */

SQLCnt = Ø                   /* SQL line counter                     */


Do LineCnt = 1 to IndexLine.Ø /* Continue for all NPIs               */

  Obje = Strip(SubStr(Strip(IndexLine.LineCnt),1,44)) /* Index name  */
  ObjeII = Translate(Obje,' ','.')

  SQLQUERY = ,
         "SELECT A.STORNAME, C.DBNAME, C.TSNAME          " ||,
         "  FROM SYSIBM.SYSINDEXPART A,                  " ||,
         "       SYSIBM.SYSINDEXES   B,                  " ||,
         "       SYSIBM.SYSTABLES    C                   " ||,
         "WHERE B.NAME = A.IXNAME AND B.CREATOR = A.IXCREATOR AND" ||,
         "    B.DBNAME = '" || Word(ObjeII,3) || "' AND " ||,
         "    B.INDEXSPACE = '" || Word(ObjeII,4) || "' AND " ||,
         "    C.NAME = B.TBNAME AND C.CREATOR = B.TBCREATOR "
  ADDRESS LINK "REXXSQL"
  If RC = Ø & SQLSTATE = Ø & _NROWS > Ø Then
    Do
      Stogroup = Strip(STORNAME.1)    /* STOGROUP of INDEXSPACE     */
      Call BeforeProcess              /* Create QUIESCE and STOP    */
    End
  Else
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
    Do                                  /* Couldn't get STOGROUP       */
      Say "ERROR 2: Wrong SQL = " || SQLQUERY
      Say "         SQLSTATE = " || SQLSTATE
      Exit(Ø8)
    End

  Call GetStgVolumes              /* Get the volumes of STOGROUP      */

  Call GetDSetVolumes             /* Get physical volumes of INDEXSPAC*/

  Call FindUsedVolumes            /* Find Used volumes of STOGROUP    */

  Call MoveProcess                /* Create MOVE process ( DMS )      */

  Call RemoveVolumes              /* Remove physically used volumes   */

End                       /* The same process for next NPI            */

If DMSCnt = Ø Then        /* Is there any dataset to move             */
  Do                      /* There is no dataset to move, leave RC=Ø4 */
    Say "WARNING:      "
    Exit(Ø4)
  End

"EXECIO * DISKW DMS  (FINIS STEM DMSLine." /* Write down DMS stmnts. */

"EXECIO * DISKW SQL  (FINIS STEM SQLLine." /* Write down SQL stmnts. */

"EXECIO * DISKW STOP (FINIS STEM STOSTALine." /* Write down STOP cmds*/

"EXECIO * DISKW QUIESCE (FINIS STEM QUILine." /* QUIESCE statements  */

Do Cnt = 1 to STOSTACnt
  If Pos(" -STOP ",STOSTALine.Cnt) ¬= Ø Then
    STOSTALine.Cnt = " -START " || DelStr(STOSTALine.Cnt,1,7)
End
"EXECIO * DISKW START (FINIS STEM STOSTALine." /* START commands     */

Exit(ØØ)                  /* Leave the program                        */

/*****************-------------------------------------*****************/
/* BeforeProcess: Create QUIESCE and STOP statements for the objects */
/*****************-------------------------------------*****************/
BeforeProcess: Procedure,
  EXPOSE STOSTACnt STOSTALine. ObjeII DBNAME. TSNAME.,
    QUIStr QUICnt QUILine.

STOSTACnt = STOSTACnt + 1             /* STOP line counter           */
STOSTALine.STOSTACnt = " DSN SYSTEM(DB1P)"
STOSTACnt = STOSTACnt + 1
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
STOSTALine.STOSTACnt = " -STOP DATABASE(" ||,
  Word(ObjeII,3) || ") SPACENAM(" || Word(ObjeII,4) ||,
  ")"                                /* STOP command for NPI        */

TSSpec = Strip(DBNAME.1) || '.' || Strip(TSNAME.1)
If Pos(TSSpec,QUIStr) = Ø Then
  Do
    QUIStr = QUIStr || Left(TSSpec,17)
    QUICnt = QUICnt + 1              /* QUIESCE line counter        */
    QUILine.QUICnt = "  QUIESCE TABLESPACE " || TSSpec ||,
      " WRITE YES"                   /* QUIESCE command for TS      */

    STOSTACnt = STOSTACnt + 1
    STOSTALine.STOSTACnt = " -STOP DATABASE(" ||,
      Strip(DBNAME.1) || ") SPACENAM(" || Strip(TSNAME.1) ||,
      ")"                            /* STOP command for TS         */
  End
Return                               /* Go back to caller           */

/*****************-------------------------------------*****************/
/* GetStgVolumes: Get volume names of the storage group. DB2EX1 is   */
/*                the default volume name for all storage group.     */
/*****************-------------------------------------*****************/
GetStgVolumes: Procedure,
  EXPOSE StoGroup StoVol. VolUsed.

VolCnt = Ø                           /* Volume counter for STOGROUP */

SQLQUERY = "SELECT VOLID " ||,
           "  FROM SYSIBM.SYSVOLUMES " ||,
           "  WHERE SGNAME = '" || Stogroup || "' AND VOLID ¬= '" ||,
           "DB2EX1'"
ADDRESS LINK "REXXSQL"
If RC = Ø & SQLSTATE = Ø & _NROWS > Ø Then /* Volume names           */
  Do
    Do Cnt = 1 to _NROWS             /* Continue for all volumes    */
      VolCnt = VolCnt + 1
      StoVol.VolCnt = VOLID.Cnt
      VolUsed.VolCnt = Ø             /* Assume that volume not used */
    End
    StoVol.Ø = VolCnt                /* Total number of volumes     */
  End
Return                               /* Call back to caller         */

/*****************-------------------------------------*****************/
/* GetDSetVolumes: Get physical volume names of PIECEs.              */
/*****************-------------------------------------*****************/
GetDSetVolumes: Procedure,
  EXPOSE Obje DSList. DSMove. DSVolume.
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
ReturnCode = OutTrap('Mesaj.')   /* Execute command LISTC LVL       */
"LISTC LVL(" || Obje || ")"
Dummy = OutTrap('OFF')

DSCnt = Ø                        /* PIECE counter for NPI           */

Do I = 1 to Mesaj.Ø              /* Process command output          */
  If SubStr(Mesaj.I,1,7) = "CLUSTER" Then   /* Get the name of clustr*/
    Do
      IndexDSet = Strip(SubStr(Mesaj.I,17,44))   /* Name of Cluster  */

      ReturnCode = OutTrap('ListCOut.')   /* Get physical volume name*/
      "LISTC ENT(" || IndexDSet || ") VOLUME"
      Dummy = OutTrap('OFF')

      Do J = 1 to ListCOut.Ø     /* Process LISTC ENT command output */
        If SubStr(ListCOut.J,8,6) = "VOLSER" Then   /* Get volume    */
          Do
            DsCnt = DSCnt + 1
            DSList.DSCnt = IndexDSet
            DSVolume.DSCnt = SubStr(ListCOut.J,26,6)
            DSMove.DSCnt = Ø     /* Assume that No move process      */
          End
      End
      DSList.Ø = DSCnt
    End
End
Return                           /* Call back to caller             */

/*****************-------------------------------------*****************/
/* FindUsedVolumes: Find volumes that are physically used and exist  */
/*                  in STOGROUP                                      */
/*****************-------------------------------------*****************/
FindUsedVolumes: Procedure,
  EXPOSE DSList. StoVol. DSVolume. VolUsed. DSMove.

Do DSCnt = 1 to DSList.Ø         /* Find used volumes               */
  VolFound = Ø                   /* Assume that volume not found    */
  Do VolCnt = 1 to StoVol.Ø      /* For all stogroup volumes        */
    If DSVolume.DSCnt = StoVol.VolCnt Then /* Volume found ?        */
      Do
        If ¬VolUsed.VolCnt Then
          VolUsed.VolCnt = 1     /* Volume Used by the NPI          */
        Else If VolUsed.VolCnt Then /* Is it Used before ?          */
          DSMove.DsCnt = 1       /* Yes, Dataset should be moved    */
        VolFound = 1             /* Volume found                    */
      End
  End
  If ¬VolFound Then              /* Volume not exist in STOGROUP,   */
    DSMove.DSCnt = 1             /* Dataset should be moved         */
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
End
Return                            /* Go back to caller                */

/*****************-------------------------------------*****************/
/* MoveProcess: Create DMS move statements for datasets that should  */
/*              be moved.                                            */
/*****************-------------------------------------*****************/
MoveProcess: Procedure,
  EXPOSE DSList. DSMove. StoVol. VolUsed. DMSCnt DMSLine. Stogroup
Do DSCnt = 1 to DSList.Ø       /* Continue for all PIECEs            */
  If DSMove.DSCnt Then         /* The dataset should be moved ?      */
    Do
      VolOK = Ø                /* Check for Volume counter           */
      Do VolCnt = 1 to StoVol.Ø /* Continue for all STOGROUP volumes */
        If ¬VolUsed.VolCnt Then /* If the volume not Used ?          */
          Do
            VolUsed.VolCnt = 1  /* Volume is used by the PIECE       */
            DMSCnt = DMSCnt + 1 /* Create DMS control statements     */
            DMSLine.DMSCnt = "  FIND DSN=" || DSList.DSCnt
            DMSCnt = DMSCnt + 1
            DMSLine.DMSCnt = "  MOVE TOVOL=(" || StoVol.VolCnt || ")"
            DMSCnt = DMSCnt + 1
            DMSLine.DMSCnt = " "
            VolOK = 1           /* Number of volumes is enough       */
            VolCnt = StoVol.Ø   /* Do not continue for the Stogroup  */
          End
      End
      If ¬VolOK Then            /* Number of volumes, not enough     */
        Do
          Say "ERROR 3: " || Stogroup || ", not enough volume "
          Say "         Add volumes to STOGROUP, then reexecute"
          Exit(1Ø)
        End
    End
End
Return                            /* Go back to caller                */
/*****************-------------------------------------*****************/
/* RemoveVolumes: Remove volumes from the STOGROUP                   */
/*****************-------------------------------------*****************/
RemoveVolumes: Procedure,
  EXPOSE Stogroup StoVol. VolUsed. SQLCnt SQLLine.

SQLCnt = SQLCnt + 1             /* SQL line counter                  */
SQLLine.SQLCnt = "  ALTER STOGROUP " || Stogroup || ,
  " REMOVE VOLUMES( "
Do VolCnt = 1 to StoVol.Ø       /* Continue for all STOGROUP volumes */
  If VolUsed.VolCnt Then
    Do
      SQLCnt = SQLCnt + 1
      SQLLine.SQLCnt = "        " || StoVol.VolCnt || ','
```

 © 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
    End
End
SQLLine.SQLCnt = DelStr(SQLLine.SQLCnt,Length(SQLLine.SQLCnt),1)||") ; "
SQLCnt = SQLCnt + 1
SQLLine.SQLCnt = " "
Return                          /* Go back to caller                */
```

JCL FOR NPIMOVE

```
//PBSANPMV JOB ,PBSANPMV,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//NPIMOVE  EXEC PGM=IKJEFTØ1,PARM='%NPIMOVE'
//SYSEXEC  DD DISP=SHR,DSN=SØØØ.COMM.REXX
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
//QUIESCE  DD DISP=(NEW,PASS),DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB),
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//STOP     DD DISP=(NEW,PASS),DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB),
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//DMS      DD DISP=(NEW,PASS),DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB),
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//SQL      DD DISP=(NEW,PASS),DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB),
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//START    DD DISP=(NEW,PASS),DCB=(LRECL=8Ø,BLKSIZE=312Ø,RECFM=FB),
//         UNIT=SYSALLDA,SPACE=(CYL,(1,1))
//INDEX    DD *
PBSA.DSNDBC.DPBSAØ1.XBSA1KVK
PBSA.DSNDBC.DPBSAØ1.XBSA17YH
/*
//QUIESCE EXEC DSNUPROC,SYSTEM=DB1P,SIZE=4M,UID='PBSANPMV',COND=(Ø,NE)
//SYSPRINT DD SYSOUT=*
//DSNUPROC.SYSIN DD DISP=(OLD,DELETE),DSN=*.NPIMOVE.QUIESCE
/*
//STOP     EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(Ø,NE)
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DISP=(OLD,DELETE),DSN=*.NPIMOVE.STOP
/*
//DMSMOVE EXEC DMS,T=L,COND=(Ø,NE)
//SYSIN    DD   DISP=(OLD,DELETE),DSN=*.NPIMOVE.DMS
/*
//REMOVE   EXEC DYNSQL,COND=(Ø,NE)
//SYSIN    DD   DISP=(OLD,DELETE),DSN=*.NPIMOVE.SQL
/*
//START    EXEC PGM=IKJEFTØ1,DYNAMNBR=2Ø,COND=(Ø,NE)
//SYSTSPRT DD   SYSOUT=*
//SYSTSIN  DD   DISP=(OLD,DELETE),DSN=*.NPIMOVE.START
/*
```

Ergun Ozel
Systems Programmer
Aknet A S (Turkey)

© Xephon 2000

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

Index advisor model

INTRODUCTION

Choosing index columns for a table requires all programs that use the table to be scanned. This process takes a lot of a DBA's time. This model analyses all static SQL written in a DB2 subsystem, and determines the best indexes that should be defined for the tables. It also lists which SQL statements are most likely to use which indexes, with its matching column and possible number of qualified row values. Since all SQL in the system is parsed, this model can also be used to recommend some SQL enhancements.

PARSING SQL

All static SQL that is written on the subsystem is kept in two tables – SYSIBM.SYSSTMT and SYSIBM.SYSPACKSTMT. One piece of SQL code may be obtained from more than one row in the table, depending on the length of the SQL. Some SQL code consists of two or more SQL statements combined using the UNION keyword. And some SQL may have subselects, using the IN, EXISTS, =, >,<, >=,and <= operators (with or without quantified predicates), or nested table expressions. Subselects are processed as a different SQL, and replaced with the text 'SUBSELECT' in the original SQL. All simple SQL statements are extracted from the main SQL and are loaded into the TFND.SQL_STMTS table. If the SQL joins more than one table, it is inserted for each table that is involved in the join.

For the INDEX adviser model, we do not have to parse the SQL exactly. So, we do not keep records of arithmetic operations, concatenations, and all the functions that are applied to a column or a hostvar. We have to know only that there are such events, which make a column non-indexable, in the predicates. We will keep only the first function that is applied to a column or a host variable.

In the WHERE clause, there may be many parentheses, some of them

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

necessary and some not. In this model, every open bracket creates a new level. And every close bracket make the current level the previous level. Level relations are stored in the table TFND.LEVEL_RL. Within a level, there can be many predicates and they are stored in the table TFND.STATEMENTS. If the parenthesis is unnecessary, it causes unnecessary levels. These unnecessary levels must be removed so that level 0 will always be the Boolean term predicate. The removing process should reorganize the tables TFND.LEVEL_RL, TFND.STATEMENTS and TFND.STATEMENT_RL.

Let us examine the following example:

```
Statements: 1,Ø 2,Ø   2,1           2,2       3,Ø  3,1          3,2
     WHERE  (   (col1 = :H and col2 = :H) and (col3 = :H or col4 = :H))
Levels:    Ø 1             2             1            3           1 Ø
```

This statement can be simplified as follows:

```
Statements:     Ø,1           Ø,2      1,Ø   1,1          1,2
     WHERE  col1 = :H and col2 = :H and (col3 = :H or col4 = :H )
Levels    :          Ø                          1
```

Every predicate is specified with a level ID and sequence pair. Predicates can be connected to each other with AND or OR. A predicate can be connected to an entire level, which is shown as a level ID and 0 pair. The TFND.STATEMENT_RL table shows these relationships. In the above example there is an AND relation between 0,1 and 0,2. Likewise, there is an AND relation between 0,2 and 1,0. Finally there is an OR relation between 1,1 and 1,2.

A state diagram for the parsing process is shown in Figure 1. In this process, there are four states – WAIT_FOR_OPERAND1, WAIT_FOR_OPERATOR, WAIT_FOR_OPERAND2, and WAIT_FOR_LOGICAL_OP. The process starts from the WAIT_FOR_OPERAND1 state when a ‘WHERE’ or ‘ON’ keyword is encountered. The WHERE condition of the SQL is read word by word and, according to the word read and current state, the new state is determined. The process continues until one of ‘WITH’, ‘OPTIMIZE’, ‘GROUP’, ‘ORDER’, or ‘FOR’ is read or the last word of the line is encountered. Further explanation of the state diagram is given below:

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

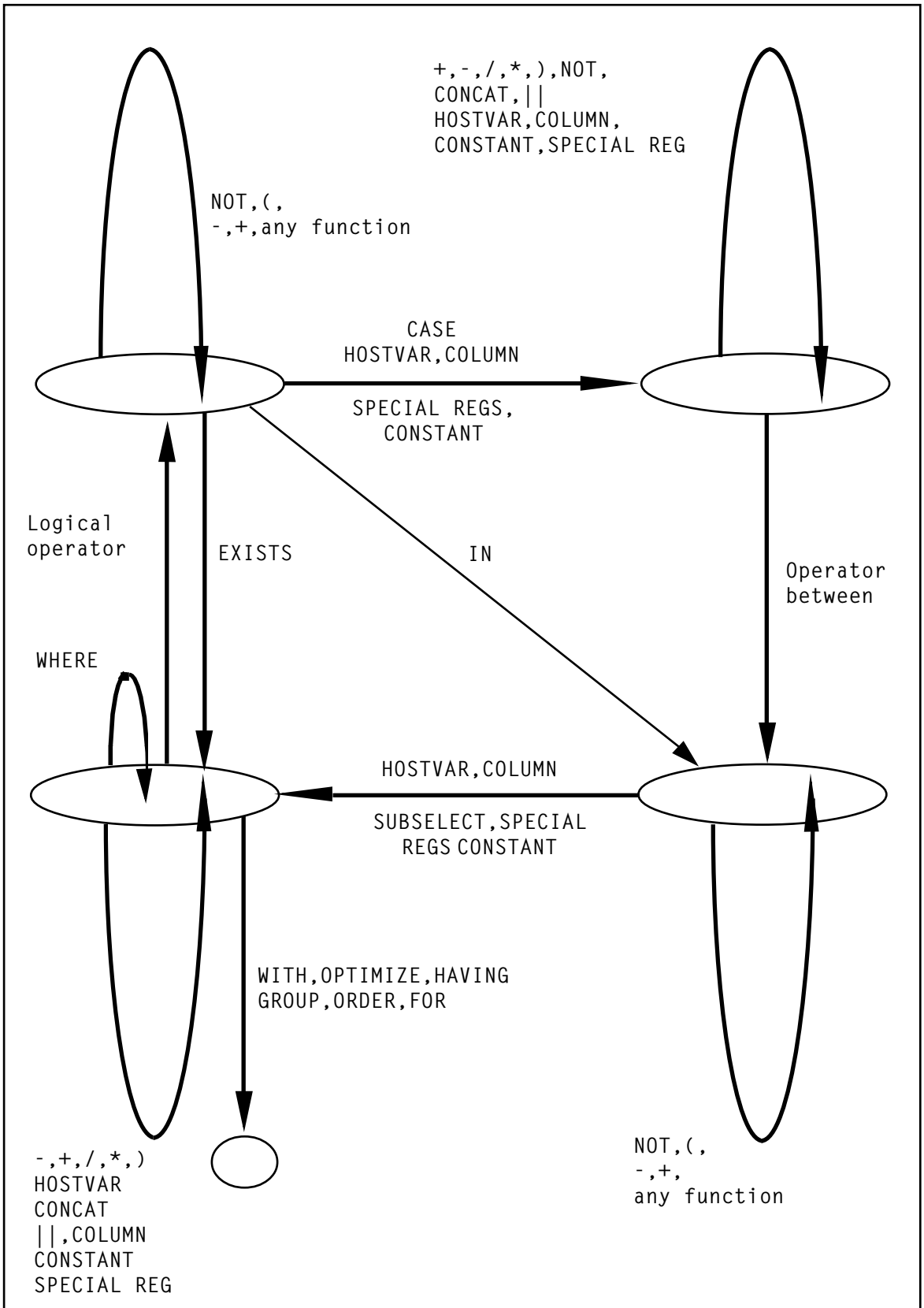

Figure 1: State diagram of parsing process

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

WAIT_FOR_OPERAND1 and WAIT_FOR_OPERAND2

If there is a function in front of the operand, the next word will be '('. Move the pointer one word ahead and stay in the current state. Only the first function of the operand will be kept in table TFND.STATEMENTS.

If any function is applied to the operand, after reading the operand, there will be ')' words. Move the pointer until the number of functions of ')' is read.

If the function is 'SUBSTR', read the second parameter of the function.

If 'NOT' comes, set the not flag of the operand to 1. Stay in the current state.

If '-' comes, stay in the current state and set the status of the operand to 'ARITHMETIC'.

If '+' comes, stay in the current state.

WAIT_FOR_OPERAND1

If EXISTS comes, the next word will be 'SUBSELECT'. Move the pointer one word ahead and go to WAIT_FOR_LOGICAL_OP state.

If '(' comes, move the pointer one word ahead and stay in the current state. Create a new level and set the current level to that.

If 'CURRENT' comes, it is a special register. Move the pointer one word ahead and go to WAIT_FOR_OPERATOR state.

If 'USER' comes, it is a special register. Go to WAIT_FOR_OPERATOR state.

If a host variable, a column, or a constant comes, go to WAIT_FOR_OPERATOR state.

If 'CASE' comes, move the pointer until the 'END' word comes. Set the status of the operand to 'CASE' and go to WAIT_FOR_OPERATOR state.

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

WAIT_FOR_OPERAND2

If '(' comes, it is an unnecessary parenthesis. Stay in the current state.

If 'CURRENT' comes, it is a special register. Move the pointer one word ahead and go to WAIT_FOR_LOGICAL_OP state.

If 'USER' comes, it is a special register. Go to WAIT_FOR_LOGICAL_OP state.

If a host variable, a column, 'SUBSELECT', or a constant comes, go to WAIT_FOR_LOGICAL_OP state.

WAIT_FOR_OPERATOR and WAIT_FOR_LOGICAL_OP

If any of '*', '/', '+', '-', 'CONCAT', '||' comes, stay in the current state. If the word is 'CONCAT' or '||', the state of the operand will be 'CONCAT', otherwise it is 'ARITHMETIC'.

Host variable, column name, constant or special register can only come after one of the operators '-', '+', '/', '*', '||', 'CONCAT' comes. Stay in the current state.

If one of 'YEARS', 'DAYS', 'MONTHS', 'YEAR','DAY', 'MONTH','HOURS','HOUR' comes, this word comes after an arithmetic operation of the operand. For example col1 + 2 HOURS. Stay in the current state.

WAIT_FOR_OPERATOR

Any operator may be '=', '<>', '>', '<', '>=', '<=', '¬=', '¬>', '¬<', 'LIKE' or 'IS'. If one of them comes, go to WAIT_FOR_OPERAND2 state.

If 'BETWEEN' comes, read the first parameter (it may be a hostvar, a column, or function of them). Move the pointer after the word 'AND' and go to 'WAIT_FOR_OPERAND2' state.

If 'IN' comes, read the next word. It can be 'SUBSELECT' or a set of values. Count the number of values and move the pointer after the word ')' or 'SUBSELECT'. Go to 'WAIT_FOR_LOGICAL_OP' state.

 © 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

If ‘)’ comes, an unnecessary open parenthesis exists in front of the operand1. Remove this level and stay in the current state.

If ‘NOT’ comes, set the not flag of the operator to 1. Stay in the current state.

WAIT_FOR_LOGICAL_OP

If ‘)’ comes, stay in the current state. The current level is set to the previous level.

If a logical operator (‘AND’ or ‘OR’) comes, insert the statement into the table TFND.STATEMENTS. Go to ‘WAIT_FOR_OPERAND1’ state.

If one of ‘WITH’, ‘OPTIMIZE’, ‘GROUP’, ‘ORDER’, or ‘FOR’ comes, finish the process.

If ‘WHERE’ comes, it is an SQL with a join. Suppose that an open parenthesis exists, create a new level and set a flag so that it would make close parenthesis processes at the end of the SQL.

The table definitions used in the parsing process are described below.

TABLE TFND.SQL_STMTS

All SQL written in the subsystem is inserted into the TFND.SQL_STMTS table:

```
PROGNAME          CHAR(8),        PK
TABLENAME         VARCHAR(18),    PK
STMTNO            SMALLINT,       PK
STMTSEQ           SMALLINT,       PK
  # For subselects and unions, this column can be more than 0.
TABLEALIAS        VARCHAR(18),
IS_STMT_SUBS      CHAR(1),
IS_STMT_UNION     CHAR(1),
IS_STMT_JOIN      CHAR(1),
STMT              VARCHAR(10000)
```

TABLE TFND.LEVEL_RL

Level relationships are kept in the TFND.LEVEL_RL table:

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
LEVEL_ID        SMALLINT,    PK
PREV_LEVEL_ID   SMALLINT,
LEVEL_NOT_FLAG  CHAR(1)
```

TABLE TFND.STATEMENTS

Every predicate has a row in the TFND.STATEMENTS table:

```
STMT_LEVEL_ID        SMALLINT,     PK
STMT_SEQNO           SMALLINT,     PK
STMT_ROW_SEQ         SMALLINT,
STMT_OP_NOT_FLG      CHAR(1),
STMT_OP1_TBNAM       VARCHAR(18),
 #  If operand is a column name, the table name of the column.
STMT_OP1             VARCHAR(2Ø),
 #  First operand.
STMT_OP1_FUN         VARCHAR(15),
 #  First function of the operand
STMT_OP1_PARM1       SMALLINT,
 #  If the function is SUBSTR, the second parameter of the function
STMT_OP1_ST          VARCHAR(1Ø)
 #  May be 'CONCAT' , 'ARITHMETIC' or 'CASE'
STMT_OP1_IS_COLUMN      CHAR(1),
 #  1 if the operand1 is the column of this table.
STMT_OPRTR           VARCHAR(1Ø),
 # may be one of '=', '<>', '>', '<', '>=', '<=', '¬=', '¬>', '¬<',
'LIKE', 'IS' or 'BETWEEN'
STMT_OPRTR_NOT_FLG   CHAR(1),
 #  1 if there is NOT in front of the operator.
STMT_OP2_TBNAM       VARCHAR(18),
STMT_OP2             VARCHAR(2Ø),
STMT_OP2_FUN         VARCHAR(15),
STMT_OP2_PARM1       SMALLINT,
STMT_OP2_ST          VARCHAR(1Ø)
STMT_OP2_IS_COLUMN   CHAR(1),
STMT_OP3_TBNAM       VARCHAR(18),
STMT_OP3             VARCHAR(2Ø),
 #  If operator is BETWEEN, second operand of the BETWEEN.
STMT_OP3_FUN         VARCHAR(15),
STMT_OP3_PARM1       SMALLINT,
STMT_OP3_IS_COLUMN   CHAR(1),
STMT_VAL_CNT         SMALLINT,
 #  If operator is IN, keeps the count of the values.
STMT_FREQ_PRTY       DEC(7)
 #  Frequency priority of the statement.
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

TABLE TFND.STATEMENT_RL

Table TFND.STATEMENT_RLshows the relationships between the predicates:

```
LEVEL_ID1      SMALLINT,     PK
SEQ_NO1        SMALLINT,     PK
LEVEL_ID2      SMALLINT,     PK
SEQ      _NO2  SMALLINT,     PK
LEVEL_ROW_SEQ  SMALLINT,
LOGICAL_OP     CHAR(3)
 #  May be AND or OR.
```

Using this model, let us examine the following SQL:

```
FROM DHST . CST_PSTG_ENT WHERE ISL_BRM_KOD IN ( 1Ø3 , 117 , 85Ø ) AND
KYNK_KOD = ''AAA'' AND BRM_KOD NOT IN ( SELECT BRM_KOD FROM DHST . OU
WHERE NOT BRM_TIP_KOD =  : H  ) AND ( SUBSTR ( TXN_EV_TIP_K OD , 2 , 3 )
= ''KYT'' OR ON_OFF_FLAG = ''1'' OR IPTAL_FLAG = ''H'' OR MHSB_SIRA _NO
= 1 ) AND VLR - 1 DAYS > CURRENT DATE AND SIS_TAR_ZMN BETWEEN : H AND :
H
```

After extracting subselects, the SQL will look like the following:

```
FROM DHST . CST_PSTG_ENT WHERE ISL_BRM_KOD IN ( 1Ø3 , 117 , 85Ø ) AND
KYNK_KOD = 'AAA' AND BRM_KOD NOT IN SUBSELECT  AND ( SUBSTR (
TXN_EV_TIP_KOD , 2 , 3 ) = 'KYT' OR ON_OFF_FLAG = '1' OR IPTAL_FLAG =
'H' OR  MHSB_SIRA_NO = 1 ) AND VLR - 1 DAYS > CURRENT DATE AND
SIS_TAR_ZMN BETWEEN : H AND : H
```

Subselect will be inserted into TFND.SQL_STMTS for table name OU.

After parsing this SQL, the rows shown in Figure 2 are inserted into the tables.

STATEMENT_RL TABLE

The STATEMENT_RL table is shown below:

```
  Ø        1        Ø        2      AND
  Ø        2        Ø        3      AND
  Ø        3        1        Ø      AND
  1        1        1        2      OR
  1        2        1        3      OR
  1        3        1        4      OR
  1        Ø        Ø        4      AND
  Ø        4        Ø        5      AND
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
LEVEL_RL TABLE
       1   Ø   Ø

STATEMENTS TABLE

Ø 1 1 Ø '' ISL_BRM_KOD             Ø  ''          1  IN      Ø '' HOSTVAR      '' Ø  '' Ø '' ''        ''   Ø Ø 3  1
Ø  2 2 Ø '' KYNK_KOD               Ø  ''          1 =        Ø '' 'AAA'        '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
Ø 3 3 Ø '' BRM_KOD                 Ø  ''          1  IN      1 '' SUBSELECT    '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
1 1 4 Ø '' TXN_EV_TIP_KOD SUBSTR   2  ''          1  =       Ø '' 'KYT'        '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
1 2 5 Ø '' ON_OFF_FLAG             Ø  ''          1  =       Ø '' '1'          '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
1 3 6 Ø '' IPTAL_FLAG              Ø  ''          1  =       Ø '' 'H'          '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
1 4 7 Ø '' MHSB_SIRA_NO            Ø  ''          1  =       Ø '' 1            '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
Ø 4 8 Ø '' VLR                     Ø ARITHMETIC   1  >       Ø '' CURRENT DATE '' Ø  '' Ø '' ''        ''   Ø Ø Ø  1
Ø 5 9 Ø '' SIS_TAR_ZMN             Ø  ''          1 BETWEEN Ø '' HOSTVAR      '' Ø  '' Ø '' HOSTVAR   ''   Ø Ø Ø  1
```

Figure 2: Rows to be inserted into the tables

CRITERIA USED FOR CHOOSING INDEX COLUMNS

All SQL statements in the TFND.SQL_STMTS table are read in table name order. So, all SQL statements using a table are parsed, indexable columns are determined, and the best indexes are chosen. Tables used for the index choosing process are listed below. All tables except TFND.FREQ_PRTY are deleted before each table is processed.

TABLE TFND.ST_INDEX_COLS

The TFND.ST_INDEX_COLS table keeps all the indexable predicates of an SQL statement:

```
STIX_PKG_NAME     CHAR(8),
STIX_STMT_NO      SMALLINT,
STIX_STMT_SEQ     SMALLINT,
STIX_STMT_PART    SMALLINT,
 #  Used for multiple index access.
STIX_COL_NAME     VARCHAR(18),
STIX_PRTY         DEC(8,1),
 #  Frequency priority is multiplexed by Ø.1 or 1 depending on the
predicate type.
STIX_PRED_FF      DEC(4,3),
 #  Predicate's filter factor.
STIX_TYPE         CHAR(1)
```

TABLE TFND.INDEXES

The TFND.INDEXES table keeps the suggested indexes for a table:

```
IX_NO          SMALLINT,
IX_COL_NAME    VARCHAR(18),
IX_SEQ         SMALLINT,
IX_PRTY        DEC(15,1),
 #  Calculated priority of this index column.
IX_COUNT       SMALLINT,
 #  How many SQL would probably use this index with a matching column of
IX_SEQ.
IX_UNIQUE      CHAR(1),
IX_COLCARD     INTEGER
```

TABLE TFND.ST_IX_RL

The TFND.ST_IX_RL table shows the relationships between statement and index:

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
STIX_PKG_NAME      CHAR(8),
STIX_STMT_NO       SMALLINT,
STIX_STMT_SEQ      SMALLINT,
IX_NO              SMALLINT,
MATCH_COLS         SMALLINT,
MATCH_COLS_EQUAL   SMALLINT,
 #  Matching column count that all of the columns are used in equal
predicates.
STIX_ROWS_QLFYD    DEC(15,5)
```

TABLE TFND.COLUMNS

The TFND.COLUMNS table keeps the column priorities:

```
COLNAME       VARCHAR(18),
PRTY          DEC(15,2)
 #  Calculated by using COLCARD and CARD.
```

TABLE TFND.FREQ_PRTY

The TFND.FREQ_PRTY table keeps the run-time frequencies of statements:

```
PROG_NAME    VARCHAR(18),
STMT_NO      SMALLINT,
FREQ_PRTY    INTEGER
```

SELECTING COLUMNS

All Boolean term predicates are checked in the WHERE clause. If any of them are connected with an OR, the statement needs multiple index access. All predicates, which are one of the type of equal, in-list, LIKE, BETWEEN, LESS THAN, and GREATER THAN, are selected as indexable predicates. Indexable columns can be selected by running the following SQL:

```
SELECT STMT_OP1,STMT_OP2,STMT_OPRTR,STMT_VAL_CNT
         FROM TFND.STATEMENTS
         WHERE STMT_OP_NOT_FLG = 'Ø' AND
               STMT_LEVEL_ID = Ø AND
               STMT_OP1_FUN = '' AND
               STMT_OP2_FUN = '' AND
               NOT STMT_OP2 = 'SUBSELECT' AND
               STMT_OP1_ST = '' AND
               STMT_OP2_ST = '' AND
               STMT_OPRTR_NOT_FLG = 'Ø' AND
```


© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

```
( STMT_OP1_IS_COLUMN = '1' AND
  STMT_OP2_IS_COLUMN = '0' OR
  STMT_OP1_IS_COLUMN = '0' AND
  STMT_OP2_IS_COLUMN = '1' ) AND
( STMT_OPRTR IN ('=','<','<=','>','>=') OR

  STMT_OPRTR = 'BETWEEN' AND
  STMT_OP2_IS_COLUMN = '0' AND
  STMT_OP3_IS_COLUMN = '0' OR

  STMT_OPRTR = 'LIKE' AND
  SUBSTR(STMT_OP2,1,1) NOT IN ('_','%') OR

  STMT_OPRTR = 'IN' AND
  STMT_OP2 <> 'SUBSELECT' ) ;
```

A priority is defined for frequency. One SQL may be run 1,000 times in a day while another SQL is run once a month. The TFND.FREQ_PRTY table is defined for this priority.

For example, priority 1 is given for SQL that is run once a month, and other SQL will have the priority that is the value of the number of runs in a month. If a priority with a program name and statement number is inserted into this table, this value is taken as its SQL priority. If a statement number is entered as 0, all SQL in this program will have that priority.

In the table TFND.STATEMENTS, a priority for the predicates is kept. This priority depends on the predicate type and frequency priority. If the predicate is an equal predicate, the priority factor is 1, otherwise it is 0.1. This means that each equal predicate is ten times as valuable as the other predicate types. The frequency priority of the SQL is multiplied by this factor and the result is kept in table TFND.STATEMENTS.

Equal predicates are marked as type 1 indexable predicates. Others are type 2 indexable predicates. Type 1 indexable predicates enable the next column of the index to be used as a member of matching columns. For example, if the column is used in a range predicate and that column is used as the first column of a multi-column index, other columns of the index in the statement cannot be used as a member of matching columns.

SQL statements and their index usage are stored in table ST_IX_RL.

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

Each row represents an SQL and an index that the SQL uses. It also contains the possible number of rows qualified and the number of matching columns. There are two types of matching column. The first, MATC_COLS, is the number of index columns that are used by the SQL. The second, MATCH_COLS_EQUAL, is the number of index columns that are used in the equal predicate in the SQL. MATCH_COLS is either equal to or greater by 1 than MATCH_COLS_EQUAL. Each row contains the number of qualified rows. The number of qualified rows is calculated as follows:

```
ROWS_QLFYD = CARD / ( COLCARD1 * COLCARD2 * …… * COLCARDn)
```

If the *n*th column is a type 2 indexable predicate, ROWS_QLFYD is multiplied by a filter factor which is listed in the predicate types shown in Figure 3.

Operator	*Filter factor*
=	1
IN	1/IN-List Count
BETWEEN	0.4
LIKE	if the second character is not _ or % then 1/2 Else 1/30 If the operand is HOSTVAR then 0.4
<,>,<=,>=	0.2

Figure 3: Predicate types

Another priority is given to each column. This priority is calculated as a function of COLCARD/CARD and is between 1 and 2. If COLCARD/CARD equals 0, then the priority will be 1. If the proportion is 1, the priority will be 2. The function is not linear. It is formulated as the upper left quarter of a circle. The formula is as follows:

```
COL_PRTY=SQRT ( 1- ( COLCARD/CARD -1 ) *  ( COLCARD/CARD - 1 ) ) + 1
```

In this formula, COLCARD is the column cardinality and CARD is the table cardinality.

For example, let there be two columns. Let the first column have a distinct value of 1, and let another column's cardinality be equal to the

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

table's cardinality. The second column is twice as valuable as the first one from the point of indexibility.

In TFND.ST_IX_RL table, if IX_NO column is equal to 0, it means no index has yet been found for the statement. The first column of the index is selected by looking at type 1 indexable predicates to which an index has not been assigned. Some SQL which has an index assigned is re-evaluated if the number of qualified rows is not satisfied. If all indexable columns are used in the index and the number of qualified rows is not small enough, those statements are not re-evaluated. The column, which has the maximum value of the summary of the multiplication of column priority and predicate priority, is chosen as the first column of the index.

The first column of the index can be selected by using the following SQL:

```
SELECT STIX_COL_NAME,SUM,CNT FROM
             (SELECT STIX_COL_NAME,
                 SUM(DECIMAL(STIX_PRTY,15,1)*DECIMAL(PRTY,15,1))
                         AS SUM,
                      COUNT(*) AS CNT
             FROM TFND.ST_INDEX_COLS A,TFND.ST_IX_RL B,
             TFND.COLUMNS C
             WHERE (B.IX_NO=Ø                          OR
                    B.IX_NO<>Ø AND
                    B.STIX_ROWS_QLFYD>:MAX_QLFYD_ROWS AND
                    B.MATCH_COLS_EQUAL <>
                          ( SELECT COUNT(DISTINCT(STIX_COL_NAME))
                            FROM TFND.ST_INDEX_COLS
                            WHERE A.STIX_PKG_NAME=STIX_PKG_NAME AND
                                  A.STIX_STMT_NO=STIX_STMT_NO   AND
                                  A.STIX_STMT_SEQ=STIX_STMT_SEQ AND
                                  STIX_TYPE='1' )                OR
                    B.IX_NO<>Ø AND
                    B.STIX_ROWS_QLFYD>:MAX_QLFYD_ROWS AND
                    B.MATCH_COLS_EQUAL = B.MATCH_COLS AND
                    EXISTS ( SELECT 1
                             FROM TFND.ST_INDEX_COLS
                             WHERE A.STIX_PKG_NAME=STIX_PKG_NAME AND
                                   A.STIX_STMT_NO=STIX_STMT_NO AND
                                   A.STIX_STMT_SEQ=STIX_STMT_SEQ AND
                                     STIX_TYPE='2'  AND
                              STIX_COL_NAME NOT IN
                              (SELECT IX_COL_NAME FROM
                               TFND.INDEXES
```

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

```
                        WHERE IX_NO=B.IX_NO) )) AND
                A.STIX_PKG_NAME=B.STIX_PKG_NAME    AND
                A.STIX_STMT_NO =B.STIX_STMT_NO     AND
                A.STIX_STMT_SEQ=B.STIX_STMT_SEQ    AND
                A.STIX_COL_NAME=C.COLNAME          AND
                A.STIX_TYPE='1'
          GROUP BY STIX_COL_NAME) AS QRY1
          ORDER BY 2 DESC;
```

The first row of the result set gives the column. The IX_NO of the related rows of the TFND.ST_IX_RL table must be updated.

If the number of qualified rows is small enough, the index will have only one column. Otherwise other columns of the index will be included. This process is repeated until no indexable columns are found or the number of qualified rows is less than a predetermined value.

Other columns of the index can be both type 1 and type 2 indexable predicates. The SQL for these columns is as follows:

```
SELECT STIX_COL_NAME,SUM,CNT FROM
          (SELECT STIX_COL_NAME,
                           SUM(DECIMAL(STIX_PRTY,15)*DECIMAL(PRTY,15))
                           AS SUM,
                          COUNT(*) AS CNT
          FROM TFND.ST_INDEX_COLS A,TFND.ST_IX_RL B,
                      TFND.COLUMNS C
          WHERE B.IX_NO=:IXNO AND
                        B.MATCH_COLS_EQUAL=:IXSEQ-1 AND
                        A.STIX_COL_NAME NOT IN
                        (SELECT IX_COL_NAME
                          FROM TFND.INDEXES
                         WHERE IX_NO=:IXNO) AND
                                    A.STIX_PKG_NAME=B.STIX_PKG_NAME AND
                                    A.STIX_STMT_NO =B.STIX_STMT_NO  AND
                                    A.STIX_STMT_SEQ=B.STIX_STMT_SEQ AND
                                    A.STIX_COL_NAME=C.COLNAME
          GROUP BY STIX_COL_NAME) AS QRY1
          ORDER BY 2 DESC;
```

The first row of the result set gives the column. The IX_NO of the related rows of the TFND.ST_IX_RL table must be updated.

Finding the first and then other columns of the index is repeated until all statements have an index assigned.

Unique indexes that are already defined on tables must remain with

© 2000. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.

the same columns, since they are used for both indexed access and uniqueness. Therefore, they are kept with the same columns. But the column order can be changed according to the need of indexed access. All of the unique indexes are tried to see if they conform to any of the suggested indexes. The index, which has the most columns of the unique index, is choosen. If none of the suggested indexes has any columns of the unique index, a new index is created.

SQL OPTIMIZATION

Having parsed all the SQLstatements, some suggestions on how they can be optimized can be made. Some examples are listed below:

- If the SUBSTR function is used on a column which is the first of *n* columns, using LIKE will make it possible to use an index on that column.
- If there are any column-to-column operations, the index on one of the columns will not be usable.
- If any conversion function is used against a hostvar, a suggestion would be made to convert the hostvar in the host language.
- If all the predicates in the WHERE clause are the BETWEEN type, or the cardinality of the EQUAL predicate is too small to eliminate most of the rows, the REOPTS(VAR) option of the bind process is recommended.

Abdullah Ongul
DBA
Disbank (Turkey)

© Xephon 2000

As a free service to subscribers and to avoid the need to re-key the scripts, code from individual articles of *DB2 Update* can be accessed on our Web site.

You will need the user-id printed on the envelope containing your *Update* issue and a copy of the printed issue. Once you have registered, any code requested will be e-mailed to you.

© 2000. Reproduction prohibited. Please inform Xephon of any infringement.

IBM has updated Version 6 of its DB2 UDB for OS/390, integrating and packaging the accumulated service packs into Version 6.

DB2 Performance Monitor has been enhanced for usability, provides support for new DB2 function, and has a new API to the On-line Monitor Data Collector, for retrieving performance information about the subsystem being monitored. Users can get raw data and derived performance information including snapshot information and recent history collected to a dataset, including exception alerts based on DB2 events.

Also included is DB2 Forms for OS/390, which is now available as an optional feature of DB2.

For further information contact your local IBM representative.
URL: http://www.software.ibm.com/data.db2.

* * *

NEON Systems has announced an OEM licence agreement allowing Landmark Systems to develop a new Internet-based system to manage IT and business operations on System/390 mainframes using NEON's Shadow Web Server for OS/390.

It will let sites monitor information remotely via the Web and provide integration to DB2, IMS, CICS, or VSAM data and transactions.

For further information contact:
NEON Systems, 14100 Southwest Fwy, Suite 500, Sugar Land, TX 77478, USA.
Tel: (800) 505-6366.
URL: http://www.neonsys.com.

Compuware has new releases of File-AID/Express and File-AID/Express for MVS, two EAI tools for integrating and migrating large-scale production without, it's claimed, writing custom programs.

Complex conversions between multiple data formats will be possible across an enterprise, with support for DB2, VSAM, IMS, sequential, Oracle, SQL Server, DB2 UDB, Sybase, XML, and flat files.

Conversion engines of the Express products will help convert data for one-time migrations and for both temporary and permanent data interfaces between applications in production. Users can execute the engine on OS/390, Unix, or NT servers.

Developers will be able to convert subsets of data required for testing the application or the interface.

Specific features of the two tools include performing extracts, transforms and loads of complex structures, creating and interpreting XML formatted documents, validation and data cleansing capabilities, and a scalable architecture for high-speed and high-volume migrations.

For further information contact:
Compuware, 31440 Northwestern Highway, Farmington Hills, MI 48334-2564, USA.
Tel: (248) 737 7300.
Compuware, 163 Bath Road, Slough, SL1 4AA, UK.
Tel: (01753) 774000.
URL: http://www.compuware.com/products/file.

* * *