Effective DB2 Object Monitoring Using The DB2 Catalog

By Craig S. Mullins

In order to maintain efficient production DB2-based systems, it is necessary to monitor periodically the DB2 objects comprising those systems. Monitoring is an essential component of post-implementation duties because the production environment is dynamic. Fluctuations in business activity, errors in logical and/or physical design or lack of communication can all cause a system to perform inadequately. An effective strategy for monitoring DB2 objects in the production environment will catch and forestall potential problems before they adversely impact performance. The job of monitoring DB2 objects usually is performed by a database administrator, performance analyst or system administrator.

An effective monitoring strategy will include scheduled performance monitor jobs, regular monitor runs from all on-line environments where DB2 transactions execute (that is, CICS, IMS/DC, TSO) and regularly-scheduled reports from the DB2 catalog. This article will detail a strategy for accomplishing the latter objective.

By querying the DB2 catalog on a consistent basis, many possible hazards can be analyzed and acted on in a proactive mode. This article describes QMF queries which access the DB2 catalog. An analysis accompanies each query highlighting some of the potential problems that can be trapped by reviewing the output from each query.

To implement this strategy, certain basic assumptions have been made. It is assumed that all application plans are bound with the EXPLAIN(YES) option and that each application has its own PLAN_TABLE for the storage of the EXPLAIN results. It is also assumed that scheduled production STOSPACE and RUNSTATS jobs are executed and that plans are rebound whenever RUNSTATS has been executed. This is necessary to maintain current statistical information about all DB2 objects for an application. It is also assumed that RUNSTATS has been performed on the DB2 catalog tablespaces. This will allow the optimal performance of these queries.

It will also be useful to have a report of each PLAN_TABLE for each application. This will allow for cross-checking the DB2 catalog against application plan and optimizer path selection information. Run the query in Figure 1 for each PLAN_TABLE in order to produce these reports. Note: It is crucial for the TIMESTAMP column to be in descending order. This will cause the EXPLAIN data in the report to be sorted in order from the most recent to the oldest for each query in the PLAN_TABLE.

This is important if the PLAN_TABLE(s) being used are not purged. The *DB2 Application Programming Guide* contains information about EXPLAIN and the PLAN_TABLE.

The queries and forms presented in this article were developed using QMF and are run weekly using a batch QMF job. This is easier than submitting the queries from QMF weekly or through SPUFI. Simply build batch QMF JCL, incorporate all these queries and forms into a proc and run the proc.

If these basic assumptions and conditions are not met, it is still possible to implement the queries. To accomplish this, changes may need to be made to the SQL or to the process by which they are submitted. In addition, if STOSPACE, RUNSTATS and EXPLAIN data is not current, the queries may not provide accurate information. With these basic caveats in mind, a description of each catalog query follows.

The Object Listing Queries

- Table Listing — Lists table columns
- Index Listing — Lists index columns

In order to perform database and system administration functions for DB2, often it is necessary to identify quickly object dependencies. For example, a DBA is confronted with the need to analyze a poorly-performing query. He has the query and a report of the EXPLAIN for the query. Missing is a listing of available indexes and candidate columns for creating indexes. The object listing queries (see Figures 2 and 3) provide this and more.

By viewing the output from these two queries a hierarchy of DB2 objects can be easily ascertained (indexes within tables within tablespaces within databases). The output from these queries is superb for navigational purposes. It is easy to get lost in a flood of production objects. By periodically running these queries and saving the output, a DBA can have an up-to-date profile of the environment established in each DB2 subsystem that he must monitor.

The WHERE T.DBNAME IN clause in each of these queries is optional. This clause is used to produce reports only for those databases which need to be monitored. It is usually desirable to eliminate the sample database, the DB2 catalog database and any other extraneous databases (that is, DBEDIT, QMF and so on). This elimination is optional as a DBA may wish to monitor everything known to DB2.

Although the primary purpose of these two queries is navigational, they can also aid in problem determination and performance tuning. For example, note the following query:

```
SELECT A.COL1, A.COL2, B.COL3
FROM TABLE1 A, TABLE2 B
WHERE A.COL1 = B.COL4;
```

If this query is not performing properly, the DBA needs to know the column types and lengths for COL1 in TABLE1 and COL4 in TABLE2. The type and length for both columns should be the same. If they are not then DB2 is performing a data conversion to make the comparison. This will negatively impact performance. Should the data type and length be consistent then find what indexes are defined (if any) on these columns and analyze the EXPLAIN output. Other data such as the key ranges and cardinality may also be significant, whether an index is clustered or not (these items will influence the optimizer's choice of access path) and the number of tables in a tablespace (this may cause performance degradation for non-segmented tablespaces). All of this information can be obtained from these reports.

This is only one level of DB2 performance tuning. The next level will delve

F I G U R E 1

PLAN_TABLE Query

Query

```
        SELECT  QUERYNO, QBLOCKNO, APPLNAME, PROGNAME, PLANNO,
                METHOD, CREATOR, TNAME, TABNO, ACCESSTYPE,
                MATCHCOLS, ACCESSNAME, INDEXONLY,
                SORTN_UNIQ, SORTN_JOIN, SORTN_ORDERBY, SORTN_GROUPBY,
                SORTC_UNIQ, SORTC_JOIN, SORTC_ORDERBY, SORTC_GROUPBY,
                TSLOCKMODE, TIMESTAMP, PREFETCH, COLUMN_FN_EVAL, MIXOPSEQ
        FROM    ownerid.PLAN_TABLE
        ORDER BY
                APPLNAME,
                PROGNAME,
                TIMESTAMP DESC,
                QUERYNO,
                PLANNO
```

Form

FORM.COLUMNS userid.PLANTBF

Total Width Of Report Columns: 148

NUM	COLUMN HEADING	USAGE	INDENT	WIDTH	EDIT	SEQ
1	QN		1	8	L	1
2	QBN		1	2	L	2
3	APPLNAME		1	8	C	3
4	PROGNAME	BREAK1	1	8	C	4
5	PN		1	4	L	5
6	METHOD		1	4	L	6
7	CREATOR		1	8	C	7
8	TNAME		1	8	C	8
9	TN		1	2	L	9
10	AT		1	2	C	10
11	MCOL		1	3	L	11
12	XNAME		1	8	C	12
13	IX_ONLY		1	4	C	13
14	S_U		1	1	C	14
15	S_J		1	1	C	15
16	S_6		1	1	C	16
17	S_O		1	1	C	17
18	C_U		1	1	C	18
19	C_J		1	1	C	19
20	C_6		1	1	C	20
21	C_O		1	1	C	21
22	TSL		1	4	C	22
23	TIMESTAMP		1	16	C	23
24	PF		1	2	C	24
25	COL_EVAL		1	4	C	25
26	MULT_INDX		1	4	L	26

F I G U R E 2

Table Listing Query

Query

```
        SELECT  T.DBNAME, T.TSNAME, T.CREATOR, T.NAME, C.COLNO,
                C.NAME, C.COLTYPE, C.LENGTH, C.SCALE, C.NULLS,
                C.DEFAULT, C.COLCARD, C.HIGH2KEY, C.LOW2KEY, C.UPDATES,
                C.FLDPROC
        FROM    SYSIBM.SYSCOLUMNS C,
                SYSIBM.SYSTABLES   T
        WHERE   (T.CREATOR = C.TBCREATOR AND     T.NAME = C.TBNAME)
        AND     T.TYPE =   'T'
        AND     T.DBNAME IN ('dbname1', 'dbname2', 'dbnamex')
        ORDER BY
                T.DBNAME,
                T.TSNAME,
                T.NAME,
                C.COLNO
```

Form

FORM.COLUMNS userid.LISTTBLF

Total Width Of Report Columns: 135

NUM	COLUMN HEADING	USAGE	INDENT	WIDTH	EDIT	SEQ
1	_DATABASE	BREAK1	1	8	C	1
2	TABLE_SPACE	BREAK2	1	8	C	2
3	TABLE_CREATOR	BREAK3	1	8	C	3
4	_TABLE	BREAK3	1	18	C	4
5	COL_NO		1	3	L	5
6	COLUMN_NAME		1	18	C	6
7	COLUMN_TYPE		1	8	C	7
8	COLUMN_LENGTH		1	6	L	8
9	SCALE		1	6	L	9
10	NU_LL		1	2	C	10
11	DF_LT		1	2	C	11
12	COL_CARD		1	8	L	12
13	HIGH2_KEY		1	8	C	13
14	LOW2_KEY		1	8	C	14
15	_UPDT		1	4	C	15
16	FLD_PROC		1	4	C	16

F I G U R E 3

Index Listing Query

Query

```
        SELECT  T.DBNAME, T.TSNAME, T.CREATOR, T.NAME, I.NAME,
                I.UNIQUERULE, I.CLUSTERING, I.CLUSTERED,
                I.FIRSTKEYCARD, I.FULLKEYCARD, I.NLEAF, I.NLEVELS,
                I.ISOBID, K.COLSEQ, K.COLNAME, K.ORDERING
        FROM    SYSIBM.SYSKEYS      K,
                SYSIBM.SYSTABLES    T,
                SYSIBM.SYSINDEXES   I
        WHERE   (I.TBCREATOR = T.CREATOR AND I.TBNAME = T.NAME)
        AND     (K.IXCREATOR = I.CREATOR AND K.IXNAME = I.NAME)
        AND     (T.DBNAME IN 'dbname1', 'dbname2', 'dbnamex')
        ORDER BY
                T.DBNAME,
                T.TSNAME,
                T.NAME,
                I.NAME,
                K.COLSEQ
```

Form

FORM.COLUMNS userid.LISTINDF

Total Width Of Report Columns: 136

NUM	COLUMN HEADING	USAGE	INDENT	WIDTH	EDIT	SEQ
1	_DATABASE	BREAK1	1	8	C	1
2	TABLE_SPACE	BREAK2	1	8	C	2
3	TABLE_CREATOR	BREAK3	1	8	C	3
4	_TABLE	BREAK4	1	18	C	4
5	_INDEX	BREAK5	1	18	C	5
6	U_Q		1	1	C	6
7	C_L		1	1	C	7
8	C_D		1	1	C	8
9	1ST KEY_CARDINAL		1	8	L	9
10	FULL KEY_CARDINAL		1	8	L	10
11	NO OF_LEAFS		1	6	L	11
12	NO OF_LEVELS		1	6	L	12
13	ISOBID		1	6	C	13
14	COL_SEQ		1	4	L	14
15	COLUMN_NAME		1	18	C	15
16	O_R		1	1	C	16

deeper into the physical characteristics of DB2 objects.

The Physical Statistics Queries

- Tablespace Physical Statistics
- Index Space Physical Statistics

Quite often it will be necessary to trace a performance problem within a DB2 query to the physical level. Characteristics at the physical level are determined when DB2 objects are defined. The focus will be on tablespaces and index spaces as these two objects require that a physical data set be created to support them. Many different options need to be chosen when a DB2 is created. If poor choices are made, performance will be unsatisfactorily impacted. The physical statistics queries can be used to monitor and tune these options.

The Tablespace Physical Statistics Query (see Figure 4) provides a listing of tablespaces within the database and lists all the pertinent physical detail associated with each tablespace. The Index Physical Statistics Query (see Figure 5) provides a report of all indexes grouped by owner with the physical criteria supporting each index. These reports are invaluable tools for diagnosing performance problems when they happen and frequently for catching problems before they occur.

Both reports in this section show the CLOSE RULE associated with the tablespace or index space. Always monitor this rule for both. A CLOSE RULE of Y indicates that every time an object is accessed, a VSAM open and close will be performed by the system. The performance of any query which accesses an object defined this way will be impeded. A CLOSE RULE of N will only perform the VSAM open the first time the object is accessed. It will then remain open until DB2 is shut down. This will increase the performance of the query. It does, however, add the overhead associated with keeping a data set open. This overhead should be minimal and is usually preferred to having a slow-running query.

Each tablespace and index must be reviewed based on its desired usage to determine the CLOSE RULE for it. Certain objects which are accessed infrequently or only once per day will not need to remain open. As a basic rule, define tablespaces as CLOSE NO unless a good reason is provided otherwise. Whenever a query is found to be causing performance problems, always examine the CLOSE RULE for each tablespace and index involved in the query.

These reports are also useful in determining frequency of reorganization. By monitoring PCT Dropped, Far Off and Near Off Rows (in both the tablespace and index spaces), Leaf Distance and Cluster Ratio it can be determined whether to increase or decrease the frequency of running a reorganization. See Table 1 for an analysis of the impact of this information on reorganization frequency.

It is also useful to analyze the tablespace and index space usage. The ability to monitor SPACE USED% for a tablespace is of particular importance. Efficient DASD space usage is important to maintain an optimal operating environment for DB2 objects. When the SPACE USED% consistently remains below 75 percent for an extended period of time, the PRIQTY space for the tablespace should be decreased and the tablespace should be reorganized. If this percentage

TABLE 1

Reorganization Indicators

Column	Object	Impact
PCT Dropped	TS	+ + + + +
Near Off Rows	TS	+
Far Off Rows	TS	+ + + +
Cluster Ratio	Index	– – – – –
Near Off Rows	Index	+
Far Off Rows	Index	+ + + +
Leaf Distance	Index	+ + +

How To Read The Above Chart

A ' + ' indicates REORG more frequently when this number is large
A ' – ' indicates REORG more frequently when this number is small
The greater the number of occurrences the more urgent the need to REORG

FIGURE 4

Tablespace Physical Statistics Query

```
Query

     SELECT    T.DBNAME, T.NAME, T.IMPLICIT, T.LOCKRULE,
               T.ERASERULE, T.CLOSERULE, T.PARTITIONS, T.SEGSIZE,
               T.NTABLES, T.NACTIVE, P.CARD, P.FARINDREF,
               P.NEARINDREF, P.PERCACTIVE, P.PERCDROP,
               P.FREEPAGE, P.PCTFREE, P.STORNAME, P.VCATNAME,
               T.NACTIVE*100*T.PGSIZE/T.SPACE
     FROM      SYSIBM.SYSTABLESPACE   T,
               SYSIBM.SYSTABLEPART    P
     WHERE     T.NAME = P.TSNAME
     AND       T.DBNAME = P.DBNAME
     ORDER BY
               T.DBNAME,
               T.NAME

Form

FORM.COLUMNS                                userid.PHYTABLF

Total Width Of Report Columns: 150

NUM  COLUMN HEADING                         USAGE   INDENT  WIDTH  EDIT  SEQ

 1   _DATABASE                              BREAK1  1       8      C     1
 2   TABLE_SPACE                            BREAK2  1       8      C     2
 3   IMPL                                           1       4      C     3
 4   LOCK_RULE                                      1       4      C     4
 5   E_R                                            1       1      C     5
 6   C_R                                            1       1      C     6
 7   PARTS                                          1       5      L     7
 8   SEG_SIZE                                       1       4      L     8
 9   NO OF_TABLES                                   1       6      L     9
10   NO OF_PAGES                                    1       8      L     10
11   NO OF_ROWS                                     1       11     L     11
12   FAR OFF_ROWS                                   1       11     L     12
13   NEAR OFF_ROWS                                  1       11     L     13
14     PCT _ACTIVE                                  1       7      L     14
15     PCT _DROPPED                                 1       7      L     15
16   FREE_PAGE                                      1       4      L     16
17   PCT_FREE                                       1       4      L     17
18   STOGROUP                                       1       8      C     18
19   VCAT_NAME                                      1       5      C     19
20   SPACE_USED%                                    1       5      L     20
```

FIGURE 5

Index Physical Statistics Query

```
Query

     SELECT    I.CREATOR, I.CREATEDBY, I.NAME, I.UNIQUERULE,
               I.CLUSTERING, I.CLUSTERED, I.CLUSTERRATIO,
               I.FIRSTKEYCARD, I.FULLKEYCARD, I.NLEAF, I.NLEVELS,
               4096/I.PGSIZE, I.ERASERULE, I.CLOSERULE, P.CARD,
               P.FAROFFPOS, P.LEAFDIST, P.NEAROFFPOS, P.FREEPAGE,
               P.PCTFREE
     FROM      SYSIBM.SYSINDEXES     I,
               SYSIBM.SYSINDEXPART   P
     WHERE     I.NAME = P.IXNAME
     AND       I.CREATOR = P.IXCREATOR
     ORDER BY
               I.CREATOR,
               I.NAME

Form

FORM.COLUMNS                                userid.PHYINDXF

Total Width Of Report Columns: 143

NUM  COLUMN HEADING                         USAGE   INDENT  WIDTH  EDIT  SEQ

 1   INDEX_OWNER                            BREAK1  1       8      C     1
 2   _INDEX                                         1       18     C     2
 3   U_Q                                            1       1      C     3
 4   C_L                                            1       1      C     4
 5   C_D                                            1       1      C     5
 6   CLSTR_RATIO                                    1       5      L     6
 7   FIRST KEY_CARDINALITY                          1       11     L     7
 8   FULL KEY_CARDINALITY                           1       11     L     8
 9   NO OF_LEAFS                                    1       6      L     9
10   NO OF_LEVELS                                   1       6      L     10
11   SUB_PAGES                                      1       5      L     11
12   E_R                                            1       1      C     12
13   C_R                                            1       1      C     13
14   NO OF ROWS_REF (CARD)                          1       11     L     14
15   ROWS AT_FAROFF POS                             1       11     L     15
16   LEAF_DISTANCE                                  1       8      L     16
17   ROWS AT_NEAROFF POS                            1       11     L     17
18   FREE_PAGE                                      1       4      L     18
19   PCT_FREE                                       1       4      L     19
```

FIGURE 6

Tablespace Scan Query

```
Query

      SELECT   E.APPLNAME, E.PROGNAME, E.QUERYNO, E.TNAME, T.NPAGES,
               E.TIMESTAMP, S.SEQNO, S.TEXT
      FROM     ownerid.PLAN_TABLE   E,
               SYSIBM.SYSTABLES     T,
               SYSIBM.SYSSTMT       S
      WHERE    ACCESSTYPE = 'R'
      AND      (T.NPAGES > 50 OR T.NPAGES < 0)
      AND      T.NAME = E.TNAME
      AND      T.CREATOR = E.CREATOR
      AND      S.NAME = E.PROGNAME
      AND      S.PLNAME = E.APPLNAME
      AND      S.STMTNO = E.QUERYNO
      ORDER BY
               E.APPLNAME,
               E.PROGNAME,
               E.TIMESTAMP DESC,
               E.QUERYNO,
               S.SEQNO

Form

FORM.COLUMNS                                 userid.SCANTSF

Total Width Of Report Columns: 329

NUM  COLUMN HEADING                          USAGE   INDENT  WIDTH  EDIT  SEQ

 1   PLAN_NAME                               BREAK1  1       8      C     1
 2   DBRM_NAME                               BREAK2  1       8      C     2
 3   QRY_NO                                  BREAK3  1       7      L     3
 4   TABLE_NAME                              BREAK3  1       18     C     4
 5   NO OF_PAGES                             BREAK3  1       7      L     5
 6   _TIME STAMP                             BREAK3  1       16     C     6
 7   SEQ_NO                                          1       3      L     7
 8   SQL_CODE                                        1       254    C     8
```

FIGURE 7

Index Space Scan Query

```
Query

      SELECT   E.APPLNAME, E.PROGNAME, E.QUERYNO, I.NAME, I.NLEAF,
               I.COLCOUNT, E.MATCHCOLS, E.INDEXONLY, E.TIMESTAMP,
               S.SEQNO, S.TEXT,
      FROM     ownerid.PLAN_TABLE   E,
               SYSIBM.SYSINDEXES    I,
               SYSIBM.SYSSTMT       S
      WHERE    E.ACCESSTYPE = 'I'
      AND      N.LEAF > 100
      AND      E.MATCHCOLS < I.COLCOUNT
      AND      I.NAME = E.ACCESSNAME
      AND      I.CREATOR = E.ACCESSCREATOR
      AND      S.NAME = E.PROGNAME
      AND      S.PLNAME = E.APPLNAME
      AND      S.STMTNO = E.QUERYNO
      ORDER BY
               E.APPLNAME,
               E.PROGNAME,
               E.TIMESTAMP DESC,
               E.QUERYNO,
               S.SEQNO

Form

FORM.COLUMNS                                 userid.SCANIXF

Total Width Of Report Columns: 351

NUM  COLUMN HEADING                          USAGE   INDENT  WIDTH  EDIT  SEQ

 1   PLAN_NAME                               BREAK1  1       8      C     1
 2   DBRM_NAME                               BREAK2  1       8      C     2
 3   QRY_NO                                  BREAK3  1       7      L     3
 4   _INDEX                                  BREAK3  1       18     C     4
 5   NO OF_LEAF PAGES                        BREAK3  1       8      L     5
 6   COLUMNS_IN KEY                          BREAK3  1       6      L     6
 7   MATCHING_COLUMNS                        BREAK3  1       8      L     7
 8   INDX_ONLY                               BREAK3  1       4      C     8
 9   _TIME STAMP                             BREAK3  1       16     C     9
10   SEQ_NO                                          1       3      L     10
11   SQL_CODE                                        1       254    C     11
```

is 100 percent and growth is expected, the PRIQTY space should be increased.

Other space considerations can be analyzed via monitoring PCT ACTIVE, FREE PAGE and PCT FREE. In conjunction with a data set allocation report, space can be reviewed and modified as necessary. As a general rule when PCT ACTIVE is low it may be necessary to redefine with a smaller PRIQTY (and/or SECQTY). Free space can also be changed as well. In any event it will be necessary to monitor these reports with the actual data set statistics. Also, remember that in order for changes-to-space characteristics to take affect, the tablespace being altered must be reorganized.

After this level of performance analysis has been exhausted it becomes necessary to broaden the scope of the tuning effort. This will involve analysis of application programs (plans) and may necessitate building new indexes or changing SQL in application queries.

The Scan Queries

- Tablespace Scans Greater Than 50 Pages
- Index Space Scans Greater Than 100 Pages

The scan queries (see Figures 6 and 7) produce reports that can detect many potential performance problems. By combining the DB2 catalog information with the output from EXPLAIN, a series of potential "problem queries" can be identified. These "problem queries" will be grouped into two categories; tablespace scans and index space scans. In each instance, DB2 will scan data sets to satisfy the query. A tablespace scan will not use an index and will read every page in the tablespace. An index space scan will not necessarily read every index subpage but has that potential.

These queries will probably be long-running. Do not execute them in parallel with heavy production DB2 processing or during the on-line DB2 transaction window. For the scan queries to operate efficiently, ensure that the PLAN_TABLE being used in each query does not contain excessive extraneous data. Strive to maintain only the most recent EXPLAIN data from production BIND jobs in the table. Also, only keep EXPLAIN information for plans which need to be monitored. Executing RUNSTATS on the PLAN_TABLES can also positively affect the performance of these queries.

The tablespace scan report will list queries scanning more than 50 pages and queries accessing tables without current RUNSTATS information. If the NO OF PAGES is −1 for any table then RUNSTATS has not been run. A RUNSTATS job should be executed as soon as possible followed by a rebind of any plan which uses this table. Everything else on this report should be monitored closely. For tables over the 50-page threshold, the performance impact is uncertain. The greater the number of pages scanned, the greater the potential for performance problems.

The 50-page cutoff for inclusion on this report is somewhat arbitrary and may need to be redefined as the information returned is gauged. To monitor large tables only, the number may be increased. If a bufferpool is small (under 1000 buffers), the number may be reduced. Some potential numbers and their significance follow:

100	Increase this number to 100 (or larger) to monitor only those queries accessing large tables. This number will vary according to the definition of "large table."
20	For tables with 20 or more pages it is recommended that indexes be created in order to satisfy the predicates in the query. It is not always possible to create an index for every predicate however, so this is only a guideline. Various DB2 references recommend indexes be considered when the number of pages in a tablespace reaches five, six and/or 15. In practice though, 20 pages seems to be a good number.

The index space scan report will list all queries scanning more than 100 leaf pages where a match on the columns in the query is not a complete match on all index columns. As the number of MATCHING COLUMNS increases the possibility of performance problems decrease. The

See DB2 Catalog page 106

simple COBOL programs for the product to pay for itself," he adds.

For NIPSCO the initial goal in testing Compile/QMF was to use it as a tool for migrating QMF queries to production and to save CPU resources. "But," notes Adams, "after finding out how easy it is to use, we are going to give it to our developers for prototyping applications."

For the technical staff, the benefit is increased programmer productivity. For the end user, the benefit is increased responsiveness. "We can be more responsive to user requests," says Adams, "and we can free up our programmers to do more complex tasks."

Compile/QMF provides an easy way around one of the big drawbacks to QMF: its limitations for certain types of reports, including any reports with IF THEN logic. True, Compile/QMF will only compile the existing logic into the COBOL program. But once the COBOL code is generated, it is relatively simple to modify, adding the required additional logic. (Compile/QMF generates programs that are structured and well-commented with meaningful paragraph names.)

Reports with IF logic were the driving force for trying Compile/QMF for John Mackintosh, manager of applications development for Black Box Corporation (Pittsburgh, PA). Black Box, a manufacturer and distributor of data communications equipment, recently upgraded its order entry system. One result was a backlog of reports that had to be converted to the new system. While many of the reports have been generated in QMF, many more are too cumbersome, needing multiple selects, builds and joins in QMF to replace a single IF statement.

Compile/QMF should provide Black Box with a cost-effective solution. "We can still do the QMF query, use Compile/QMF to generate the COBOL code and add the IF statement," says Mackintosh. That was the initial reason for installing Compile/QMF. Now that it is in-house, Mackintosh sees potential incremental value for it in conjunction with an automated scheduling tool they are also installing. "The automated scheduler needs return codes from SQL errors. By converting our QMF queries to COBOL we could generate the true error codes we need."

Like QMF, however, Compile/QMF does not do it all. Compile/QMF currently cannot handle save data. Another problem for some users is the reliance on external routines for certain formatting functions. This is because Compile/QMF was designed to accommodate those features of QMF that cannot be handled with COBOL, specifically word wrapping. By using external routines, the end user does not lose those features when the reports are converted to COBOL. (SableSoft reports that the next release, currently being field tested, does handle save data and offers the option of eliminating the external routines.)

Technical Support

Technical support, especially for new products, is always a paramount concern. SableSoft offers 24-hour technical support for Compile/QMF. Both Donahue and Adams asked for technical support and got what they needed expeditiously. Donahue reported that he found four minor problems in initial testing and they were all corrected in about a day. (The problems he found involved rounding, numbers of places to the right of the comma, a heading value not appearing on the first page of the report and failure to block the work file.)

Adams, too, was pleased with the responsiveness when he asked for help, especially considering that he was asking for help doing something the product was not designed to do. He explains, "The product is designed to be run from a TSO terminal, using TSO Attach. I wanted to run in the production control environment using Call Attach so I called SableSoft and they gave me the JCL for relinking the modules."

Regarding evaluation criteria for the three users surveyed, all of them claimed productivity and performance benefits are more important than cost, customer support or the company's background. Donahue said the cost was attractive and he expects the payback period to be short; however, he was more concerned about getting COBOL code that is maintainable. For Adams, the issue was performance and for Mackintosh, the key issue was ease of use. At $13,000 to $16,900 for the first license, it should not take long to cost-justify Compile/QMF.

Compile/QMF operates with any IBM 370 or plug-compatible system. It requires MVS or VM, DB2 or SQL/DS, and a COBOL compiler. For more information, contact SableSoft, Inc., 2695 Winding Trail Drive, Boulder, CO 80304, (303)443-7791. ⊜

ABOUT THE AUTHOR

Renee Peterson reports on mainframe software developments and is a frequent contributor to MAINFRAME JOURNAL.

DB2 Catalog *from page 18*

worst possible case is to have zero MATCHING COLUMNS, but even this may be acceptable for an index-only scan.

The usage of 100 pages as the cutoff value for the index space scan report may need to be modified. Some suggestions would be to use the same number as chosen for the tablespace scan report or to choose a value that is appropriate for the environment.

The pertinent point to be made is to carefully analyze the results of these queries. The SQL for each query is listed in the report. Every query listed on the report is not necessarily a problem query. Each query, however, should be closely monitored. Corrective actions can be taken for poorly performing queries identified via these reports. Some of these actions are listed below:

- Ensure that appropriate indexes are available
- Change current indexes to include additional columns used in a problem query, which necessitates dropping and re-creating the index — a potentially long process for large tables
- Recode the queries for efficiency (that is joins versus sub-selects, use stage 1 predicates, eliminate usage of OR and so on.)
- Eliminate any unnecessary joins.

Summary

Many reports can be produced from the DB2 catalog to aid in performance monitoring. This article details some sample reports in three basic areas: navigational, physical and plan. Each area provides a proactive means of preventing performance bottlenecks before they occur. They are only suggestions. Changing them to suit specific needs should be easy due to the ad hoc nature of SQL. Good luck in tuning. ⊜

ABOUT THE AUTHOR

Craig S. Mullins is an officer of Mellon Bank Corporation in Pittsburgh, PA. He is the DB2 system and database administrator with more than five years of experience in database management systems. He is also a vice-president and co-founder of ASSET, Inc., a Pittsburgh-based consulting and software development firm.