Using AS/400 Database Monitor

To Identify and Tune SQL Queries

by

Rick Peterson
Dale Weber
Richard Odell
Greg Leibfried

AS/400 System Performance
IBM Rochester Lab

May 2000

Table of Contents

Introduction

The purpose of this white paper is to assist the reader in understanding the basics of how to identify and tune the performance of Structured Query Language (SQL) statements accessing DB2® Universal Database for AS/400 (DB2 UDB for AS/400). Database Monitor for AS/400 (DB Monitor) is a tool that can be used to analyze database performance problems after SQL requests have been run. The tuning tips and examples may be used to demonstrate how to get the most out of both DB2 UDB for AS/400 and query optimizer when using SQL. The reader should be aware that usage of DB Monitor is technically demanding and IBM offers Learning Services courses on database tools and analysis.

The DB monitor tool has been part of OS/400® since Version 3 Release 6. An AS/400 performance analyst will use the DB monitor to gather database and performance data generated when SQL queries are executed. Then, using customized SQL programs or examples included in this paper, the analyst will be able to view, analyze and conclude the most appropriate actions to be taken in order to generate the most efficient SQL queries possible for their application.

What is the Database Monitor for AS/400 tool?

The Database Monitor for AS/400 tool is an AS/400 based tool used to gather performance related statistics for SQL queries run on AS/400. Data collected by the DB monitor is stored in an AS/400 database file where it can be queried to help identify and tune performance problem areas. Results from the DB monitor can be useful for batch jobs and online transactions; also, the results can be used to look at SQL queries from a global system level for a specific job or a specific query. DB monitor data is most useful if the user has a basic knowledge of AS/400 query optimization techniques.

Collecting DB Monitor Data

Use STRDBMON command or STRPFRMON with STRDBMON(*YES) command to start the DB monitor. The DB monitor will collect information on previously started jobs or new jobs started after the monitor collection has begun. Because of the volume of data collected, try to gather data for a specific job only. This makes analysis easier and keeps the DB monitor file smaller. If this is not possible, collect data on all of the jobs and use queries to select the specific jobs of interest.

It should also be noted that when the DB monitor is gathering data, a significant amount of CPU utilization (20 - 30 percent) and disk usage may be temporarily required.

Start Database Monitor (STRDBMON) Parameters

The following are STRDBMON parameters and how they are used:

OUTFILE
The file name for the results file is required, but the library name is optional. The file will be created if it does not exist; it will be reused if it exists.

OUTMBR
This parameter defaults to the first member in the file. Specify the “ADD” or “REPLACE” option (the default is “REPLACE”). The “ADD” option will cause the new results to be appended to the end of the file.

JOB
This parameter defaults to the job issuing the STRDBMON command. The user can specify one job or *ALL jobs — no subsetting allowed. Two DB monitors can collect data on the same job.

TYPE
This parameter allows the user to specify the type of data to be collected — *SUMMARY, which is the default option, or *DETAIL. For most cases, *SUMMARY provides all of the necessary analysis data.

FRCRCD
This parameter allows the user to specify how often to force monitor records to the results file. For most cases, the default of *CALC is acceptable. The user can specify a larger number to reduce the overhead of the DB monitor; a smaller number will increase the overhead.

COMMENT
This parameter allows the user to provide a description of the collection. This comment is included for the *3018* record ID (discussed later in this paper).

If the DB monitor is started using the Start Performance Monitor (STRPFRMON) command, JOB(*ALL) will be used for the JOB option and data will be placed in the QAPMDBMON file in the QPFRDATA library using the same member name as specified for the STRPFRMON command. Note that the user needs to use the End Database Monitor command (ENDDBMON *ALL) to end the DB monitor for all jobs.

End Database Monitor (ENDDBMON) Parameters

The following are End Database Monitor (ENDDBMON) parameters and their functions:

JOB
The user can specify a particular job name or end all jobs (*ALL). If a particular job name is used, the DB monitor will only end the monitor that was started with that same job name. It is possible to end one monitor on a job and still have another monitor collecting on that same job

COMMENT
This parameter allows the user to provide a description of the data collection. This comment is included for the *3018* record ID (discussed later in this paper).

DB Monitor Record Types

Each record contained in the DB monitor file contains a record type field. The DB monitor uses the QQRID field to describe the type of information gathered in the particular record.

Following are the DB monitor record types most often used for performance analysis:

Record types most often used (QQRID value)

1000 — SQL summary record

3000 — Arrival sequence

3001 — Using existing index

3002 — Index created

3003 — Query sort

3004 — Temporary results file

3006 — Access plan rebuilt

3007 — Index optimization data

3010 — Host variable and ODP implementation

DB Monitor Record Types — Other record types

3005 — Table locked

3008 — Subquery processing

3014 — Generic query information

3018 — STRDBMON and ENDDBMON data

3019 — Records retrieved detail (only with *DETAIL)

Global DB Monitor Data Fields

Following are data fields that are common to all record types:

QQJOB — Job name

QQUSER — Job user name

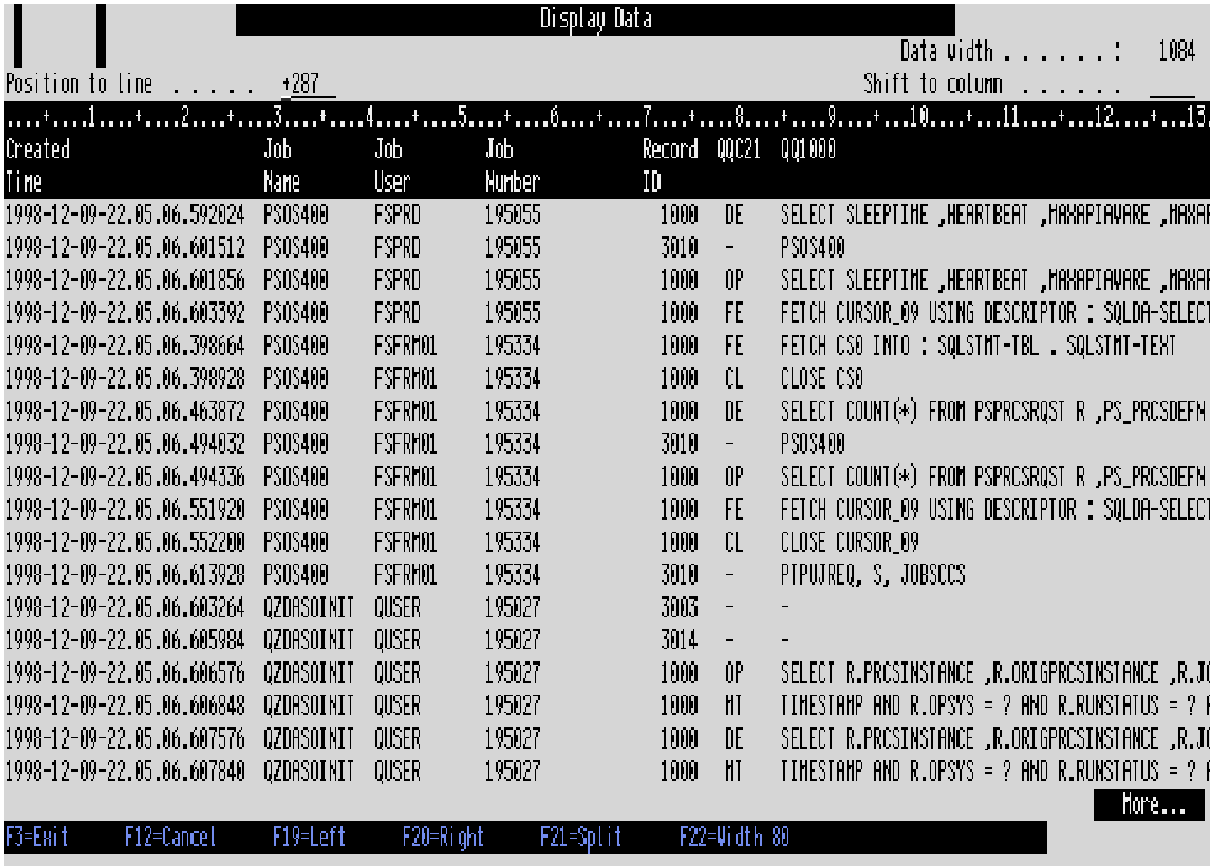
QQJNUM — Job number

The job number is very useful when multiple jobs are collected in one DB monitor file.

QQTIME — Time that the record was created

The time record can be useful when trying to find out what queries were running in a given time period.

Global Data Fields



```
                              Display Data
                                                         Data width . . . . . . :   1084
Position to line  . . . . .  +287                        Shift to column  . . . . . .  ____
....+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13
Created                     Job         Job         Job                 Record  QQC21  QQ1000
Time                        Name        User        Number              ID
1998-12-09-22.05.06.592024  PSOS400     FSPRD       195055                1000   DE    SELECT SLEEPTIME ,HEARTBEAT ,MAXAPIAWARE ,MAXAF
1998-12-09-22.05.06.601512  PSOS400     FSPRD       195055                3010   -     PSOS400
1998-12-09-22.05.06.601856  PSOS400     FSPRD       195055                1000   OP    SELECT SLEEPTIME ,HEARTBEAT ,MAXAPIAWARE ,MAXAF
1998-12-09-22.05.06.603392  PSOS400     FSPRD       195055                1000   FE    FETCH CURSOR_09 USING DESCRIPTOR : SQLDA-SELECT
1998-12-09-22.05.06.398664  PSOS400     FSFRM01     195334                1000   FE    FETCH CS0 INTO : SQLSTMT-TBL . SQLSTMT-TEXT
1998-12-09-22.05.06.398928  PSOS400     FSFRM01     195334                1000   CL    CLOSE CS0
1998-12-09-22.05.06.463872  PSOS400     FSFRM01     195334                1000   DE    SELECT COUNT(*) FROM PSPRCSRQST R ,PS_PRCSDEFN
1998-12-09-22.05.06.494032  PSOS400     FSFRM01     195334                3010   -     PSOS400
1998-12-09-22.05.06.494336  PSOS400     FSFRM01     195334                1000   OP    SELECT COUNT(*) FROM PSPRCSRQST R ,PS_PRCSDEFN
1998-12-09-22.05.06.551920  PSOS400     FSFRM01     195334                1000   FE    FETCH CURSOR_09 USING DESCRIPTOR : SQLDA-SELECT
1998-12-09-22.05.06.552200  PSOS400     FSFRM01     195334                1000   CL    CLOSE CURSOR_09
1998-12-09-22.05.06.613928  PSOS400     FSFRM01     195334                3010   -     PTPUJREQ, S, JOBSCCS
1998-12-09-22.05.06.603264  QZDASOINIT  QUSER       195027                3003   -     -
1998-12-09-22.05.06.605984  QZDASOINIT  QUSER       195027                3014   -     -
1998-12-09-22.05.06.606576  QZDASOINIT  QUSER       195027                1000   OP    SELECT R.PRCSINSTANCE ,R.ORIGPRCSINSTANCE ,R.JO
1998-12-09-22.05.06.606848  QZDASOINIT  QUSER       195027                1000   MT    TIMESTAMP AND R.OPSYS = ? AND R.RUNSTATUS = ? A
1998-12-09-22.05.06.607576  QZDASOINIT  QUSER       195027                1000   DE    SELECT R.PRCSINSTANCE ,R.ORIGPRCSINSTANCE ,R.JO
1998-12-09-22.05.06.607840  QZDASOINIT  QUSER       195027                1000   MT    TIMESTAMP AND R.OPSYS = ? AND R.RUNSTATUS = ? A
                                                                                                                          More...
F3=Exit      F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

Query Optimization Records

Record types *3000-3008* and *3014* occur during a full open and can be referred to as optimization records. Optimization records are much like debug messages. These records are necessary to determine the access plan for any given query in the DB monitor data. These records use AS/400 10-character short names for all table, index, view and column names. It may be necessary to run separate queries to determine the corresponding SQL long name.

The actual SQL statement in the DB monitor data uses SQL long names.

Collecting Optimization Data in the DB Monitor

Optimization records in the DB monitor

Optimization records/data will not appear for queries which are already in reusable ODP mode when the monitor is started. To ensure the capture of this data for a batch job, start the DB monitor before the batch job starts and collect it over the entire length of the job (or as much as needed). For online transactions, start the monitor before connecting to the QZDASOINIT server job to ensure optimization data is collected. The STRDBMON JOB(*ALL) command is needed in both of these cases.

If optimization data was not collected for a given query, run the query using Start SQL (STRSQL) command or other tools and collect debug messages or DB monitor data. Obtain the query text from the DB monitor data or from the step mode trace. If the DB monitor is ended while the query is in progress, optimization data is collected but other data for that query (SQL text, etc.) is not.

Query Data Organization in Monitor Data

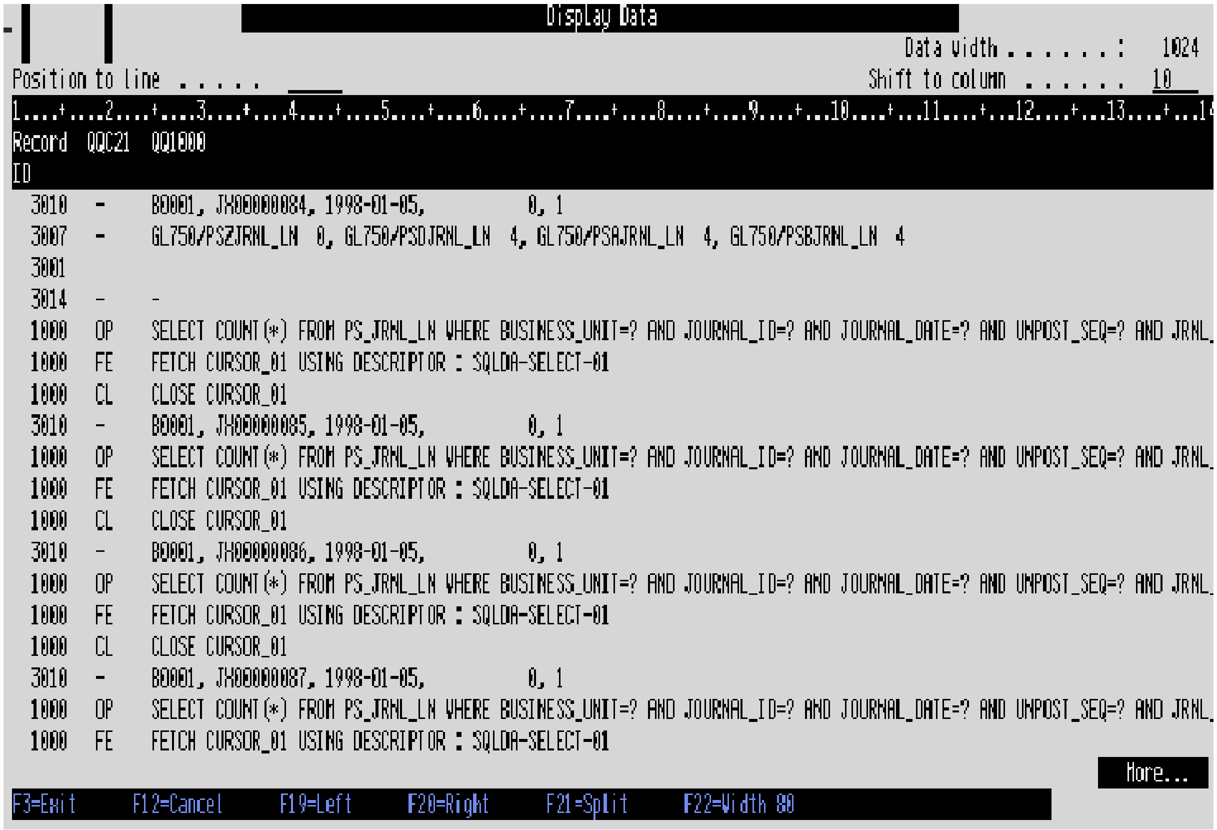
The first occurrence of a unique query within the job always results in full open. A "unique" query is one that requires a new ODP — SQL has determined that there is no existing ODP that can be used.

The presence of optimization records indicates a full open for an open, select into, update, delete, or insert operation. Optimization records are immediately followed by SQL summary records (QQRID=1000) for that operation.

Subsequent occurrences of this query within the same job either run in reusable ODP or non-reusable ODP mode. Non-reusable mode is indicated by the presence of optimization records each time a particular query is run (full open). Reusable ODP mode is indicated by only *3010* and *1000* records each time the given query is run (no optimization records or full open).

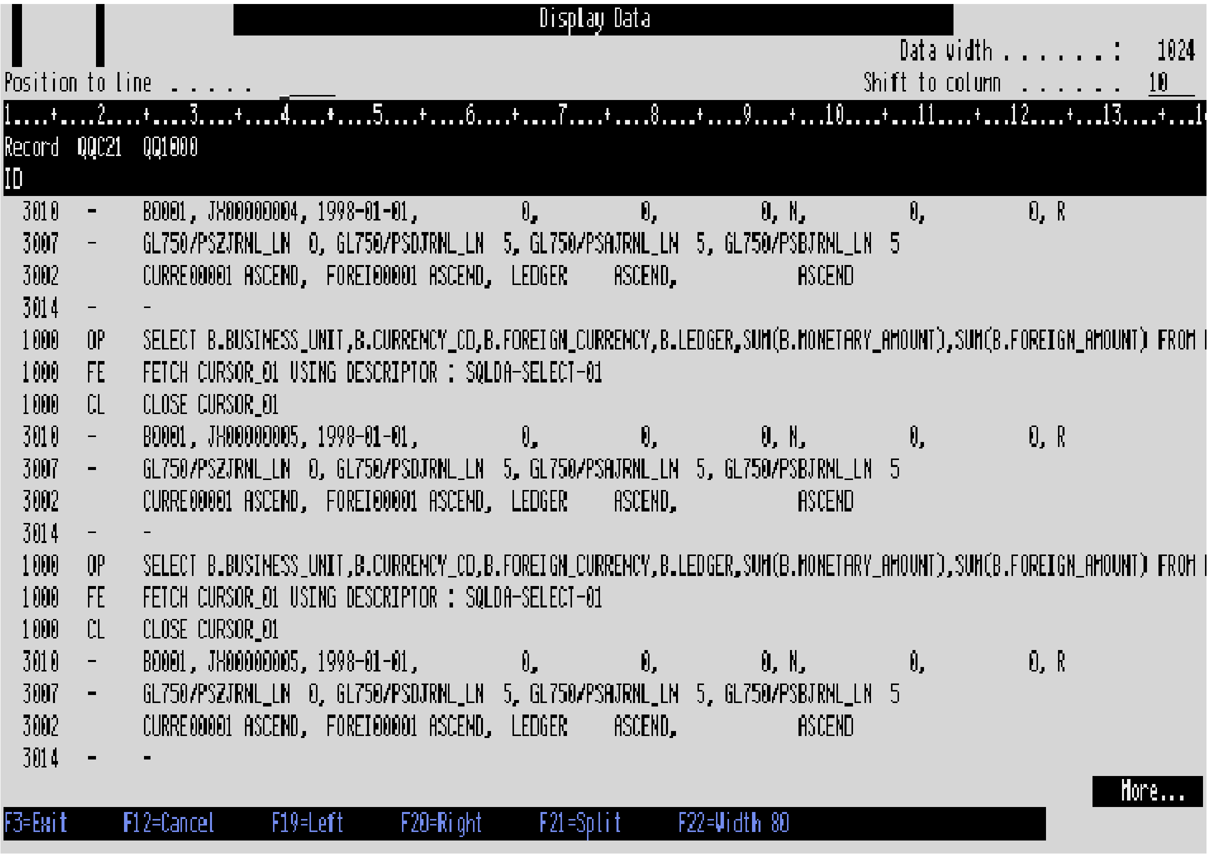
On the next page are two examples of what the Reusable ODP and Non-Reusable ODP Modes look like.

Reusable ODP Mode



```
                              Display Data
                                                      Data width . . . . . . :    1024
Position to line  . . . . .  _____                   Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record  QQC21   QQ1000
ID
  3010   -     B0001, JX00000084, 1998-01-05,          0, 1
  3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  4, GL750/PSAJRNL_LN  4, GL750/PSBJRNL_LN  4
  3001
  3014   -     -
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     B0001, JX00000085, 1998-01-05,          0, 1
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     B0001, JX00000086, 1998-01-05,          0, 1
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     B0001, JX00000087, 1998-01-05,          0, 1
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
                                                                                                                  More...
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

Non-Reusable ODP Mode



```
                              Display Data
                                                         Data width . . . . . . :    1024
Position to line  . . . . .  _____                       Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  QQC21  QQ1000
ID
  3010   -     BO001, JH00000004, 1998-01-01,          0,          0,          0, N,          0,          0, R
  3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL750/PSAJRNL_LN  5, GL750/PSBJRNL_LN  5
  3002         CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER     ASCEND,            ASCEND
  3014   -     -
  1000   OP    SELECT B.BUSINESS_UNIT,B.CURRENCY_CD,B.FOREIGN_CURRENCY,B.LEDGER,SUM(B.MONETARY_AMOUNT),SUM(B.FOREIGN_AMOUNT) FROM
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     BO001, JH00000005, 1998-01-01,          0,          0,          0, N,          0,          0, R
  3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL750/PSAJRNL_LN  5, GL750/PSBJRNL_LN  5
  3002         CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER     ASCEND,            ASCEND
  3014   -     -
  1000   OP    SELECT B.BUSINESS_UNIT,B.CURRENCY_CD,B.FOREIGN_CURRENCY,B.LEDGER,SUM(B.MONETARY_AMOUNT),SUM(B.FOREIGN_AMOUNT) FROM
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     BO001, JH00000005, 1998-01-01,          0,          0,          0, N,          0,          0, R
  3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL750/PSAJRNL_LN  5, GL750/PSBJRNL_LN  5
  3002         CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER     ASCEND,            ASCEND
  3014   -     -
                                                                                                 More...
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

Linking Query Instances in Monitor Data

The data in the DB monitor file is arranged chronologically. This can make it difficult to find all instances of a unique query. Use the QQUCNT and QQI5 fields to view specific query instances.

- QQUCNT — Unique number given for each unique query within a job

  QQUCNT links together all DB monitor records associated with all instances of a unique query within a job, including optimization records and all *3010* and *1000* SQL summary records. The QQUCNT value assigned at full open time stays constant for all subsequent instances of that query. Non-ODP SQL operations (prepare, describe, commit) have QQUCNT = 0 and thus can't be linked to a query. But, the QQ1000 field in the prepare or describe *1000* record will contain the prepared SQL text.

- QQI5 — Refresh counter

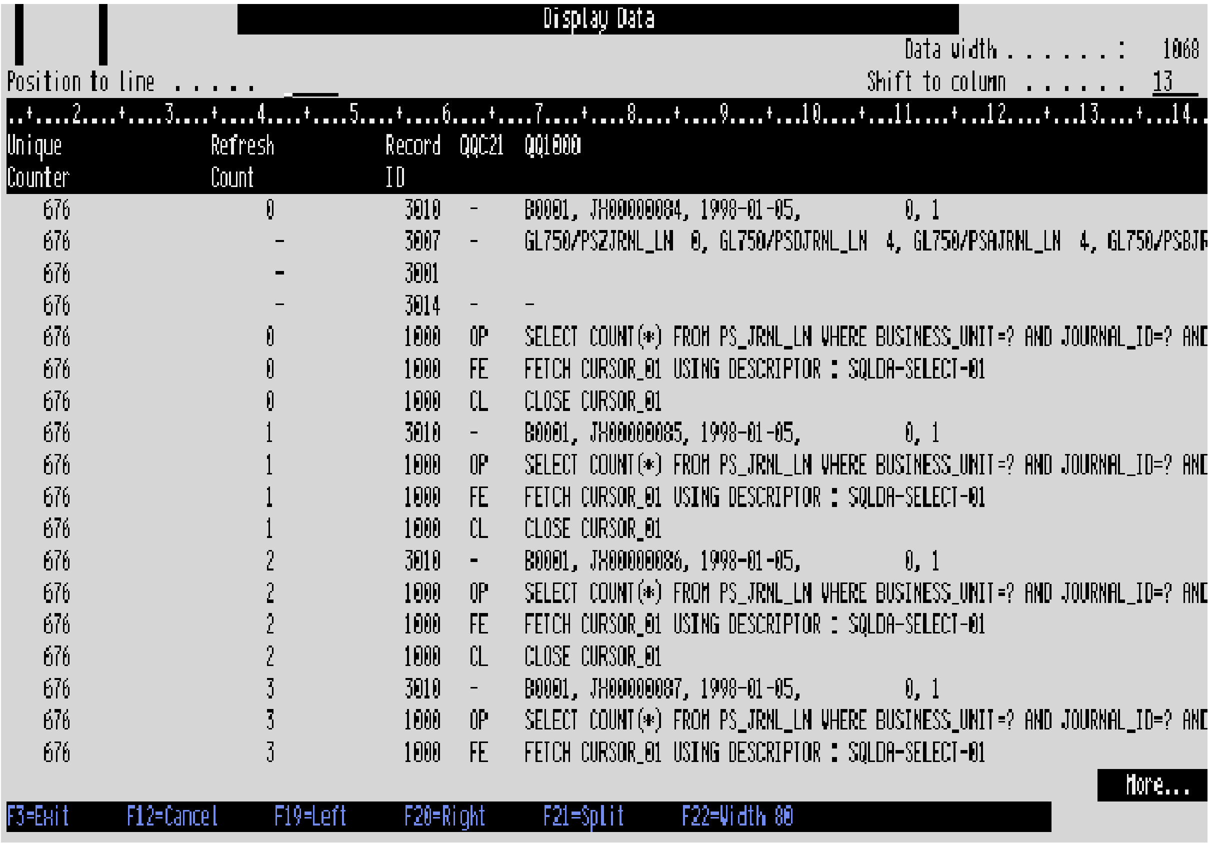
  The QQ15 record specifies the instance number for a unique query. It is used in conjunction with the QQUCNT value to look at a specific instance of a query and is only valid on *3010* and *1000* SQL summary records.

- Non-ODP *1000* records (commit, prepare, etc.) have QQI5 = 0.
- QQUCNT is not set for optimization records.

A full open occurs when the SQL operation is either an update, insert, delete or open and the QQI5 record is 0.

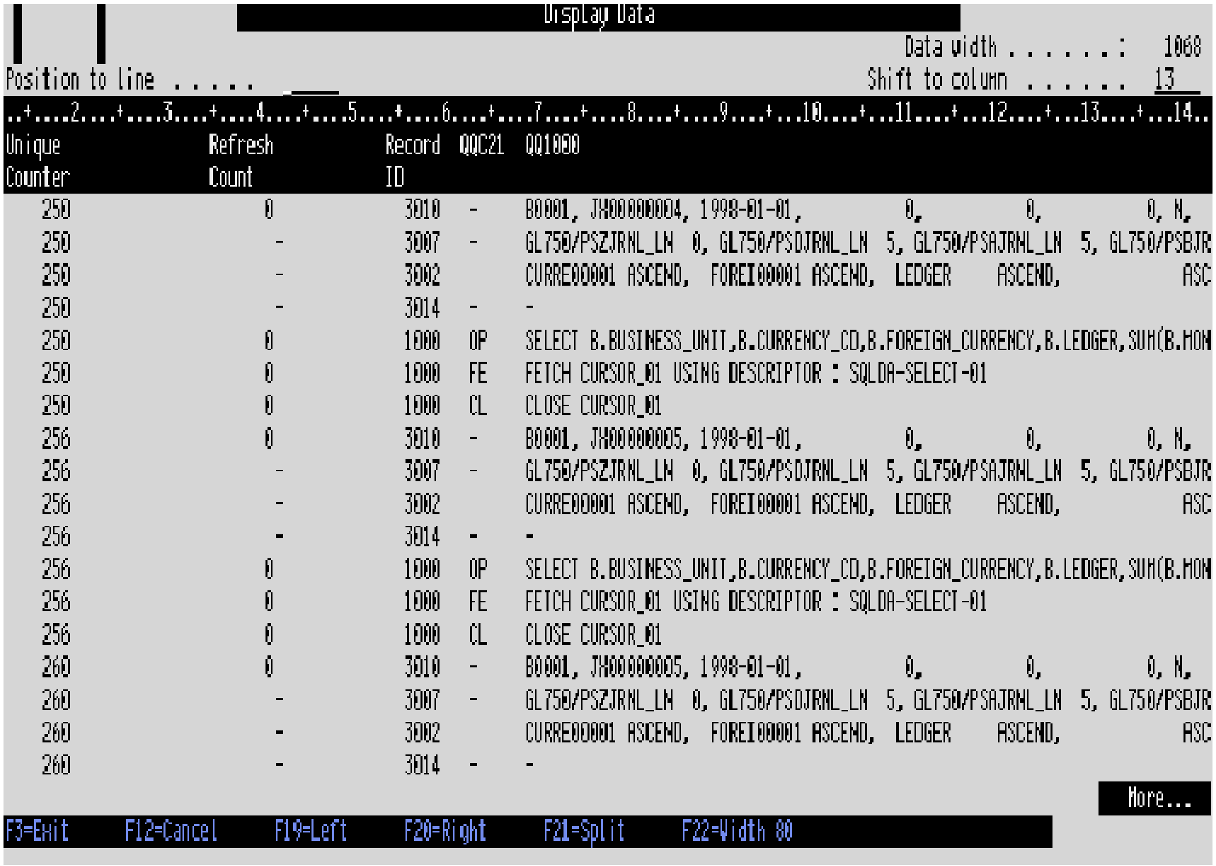
See below for examples of the QQUCNT/QQ15 — Reusable ODP mode and QQUCNT/QQ15 — Non-Reusable ODP mode

QQUCNT/QQI5 — Reusable ODP Mode



```
                                  Display Data
                                                              Data width . . . . . . :   1068
Position to line  . . . . .  ____                             Shift to column  . . . . . .  13__
..+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14..
Unique               Refresh         Record  QQC21  QQ1000
Counter              Count           ID
   676                   0             3010   -     B0001, JH00000084, 1998-01-05,          0, 1
   676                    -            3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  4, GL750/PSAJRNL_LN  4, GL750/PSBJR
   676                    -            3001
   676                    -            3014   -     -
   676                   0             1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND
   676                   0             1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
   676                   0             1000   CL    CLOSE CURSOR_01
   676                   1             3010   -     B0001, JH00000085, 1998-01-05,          0, 1
   676                   1             1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND
   676                   1             1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
   676                   1             1000   CL    CLOSE CURSOR_01
   676                   2             3010   -     B0001, JH00000086, 1998-01-05,          0, 1
   676                   2             1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND
   676                   2             1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
   676                   2             1000   CL    CLOSE CURSOR_01
   676                   3             3010   -     B0001, JH00000087, 1998-01-05,          0, 1
   676                   3             1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND
   676                   3             1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
                                                                                                                   More...
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

QQUCNT/QQI5 — Non-Reusable ODP Mode



```
                                  Display Data
                                                              Data width . . . . . . :   1068
Position to line  . . . . .  ____                             Shift to column  . . . . . .  13__
..+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14..
Unique               Refresh         Record  QQC21  QQ1000
Counter              Count           ID
   250                   0             3010   -     B0001, JH00000004, 1998-01-01,          0,          0,          0, N,
   250                    -            3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL750/PSAJRNL_LN  5, GL750/PSBJR
   250                    -            3002         CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER    ASCEND,             ASC
   250                    -            3014   -     -
   250                   0             1000   OP    SELECT B.BUSINESS_UNIT,B.CURRENCY_CD,B.FOREIGN_CURRENCY,B.LEDGER,SUM(B.MON
   250                   0             1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
   250                   0             1000   CL    CLOSE CURSOR_01
   256                   0             3010   -     B0001, JH00000005, 1998-01-01,          0,          0,          0, N,
   256                    -            3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL750/PSAJRNL_LN  5, GL750/PSBJR
   256                    -            3002         CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER    ASCEND,             ASC
   256                    -            3014   -     -
   256                   0             1000   OP    SELECT B.BUSINESS_UNIT,B.CURRENCY_CD,B.FOREIGN_CURRENCY,B.LEDGER,SUM(B.MON
   256                   0             1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
   256                   0             1000   CL    CLOSE CURSOR_01
   260                   0             3010   -     B0001, JH00000005, 1998-01-01,          0,          0,          0, N,
   260                    -            3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL750/PSAJRNL_LN  5, GL750/PSBJR
   260                    -            3002         CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER    ASCEND,             ASC
   260                    -            3014   -     -
                                                                                                                   More...
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

Reported DB Monitor Record Types

Each of the DB monitor record types provide a specific type of information gathered by the DB monitor as it collects data during SQL performance runs. Let's review the DB monitor record types and their functions.

3010 Record — Host Variable and ODP Implementation

The 3010 record shows substitution values for host variables or parameter markers in the query text (refer to QQ1000 field in the *1000* record). This record appears just prior to each instance of an open, update, delete or insert with subselect. This record does not appear for insert with values. Data may not match up exactly for updates with parameter markers in the SET clause.

Values (separated by commas) correspond left to right with host variables/parameter markers in the corresponding SQL statement. All values show up as characters; no quotes or other indicators denote the value type. All floating point values show up as *F.

Most commonly used fields:

QQ1000 — Host variable or parameter marker values

QQI5 — Refresh count

QQC11 — ODP implementation (reusable or non-reusable)

1000 Record - SQL Statement Summary

The *1000* record is the basic record type for any SQL query analysis. One record exists for each SQL operation (open,update,close,commit, etc.).

Most commonly used fields:

QQ1000 — Prepared text of SQL statement

> Literals in the original SQL text may be replaced by parameter markers in prepared text if SQL was able to convert them during prepare (desired). For original SQL text, use literal values from matching *3010* record in the place of parameter markers or obtain the text from the step mode file using the QQSTIM timestamp from this record.

QQC21 — Type of SQL operation (OP, FE, CL, UP, IN, DL, ...)

> 'MT' in this field indicates a continuation record for SQL statements that exceed 1000 characters. FE (fetch) records are summary records — one per open.

QQI2 — Number of rows updated/inserted/deleted

QQI3 — Number of rows fetched (only on FE records)

> The QQI3 field shows the actual number of rows fetched, not the number of fetch attempts.

QQI5 — Refresh counter (use in conjuction with QQUCNT)

> A full open occurs when QQI5=0 and QQC21 is UP, DL, IN or OP.

QQI4 — Elapsed time for this operation in milliseconds.

QQI6 — Elapsed time for this operation in microseconds.

QQSTIM — Timestamp for start of SQL operation (microsecond granularity).

QQETIM — Timestamp for end of SQL operation (microsecond granularity).

QQETIM-QQSTIM shows elapsed time for operation in seconds down to microsecond granularity.

For FE records, use QQI6 or QQI4; QQETIM-QQSTIM is not valid.

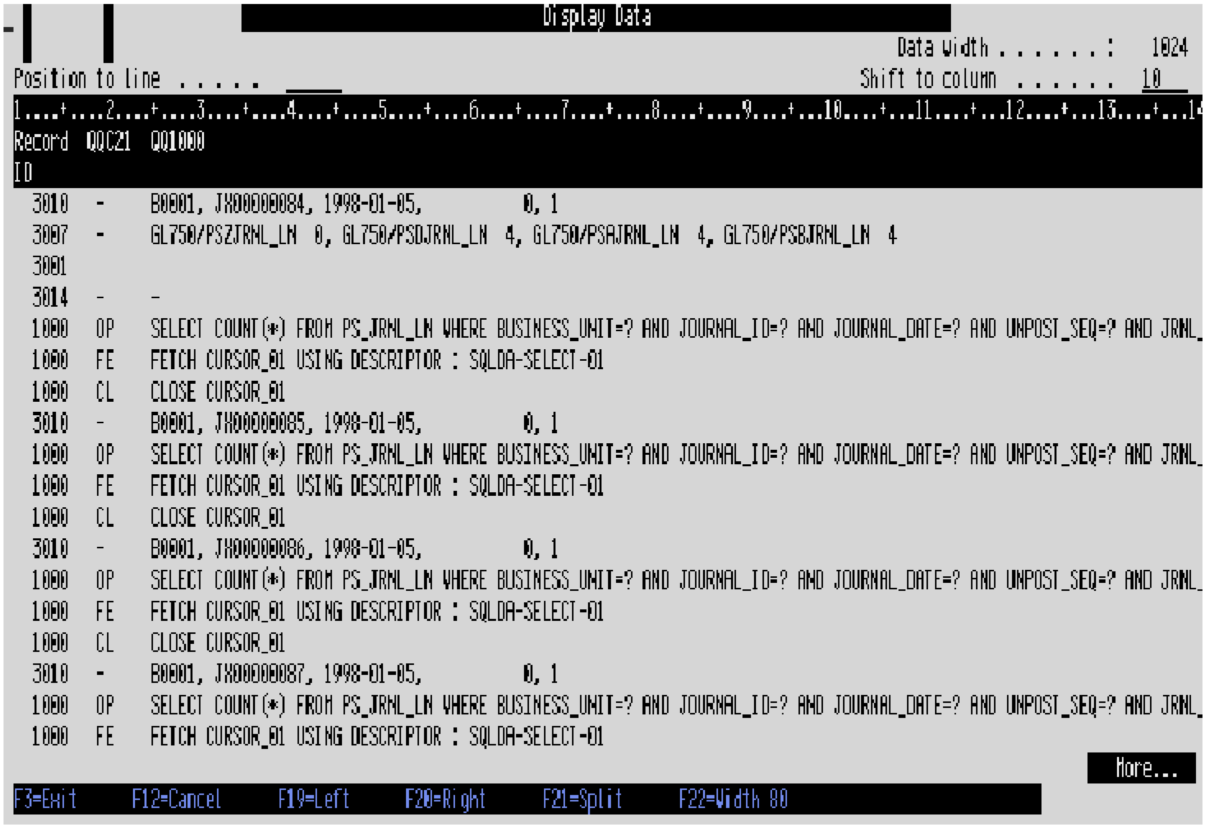
Other commonly used fields include:

QQC22 — Access plan rebuild and reason code

QQC103 and QQC104 — Package name and package library name

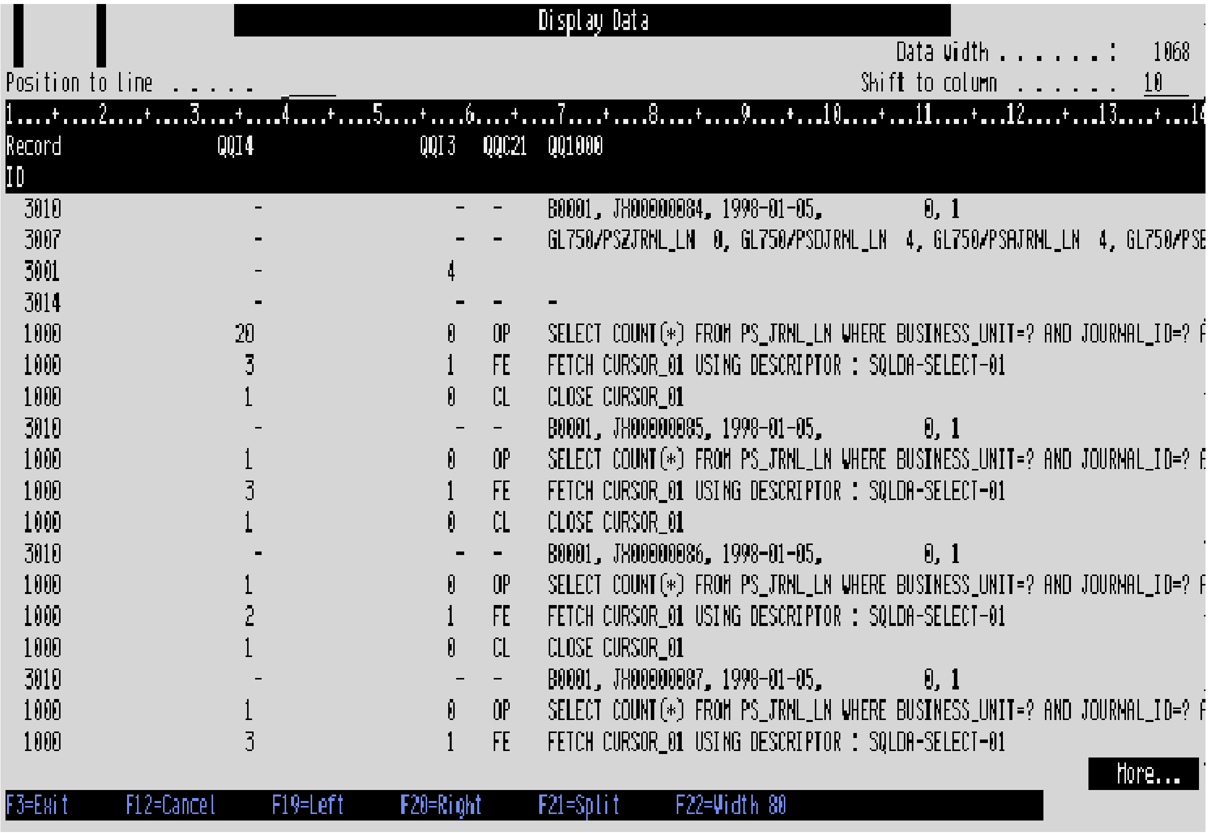
QQC181 — Cursor name
Here are some examples of a few of these fields:

3010 Record Data



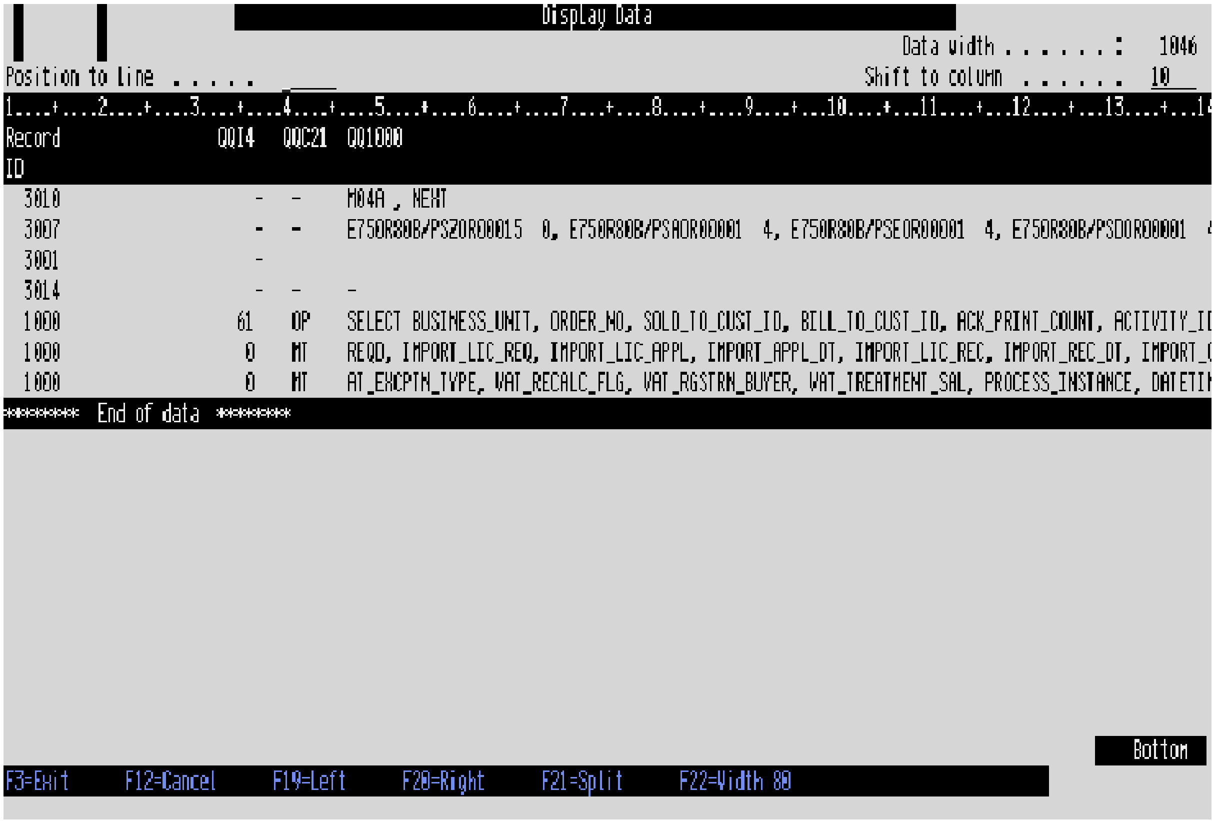
```
                                  Display Data
                                                              Data width . . . . . . :   1024
Position to line  . . . . .  _____                            Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  QQC21  QQ1000
ID
  3010   -     B0001, JX00000084, 1998-01-05,          0, 1
  3007   -     GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  4, GL750/PSAJRNL_LN  4, GL750/PSBJRNL_LN  4
  3001
  3014   -     -
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     B0001, JX00000085, 1998-01-05,          0, 1
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     B0001, JX00000086, 1998-01-05,          0, 1
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000   CL    CLOSE CURSOR_01
  3010   -     B0001, JX00000087, 1998-01-05,          0, 1
  1000   OP    SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? AND JOURNAL_DATE=? AND UNPOST_SEQ=? AND JRNL_
  1000   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
                                                                                                                      More...
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

1000 Record Data



```
                                    Display Data
                                                               Data width . . . . . . :    1068
Position to line  . . . . .    _____                           Shift to column  . . . . . .  10___
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record               QQI4               QQI3  QQC21  QQ1000
ID
  3010                  -                  -    -    B0001, JH00000084, 1998-01-05,          0, 1
  3007                  -                  -    -    GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  4, GL750/PSAJRNL_LN  4, GL750/PSE
  3001                  -                  4
  3014                  -                  -    -    -
  1000                 20                  0    OP   SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? A
  1000                  3                  1    FE   FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000                  1                  0    CL   CLOSE CURSOR_01
  3010                  -                  -    -    B0001, JH00000085, 1998-01-05,          0, 1
  1000                  1                  0    OP   SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? A
  1000                  3                  1    FE   FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000                  1                  0    CL   CLOSE CURSOR_01
  3010                  -                  -    -    B0001, JH00000086, 1998-01-05,          0, 1
  1000                  1                  0    OP   SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? A
  1000                  2                  1    FE   FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000                  1                  0    CL   CLOSE CURSOR_01
  3010                  -                  -    -    B0001, JH00000087, 1998-01-05,          0, 1
  1000                  1                  0    OP   SELECT COUNT(*) FROM PS_JRNL_LN WHERE BUSINESS_UNIT=? AND JOURNAL_ID=? A
  1000                  3                  1    FE   FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
                                                                                                            More...
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

1000 Records with Extended SQL Text



```
                                    Display Data
                                                               Data width . . . . . . :    1046
Position to line  . . . . .    _____                           Shift to column  . . . . . .  10___
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record               QQI4   QQC21  QQ1000
ID
  3010                  -    -     M04A , NEXT
  3007                  -    -     E750R80B/PSZOR00015  0, E750R80B/PSAOR00001  4, E750R80B/PSEOR00001  4, E750R80B/PSDOR00001  4
  3001                  -
  3014                  -    -     -
  1000                 61    OP    SELECT BUSINESS_UNIT, ORDER_NO, SOLD_TO_CUST_ID, BILL_TO_CUST_ID, ACK_PRINT_COUNT, ACTIVITY_ID
  1000                  0    MT    REQD, IMPORT_LIC_REQ, IMPORT_LIC_APPL, IMPORT_APPL_DT, IMPORT_LIC_REC, IMPORT_REC_DT, IMPORT_(
  1000                  0    MT    AT_EXCPTN_TYPE, VAT_RECALC_FLG, VAT_RGSTRN_BUYER, VAT_TREATMENT_SAL, PROCESS_INSTANCE, DATETIM
********  End of data  ********




                                                                                                            Bottom
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

3007 Record — Index Optimization and Selection

The *3007* record shows all indexes evaluated for a given file, including: which one, if any, was selected for use in this query, which were not selected, and why. Reason codes are listed next to each index. A reason code of 0 indicates that the index was selected. Other codes are the same as those in the second level text of CPI432C and CPI432D messages.

This record indicates whether the optimizer timed out while evaluating the indexes. Indexes are evaluated in order from newest to oldest — in the same order as shown by DSPDBR for the file, excluding views. To ensure an index is evaluated, delete and recreate it — then it will be first on the list. The record will not appear if the indexes do not exist or if only one index exists and it was selected (see *3001* record for this file).

Most commonly used fields:

QQPTFN — File name

QQPTLN — File library name

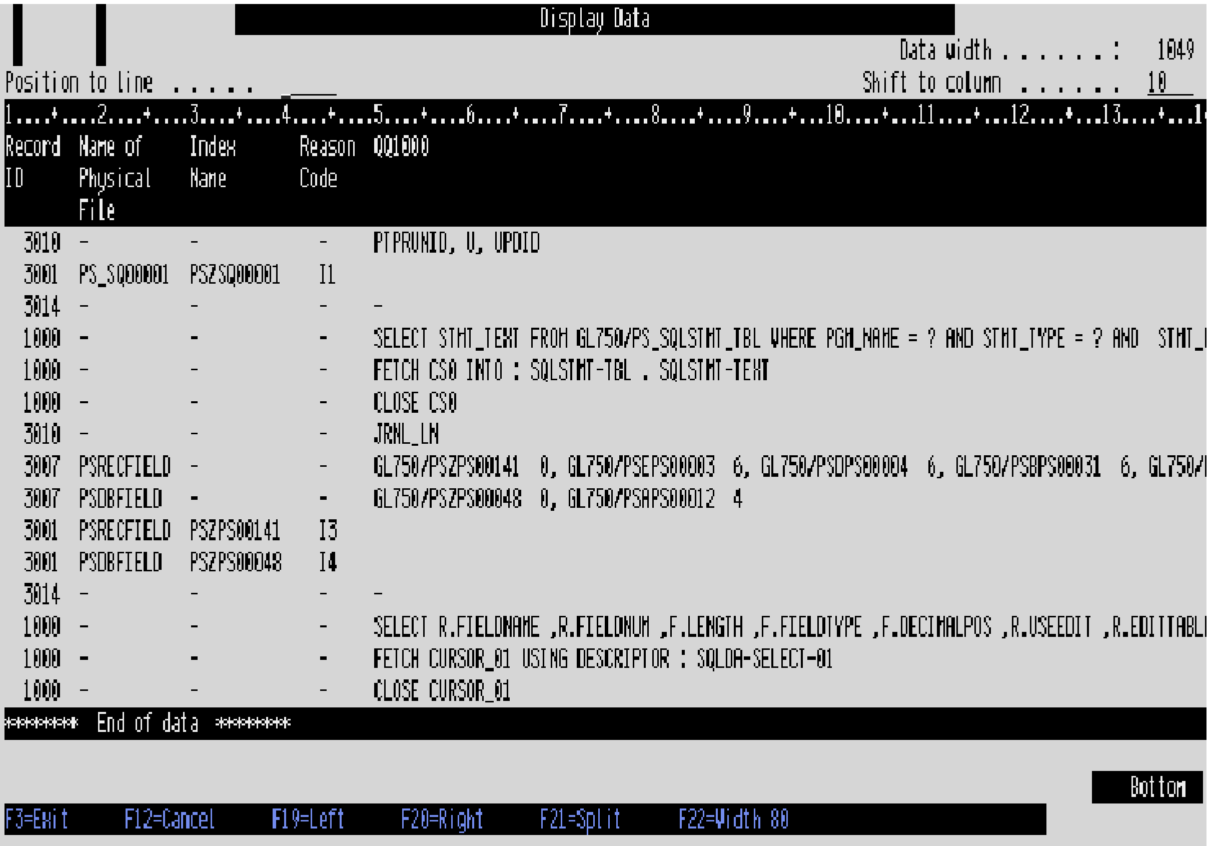
QQC11 — Optimizer timed out (Y or N)

QQ1000 — Contains library qualified index names, each with a reason code

An index from index build may still occur for the index that was selected (look for the *3002* record). If a timeout occurred, only those indexes that were evaluated will be listed.

See the example on the next page for what the 3007 Record Data looks like.

3007 Record Data



```
                                    Display Data
                                                              Data width . . . . . . :   1049
Position to line  . . . . .  _____                             Shift to column  . . . . . .  10___
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  Name of     Index       Reason  QQ1000
ID      Physical    Name        Code
        File
  3010  -           -             -     PTPRUNID, U, UPDID
  3001  PS_SQ00001  PSZSQ00001    I1
  3014  -           -             -     -
  1000  -           -             -     SELECT STMT_TEXT FROM GL750/PS_SQLSTMT_TBL WHERE PGM_NAME = ? AND STMT_TYPE = ? AND  STMT_
  1000  -           -             -     FETCH CS0 INTO : SQLSTMT-TBL . SQLSTMT-TEXT
  1000  -           -             -     CLOSE CS0
  3010  -           -             -     JRNL_LN
  3007  PSRECFIELD  -             -     GL750/PSZPS00141  0, GL750/PSEPS00003  6, GL750/PSDPS00004  6, GL750/PSBPS00031  6, GL750/
  3007  PSDBFIELD   -             -     GL750/PSZPS00048  0, GL750/PSAPS00012  4
  3001  PSRECFIELD  PSZPS00141    I3
  3001  PSDBFIELD   PSZPS00048    I4
  3014  -           -             -     -
  1000  -           -             -     SELECT R.FIELDNAME ,R.FIELDNUM ,F.LENGTH ,F.FIELDTYPE ,F.DECIMALPOS ,R.USEEDIT ,R.EDITTABL
  1000  -           -             -     FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000  -           -             -     CLOSE CURSOR_01
********  End of data  ********
                                                                                                  Bottom
F3=Exit      F12=Cancel     F19=Left      F20=Right     F21=Split     F22=Width 80
```

3002 Record — Index Create

The 3002 record shows instances in which the database optimizer decided that existing indexes are too costly or do not have the right key order for join, group by, or order by clauses. Refer to the *3007* record (discussed earlier) for this file to see why existing indexes were not selected. The newly created indexes are temporary and are not usable by other ODPs.

A temporary index build **does not** mean that the ODP is non-reusable. The database optimizer tries to reuse the temporary index for each execution of the specific query but, in some cases, cannot. For example, if the selection built into the temporary index changes with each run of the query, the temporary index may not be reused.

If the temporary index build is done during the full open for this query but the query goes into reusable ODP mode, then the temporary index is reusable.

If a particular query is run multiple times and a temporary index is built each time, a permanent index must be created to avoid the index build and to make the ODP reusable.

Indexes are never built for selection alone; they always involve a join or a group by or order by clause. No name is given to the temporary index. *TEMP is used in subsequent monitor records.

Most commonly used fields:

QQPTFN — Table name for which the index is built

> A table name of *N indicates the temporary results table (3004 record) — an index build is unavoidable.

QQPTLN — Table library name

> This field is blank if the file name is *N.

QQIFNM — Name of the index

> This field is blank if the index is built over the entire table. It will contain the index name if a index from an index build occurred.

QQILNM — Index library name

> This field is blank if the index is built over the entire table.

QQRCOD — Reason the index build was done

I2 — ordering or grouping

I3 — selection and ordering/grouping

I4 — nested loop join

I1 is listed in the DB monitor guide, but will not show up for temporary index builds

QQTOTR — Number of rows in table

QQRIDX — Number of entries in temporary index

QQSTIM — Timestamp for start of index build

QQETIM — Timestamp for end of index build

QQETIM-QQSTIM shows the elapsed time for the index build. Long running builds or builds that are repeated many times and result in a fair number of full opens are prime candidates for being replaced with permanent indexes.

QQ1000 — Contains join, order by, or group by keys used in index build

This field indicates whether the key is ASC or DESC — this is important for permanent indexes. It does not include additional selection that may have been used to build the index. If *MAP is one of the keys listed, the index build **cannot** be avoided but this does not necessarily mean that the ODP is non-reusable.

If QQTOTR=QQRIDX, then the selection is probably not built in and the permanent index can generally be built using only the keys from the QQ1000 field.

If QQTOTR>QQRIDX, then the selection was built in. If so, it is necessary to use a combination of selection keys and the keys listed in the QQ1000 field to build a permanent index. It may be necessary to look in the query text for keys in ANDed equals predicates and other selective comparisons.

If QQIDXA=Y, QQIDXD will contain good selection keys. However, it still may be necessary to look at the query text if there are problems getting the database optimizer to choose a created index. It is generally best to build a permanent index with good selection keys first, followed by join, order by, or group by keys.

QQIDXA — Index advised (Y or N)

If 'N', QQI2 and QQIDXD will not contain data.

QQI2 — Number of primary keys in QQIDXD

The QQI2 field contains the number of keys over which key positioning can be used.

QQIDXD — Suggested primary and secondary keys for index (selection only)

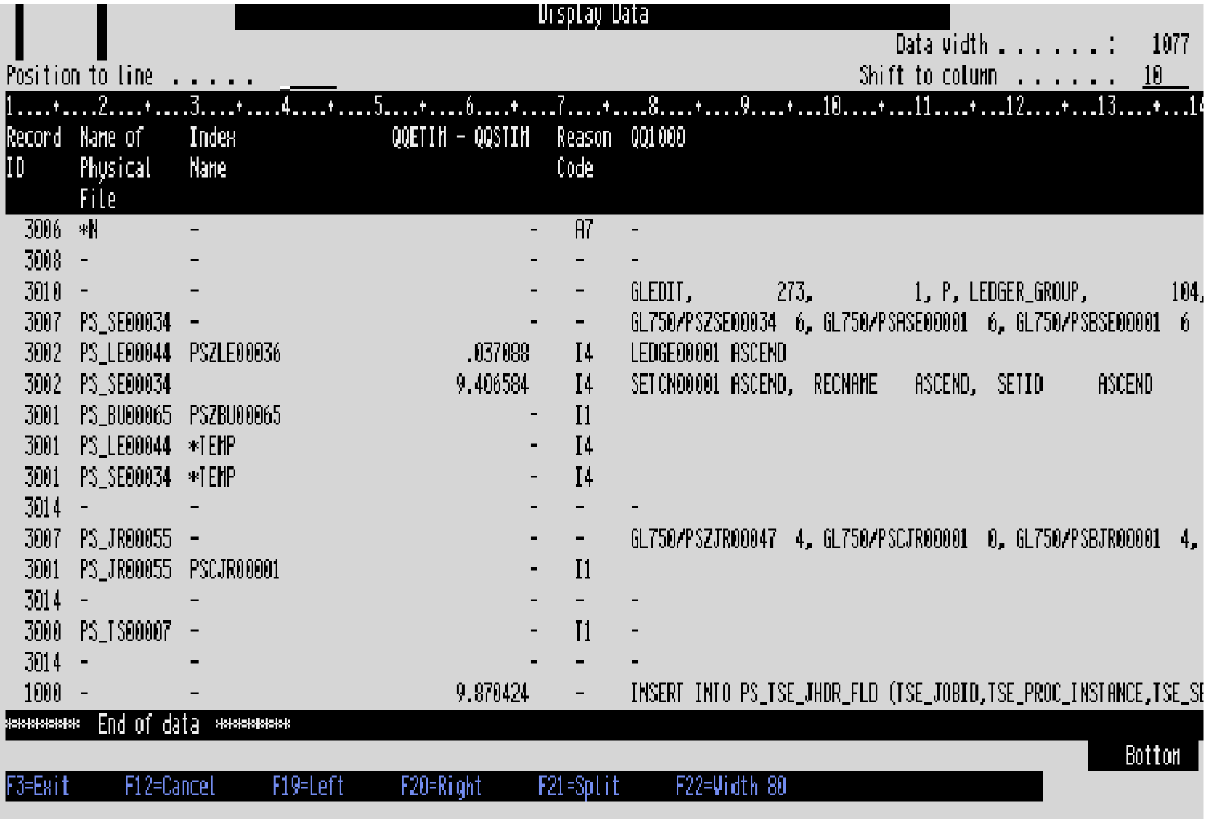
The QQIDXD field can contain both primary and secondary keys. Starting from the left, QQI2 tells how many keys are considered primary. Other keys are considered less selective (secondary). This field will be blank if an existing index contains most or all of the recommended selection.

If keys are listed, use the most selective ones combined with keys from the QQ1000 field. It is still important to include the most selective keys since the optimizer is estimating. Even if an index is not advised, it may still be best to try to create an index using the selection from the SQL statement, if it is a good selection.

Remember that building a permanent index that the optimizer will use is an iterative process, but it often has significant paybacks.

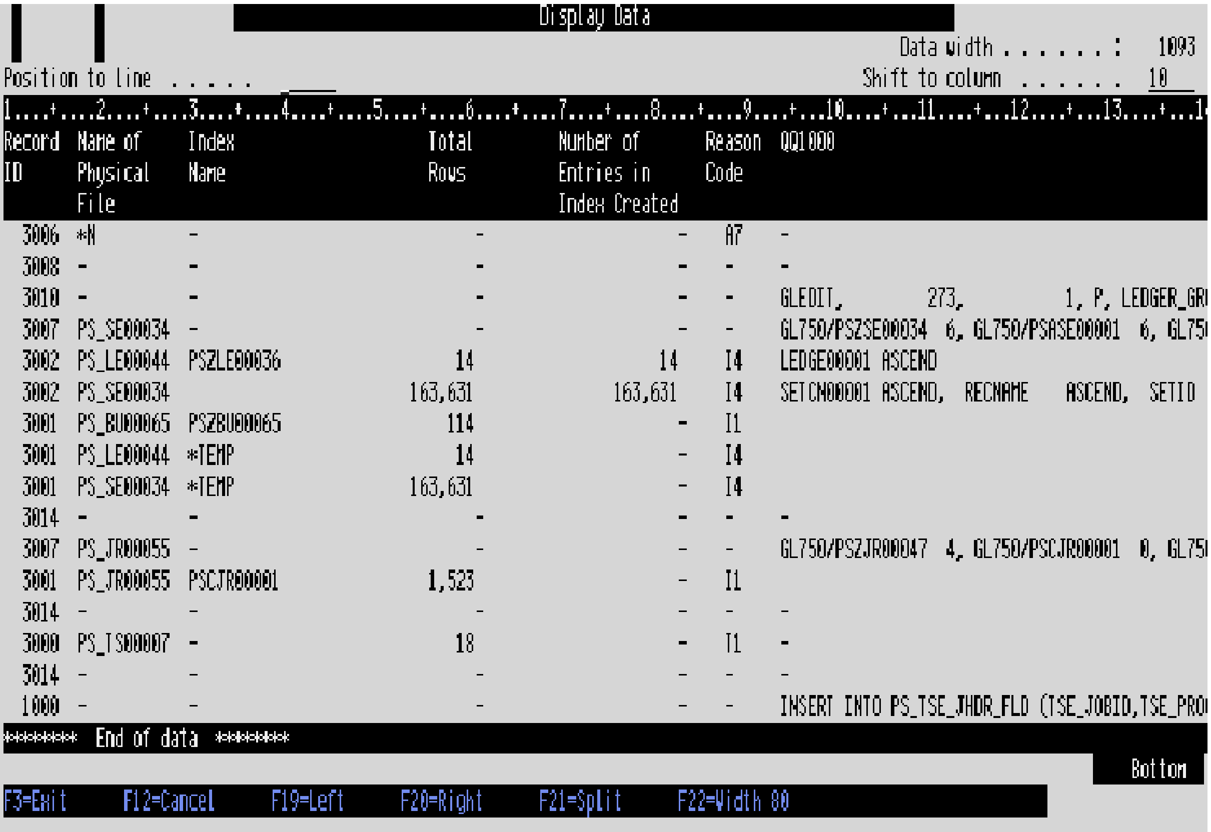
Here are some examples for what these screens will resemble:

3002 Record with QQETIM-QQSTIM



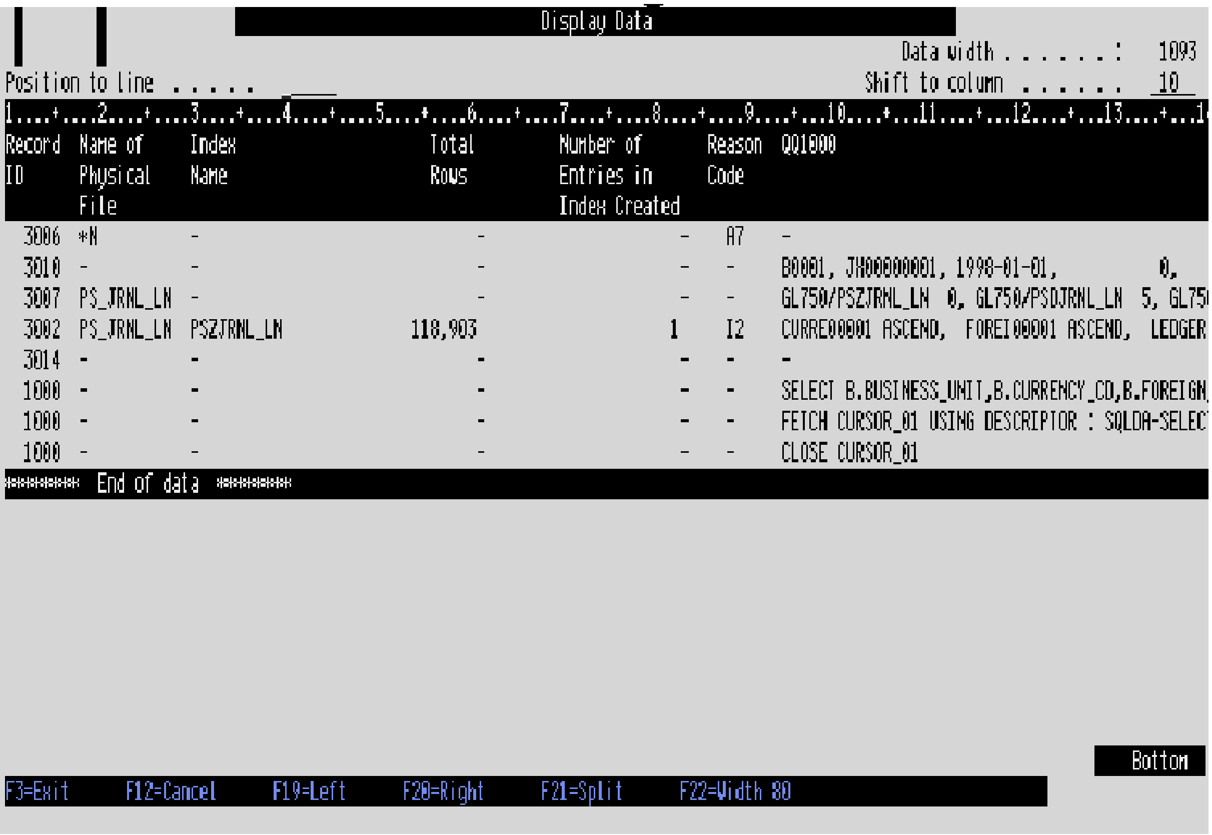
```
                                   Display Data
                                                          Data width . . . . . . :   1077
Position to line  . . . . .  _____                        Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record  Name of      Index                  QQETIM - QQSTIM  Reason  QQ1000
ID      Physical     Name                                    Code
        File
  3006  *N           -                                   -     A7    -
  3008  -            -                                   -     -     -
  3010  -            -                                   -     -     GLEDIT,        273,         1, P, LEDGER_GROUP,        104,
  3007  PS_SE00034   -                                   -     -     GL750/PSZSE00034  6, GL750/PSASE00001  6, GL750/PSBSE00001  6
  3002  PS_LE00044   PSZLE00036                    .037088     I4    LEDGE00001 ASCEND
  3002  PS_SE00034                                9.406584     I4    SETCN00001 ASCEND,  RECNAME    ASCEND,  SETID      ASCEND
  3001  PS_BU00065   PSZBU00065                          -     I1
  3001  PS_LE00044   *TEMP                               -     I4
  3001  PS_SE00034   *TEMP                               -     I4
  3014  -            -                                   -     -     -
  3007  PS_JR00055   -                                   -     -     GL750/PSZJR00047  4, GL750/PSCJR00001  0, GL750/PSBJR00001  4,
  3001  PS_JR00055   PSCJR00001                          -     I1
  3014  -            -                                   -     -     -
  3000  PS_TS00007   -                                   -     T1    -
  3014  -            -                                   -     -     -
  1000  -            -                            9.870424     -     INSERT INTO PS_TSE_JHDR_FLD (TSE_JOBID,TSE_PROC_INSTANCE,TSE_SE
********  End of data  ********
                                                                                            Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

3002 Record — Index Create for Join



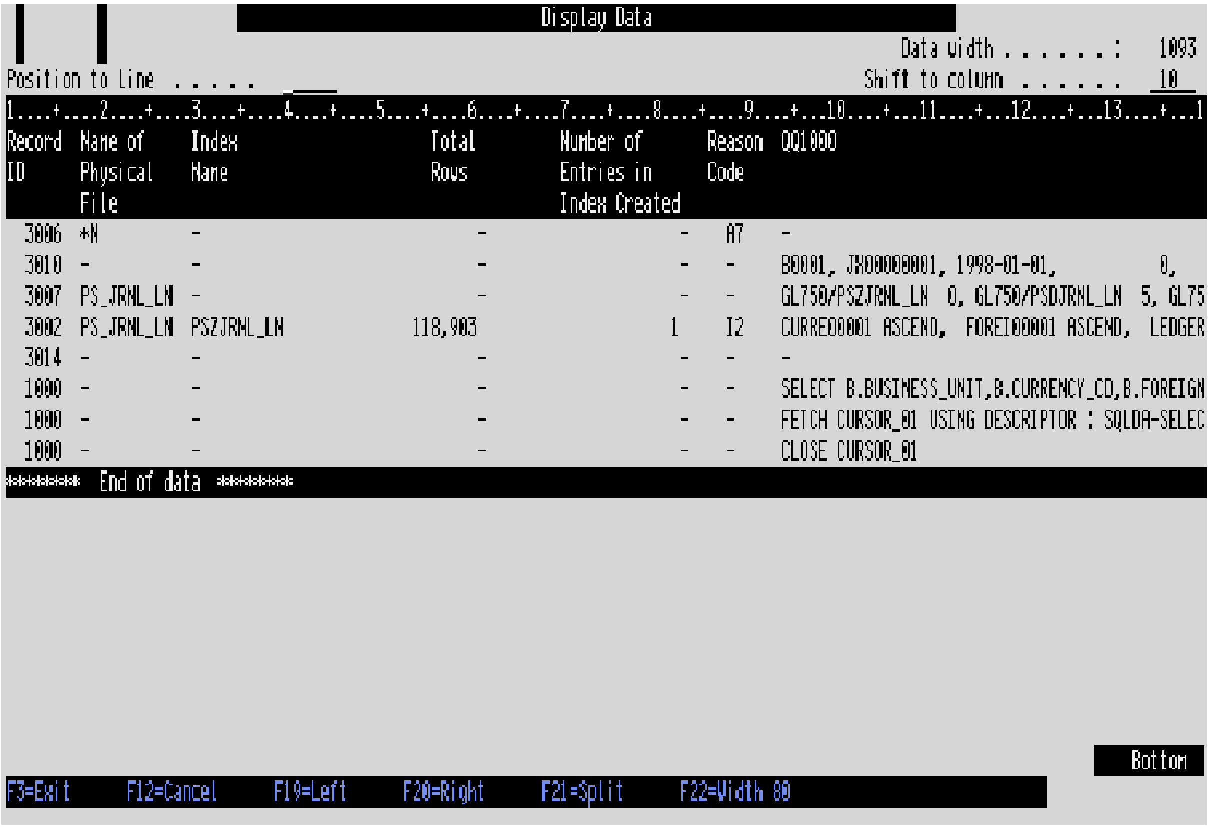
```
                                   Display Data
                                                          Data width . . . . . . :   1093
Position to line  . . . . .  _____                        Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record  Name of      Index                 Total         Number of       Reason  QQ1000
ID      Physical     Name                  Rows          Entries in      Code
        File                                             Index Created
  3006  *N           -                         -                    -     A7    -
  3008  -            -                         -                    -     -     -
  3010  -            -                         -                    -     -     GLEDIT,        273,         1, P, LEDGER_GRO
  3007  PS_SE00034   -                         -                    -     -     GL750/PSZSE00034  6, GL750/PSASE00001  6, GL750
  3002  PS_LE00044   PSZLE00036               14                   14     I4    LEDGE00001 ASCEND
  3002  PS_SE00034                       163,631              163,631     I4    SETCN00001 ASCEND,  RECNAME    ASCEND,  SETID
  3001  PS_BU00065   PSZBU00065              114                    -     I1
  3001  PS_LE00044   *TEMP                    14                    -     I4
  3001  PS_SE00034   *TEMP               163,631                    -     I4
  3014  -            -                         -                    -     -     -
  3007  PS_JR00055   -                         -                    -     -     GL750/PSZJR00047  4, GL750/PSCJR00001  0, GL750
  3001  PS_JR00055   PSCJR00001            1,523                    -     I1
  3014  -            -                         -                    -     -     -
  3000  PS_TS00007   -                        18                    -     T1    -
  3014  -            -                         -                    -     -     -
  1000  -            -                         -                    -     -     INSERT INTO PS_TSE_JHDR_FLD (TSE_JOBID,TSE_PRO
********  End of data  ********
                                                                                            Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

Index Create for Ordering and Selection



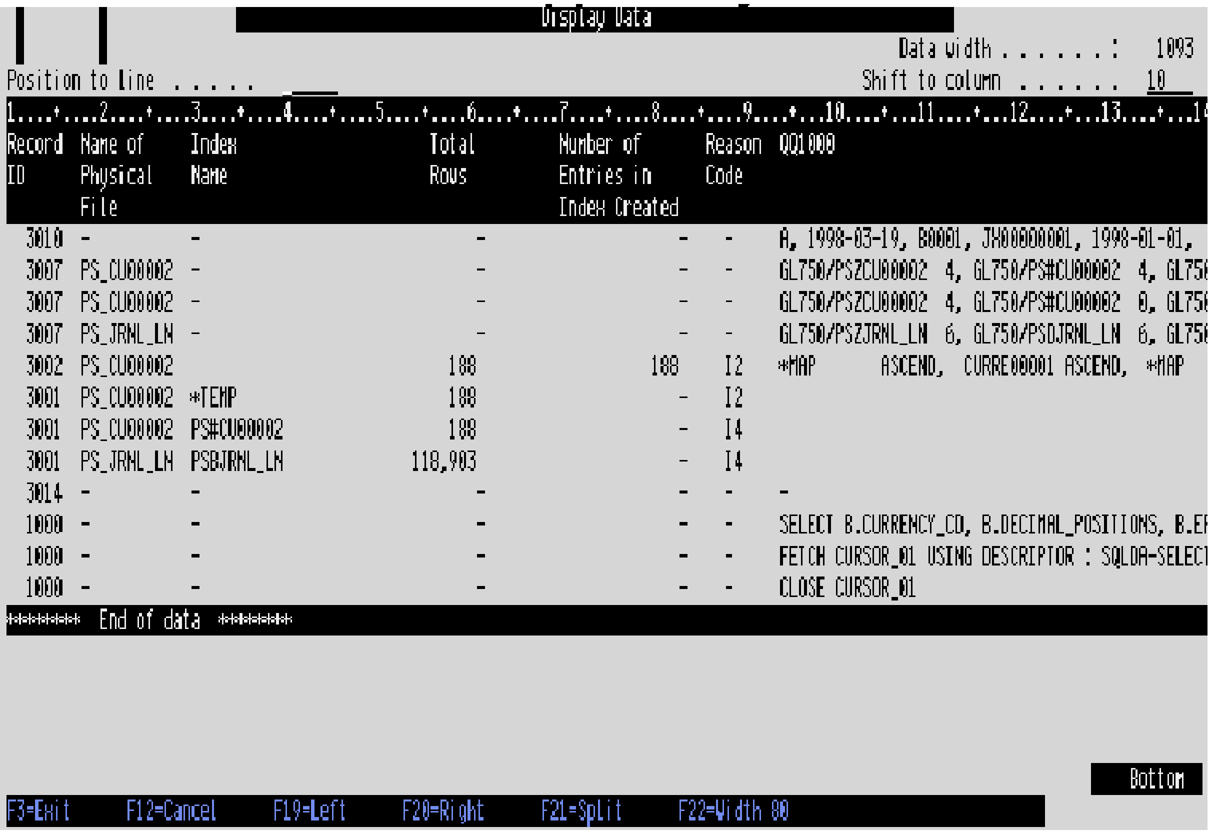
```
                                 Display Data
                                                          Data width . . . . . . :   1093
Position to line  . . . . .  _____                        Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  Name of     Index                        Total       Number of        Reason  QQ1000
ID      Physical    Name                         Rows        Entries in       Code
        File                                                 Index Created
  3006  *N          -                                -                -    A7     -
  3010  -           -                                -                -    -      B0001, JH00000001, 1998-01-01,            0,
  3007  PS_JRNL_LN  -                                -                -    -      GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL75
  3002  PS_JRNL_LN  PSZJRNL_LN                 118,903                1    I2     CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER
  3014  -           -                                -                -    -      -
  1000  -           -                                -                -    -      SELECT B.BUSINESS_UNIT,B.CURRENCY_CD,B.FOREIGN
  1000  -           -                                -                -    -      FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELEC
  1000  -           -                                -                -    -      CLOSE CURSOR_01
******** End of data ********
                                                                                                           Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

Index Advisor for Previous Index Create



```
                                 Display Data
                                                          Data width . . . . . . :   1093
Position to line  . . . . .  _____                        Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  Name of     Index                        Total       Number of        Reason  QQ1000
ID      Physical    Name                         Rows        Entries in       Code
        File                                                 Index Created
  3006  *N          -                                -                -    A7     -
  3010  -           -                                -                -    -      B0001, JX00000001, 1998-01-01,            0,
  3007  PS_JRNL_LN  -                                -                -    -      GL750/PSZJRNL_LN  0, GL750/PSDJRNL_LN  5, GL75
  3002  PS_JRNL_LN  PSZJRNL_LN                 118,903                1    I2     CURRE00001 ASCEND,  FOREI00001 ASCEND,  LEDGER
  3014  -           -                                -                -    -      -
  1000  -           -                                -                -    -      SELECT B.BUSINESS_UNIT,B.CURRENCY_CD,B.FOREIGN
  1000  -           -                                -                -    -      FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELEC
  1000  -           -                                -                -    -      CLOSE CURSOR_01
******** End of data ********
                                                                                                           Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

Index Build with Mapped Key



```
                                       Display Data
                                                                       Data width . . . . . . :   1093
Position to line  . . . . .  _____                                     Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record Name of    Index                      Total        Number of     Reason  QQ1000
ID     Physical   Name                       Rows         Entries in    Code
       File                                               Index Created
  3010 -          -                              -                  -   -      A, 1998-03-19, B0001, JX00000001, 1998-01-01,
  3007 PS_CU00002 -                              -                  -   -      GL750/PSZCU00002  4, GL750/PS#CU00002  4, GL75
  3007 PS_CU00002 -                              -                  -   -      GL750/PSZCU00002  4, GL750/PS#CU00002  0, GL75
  3007 PS_JRNL_LN -                              -                  -   -      GL750/PSZJRNL_LN  6, GL750/PSDJRNL_LN  6, GL75
  3002 PS_CU00002                              188                188   I2     *MAP      ASCEND,  CURRE00001 ASCEND,  *MAP
  3001 PS_CU00002 *TEMP                        188                  -   I2
  3001 PS_CU00002 PS#CU00002                   188                  -   I4
  3001 PS_JRNL_LN PSBJRNL_LN               118,903                  -   I4
  3014 -          -                              -                  -   -      -
  1000 -          -                              -                  -   -      SELECT B.CURRENCY_CD, B.DECIMAL_POSITIONS, B.E
  1000 -          -                              -                  -   -      FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELEC
  1000 -          -                              -                  -   -      CLOSE CURSOR_01
******** End of data ********




                                                                                                                        Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

3000 Record — Arrival Sequence (Table Scan)

The 3000 record points out queries in which the entire table is scanned without using an index. A table scan is generally acceptable in cases where a large portion of the file will be selected or the selected file contains a very small number of records. Otherwise, using an index usually provides better performance.

An insert with a subselect will have a *3000* record for the file being inserted into, but this is not a performance problem on its own. This **does not** indicate that the ODP is non-reusable. The record data may contain useful index advisor data. See the example on the next page.

Most commonly used fields:

QQPTFN — File name

QQPTLN — File library name

QQTOTR — Number of rows in table

Use QQTOTR to determine if the table scan was done for a significant number of rows.

QQRCOD — Reason code - why arrival sequence chosen

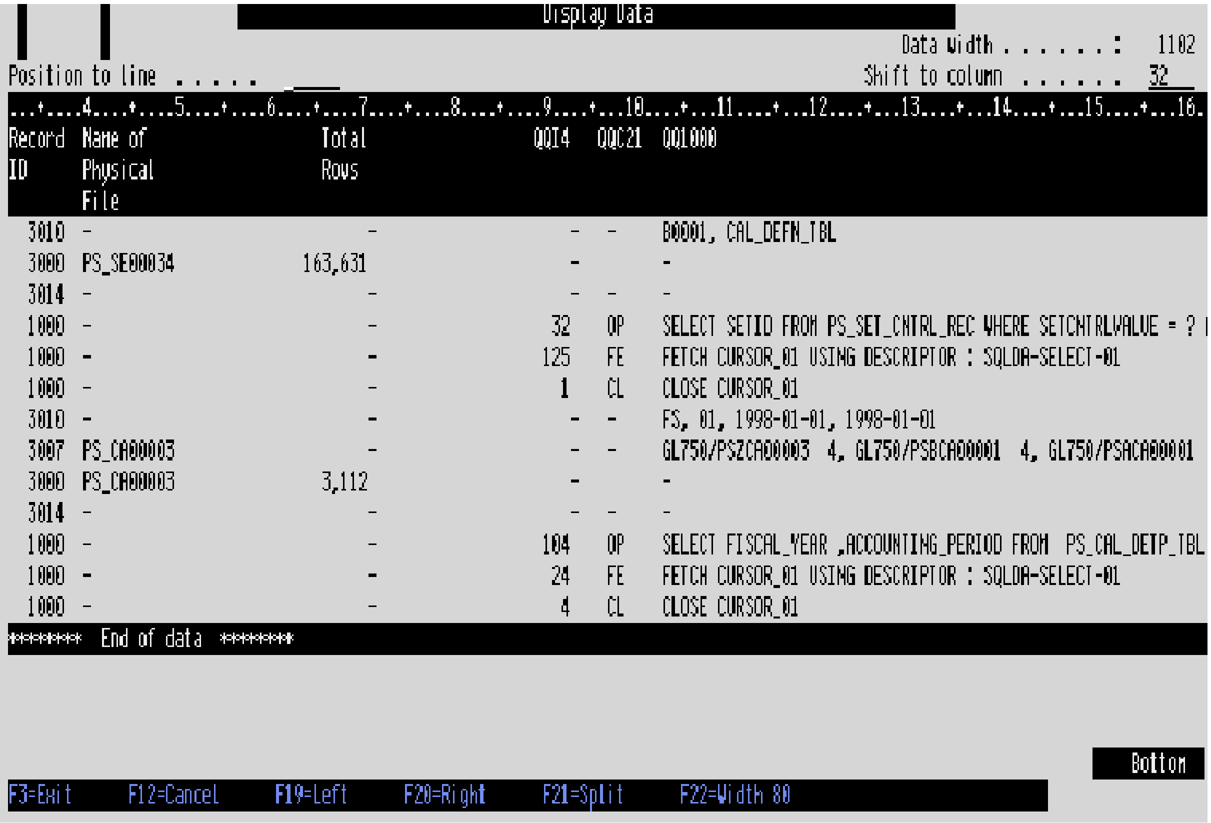
QQIDXA — Index advised (Y or N)

If the QQIDXA field is 'N', QQI2 and QQIDXD will not contain data.

QQI2 — Number of primary (key positioning) keys in QQIDXD field QQIDXD — Suggested keys for index (selection only)

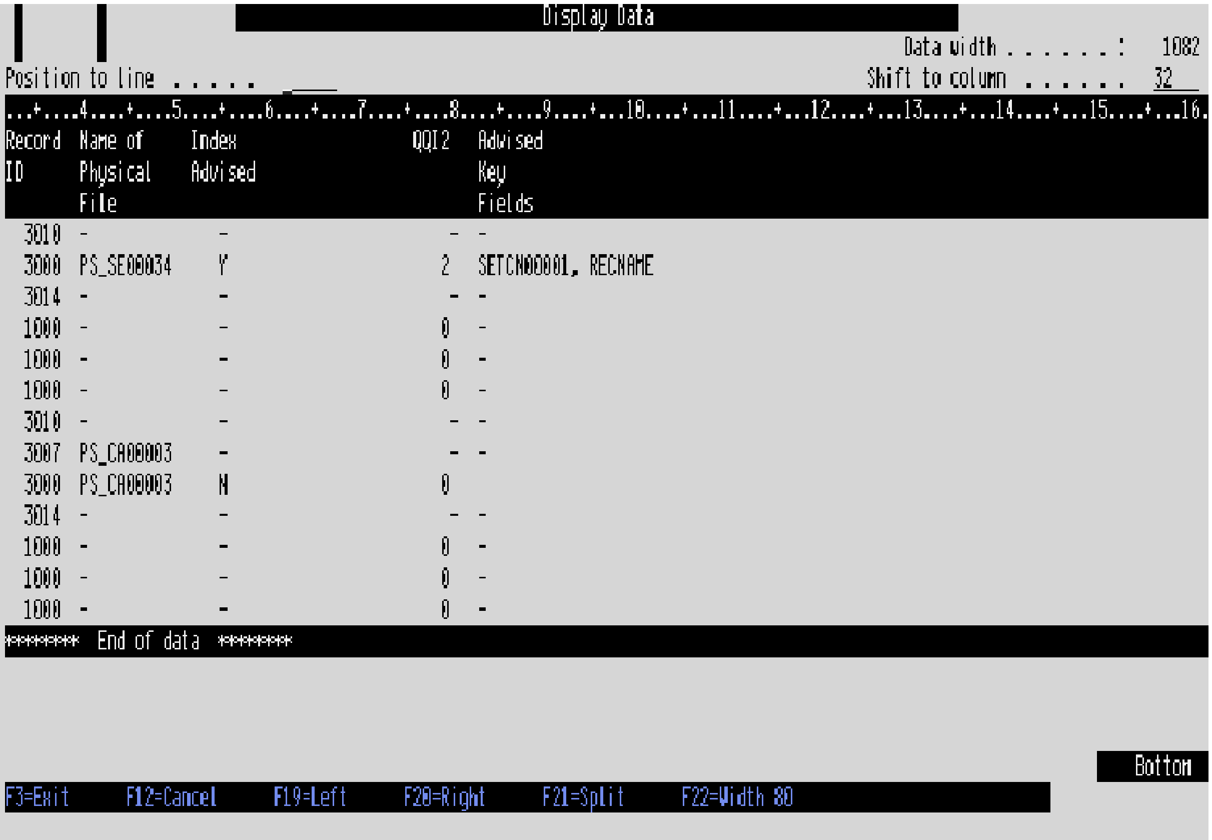
The QQIDXD field can contain both primary and secondary keys. Starting from the left, QQI2 tells how many keys are considered primary. Other keys are considered less selective (secondary). It is important to include the most selective keys since the database optimizer is estimating. Even if an index is not advised, it is still best to determine if a good index can be created for this table by looking at the selection in the SQL text. This is especially important if the cumulative time for this query is significant — this can be measured by the total of QQI6 or QQI4 values for the *1000* FE records for this query.

3000 Record Data



```
                                 Display Data
                                                         Data Width . . . . . . :   1102
Position to line  . . . . .   _____                   Shift to column  . . . . . .  32
...+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14....+...15....+...16.
Record Name of                   Total               QQI4   QQC21  QQ1000
ID     Physical                  Rows
       File
  3010 -                              -                 -    -     B0001, CAL_DEFN_TBL
  3000 PS_SE00034           163,631                     -          -
  3014 -                              -                 -    -     -
  1000 -                              -                32    OP    SELECT SETID FROM PS_SET_CNTRL_REC WHERE SETCNTRLVALUE = ?
  1000 -                              -               125    FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000 -                              -                 1    CL    CLOSE CURSOR_01
  3010 -                              -                 -    -     FS, 01, 1998-01-01, 1998-01-01
  3007 PS_CA00003                     -                 -    -     GL750/PSZCA00003  4, GL750/PSBCA00001  4, GL750/PSACA00001
  3000 PS_CA00003             3,112                     -          -
  3014 -                              -                 -    -     -
  1000 -                              -               104    OP    SELECT FISCAL_YEAR ,ACCOUNTING_PERIOD FROM  PS_CAL_DETP_TBL
  1000 -                              -                24    FE    FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000 -                              -                 4    CL    CLOSE CURSOR_01
********  End of data  ********
                                                                                     Bottom
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

Index Advisor for Previous 3000 Records



```
                                   Display Data
                                                           Data width . . . . . . :    1082
Position to line . . . . .  _____                          Shift to column . . . . . .  32
...+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14....+...15....+...16.
Record  Name of     Index                   QQI2  Advised
ID      Physical    Advised                       Key
        File                                      Fields
  3010  -             -                        -  -
  3000  PS_SE00034    Y                       2   SETCN00001, RECNAME
  3014  -             -                        -  -
  1000  -             -                       0   -
  1000  -             -                       0   -
  1000  -             -                       0   -
  3010  -             -                        -  -
  3007  PS_CA00003    -                        -  -
  3000  PS_CA00003    N                       0
  3014  -             -                        -  -
  1000  -             -                       0   -
  1000  -             -                       0   -
  1000  -             -                       0   -
********  End of data  ********
                                                                                        Bottom
F3=Exit    F12=Cancel    F19=Left    F20=Right    F21=Split    F22=Width 80
```

3001 Record — Using Existing Index

The 3001 record shows the index that will be used to access the file, and why it was chosen. If the index was chosen for join, additional information is given to help determine how the file "fits" in the join. The order of the 3001 records indicates the join order chosen by the optimizer.

Most commonly used fields:

QQPTFN — File name (*N indicates the temporary results table)

QQPTLN — File library name (blank if the file name is *N)

QQIFNM — Name of selected index (*TEMP if temporary index is used)

QQILNM — Index library name (blank if the index name is *TEMP)

QQRCOD - Reason the index was selected

I1 — selection only

I2 — ordering or grouping

I3 — selection and ordering/grouping

I4 — nested loop join

I5 — record selection using bitmap

Bitmap selection has a DB monitor record sequence of *3007*, *3000* and *3001*.

QQC21 — Join method (NL, MF, HJ)

QQC22 — Join type (IN, PO, EX)

QQC23 — Join operator (EQ, LT, GT, CP...)

QQTOTR — Number of rows in the table

QQAJN — Estimated number of joined rows from this table

> The QQAJN field helps to determine if the join order looks correct, but the user may still need to know how selective the join and selection criteria is on this table to be sure.

QQIDXA — Index advised (Y or N)

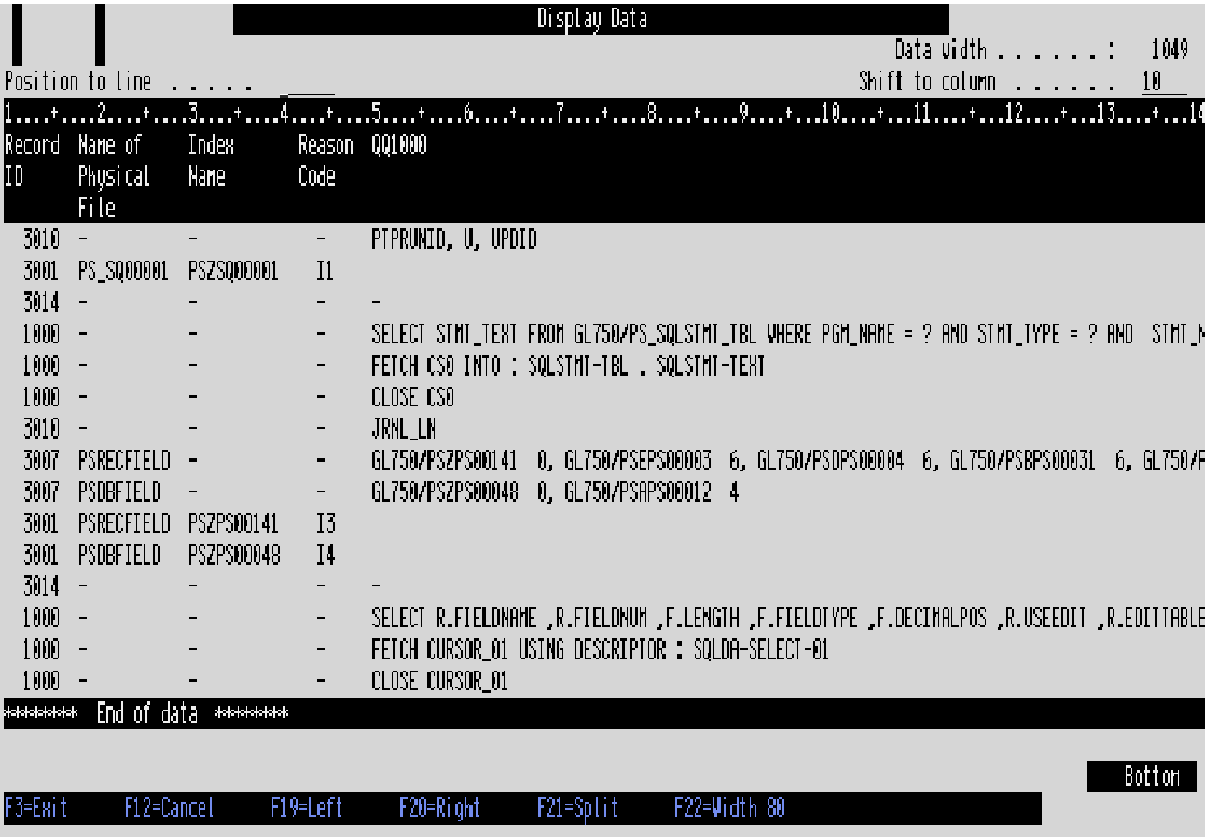
QQI2 — Number of primary (key positioning) keys in QQIDXD field

QQIDXD — Recommended primary and secondary keys for index on this table

Exercise caution: the keys listed can **possibly** be better than those in the selected index, but not always. Generally, the selected index will be fairly good or the database optimizer has chosen to build a temporary index that fits well. These keys are for selection only an the user also needs to consider join/order by/group by clause criteria.

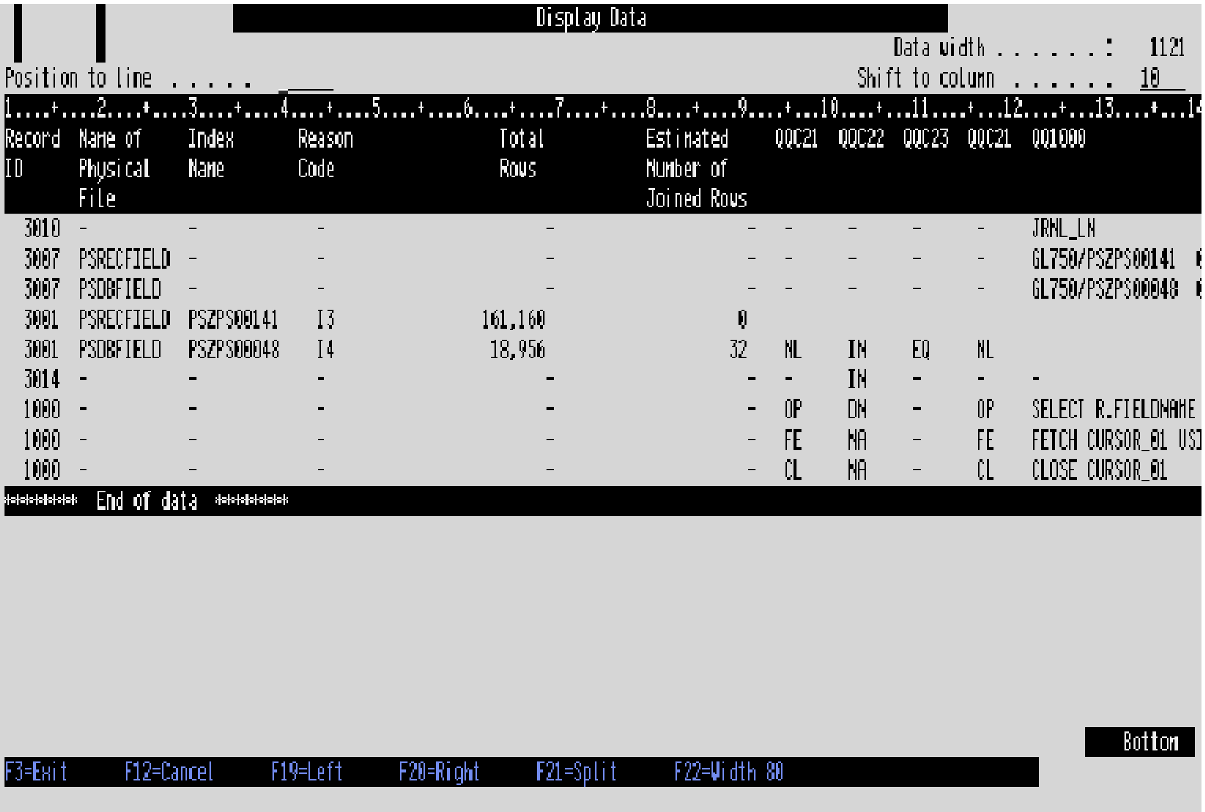
Here are some examples of the screen captures:

3001 Record Data



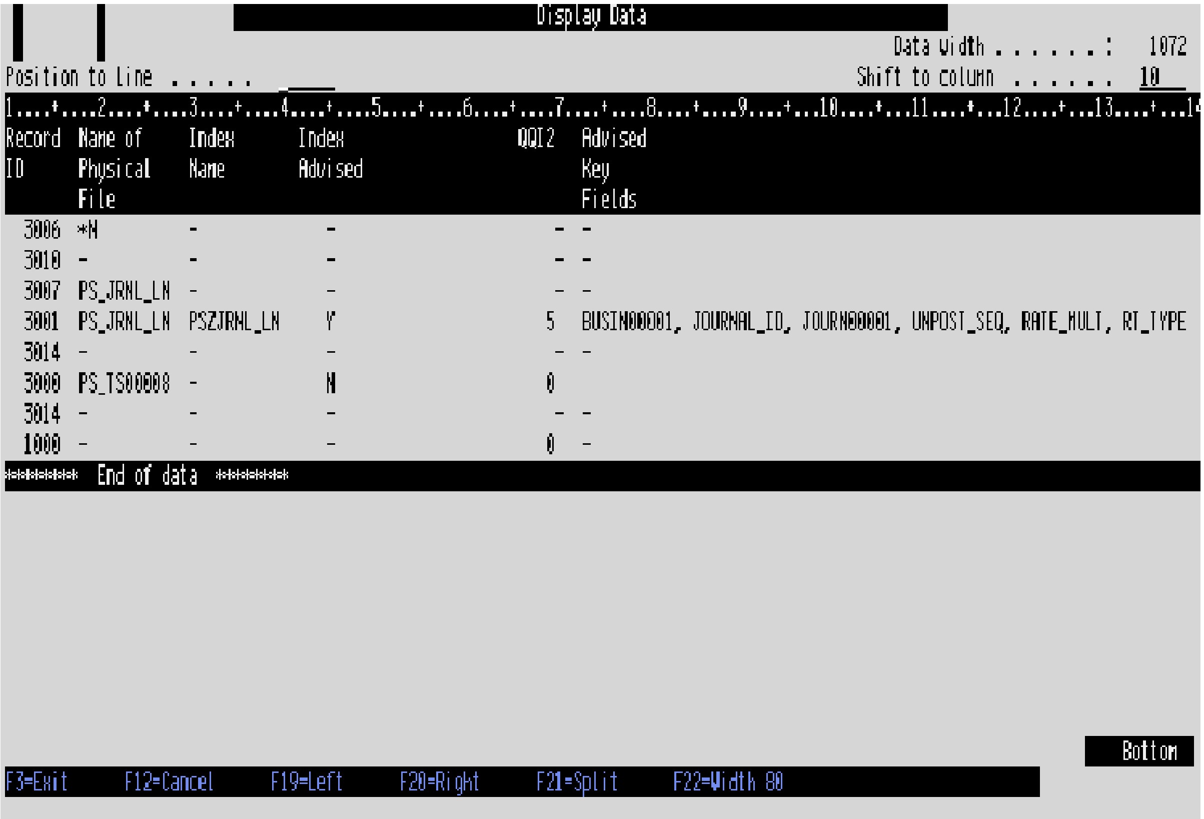
```
                                        Display Data
                                                                   Data width . . . . . . :   1049
Position to line  . . . . .  _____                               Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record  Name of     Index       Reason  QQ1000
ID      Physical    Name        Code
        File
  3010  -           -             -     PTPRUNID, U, UPDID
  3001  PS_SQ00001  PSZSQ00001    I1
  3014  -           -             -     -
  1000  -           -             -     SELECT STMT_TEXT FROM GL750/PS_SQLSTMT_TBL WHERE PGM_NAME = ? AND STMT_TYPE = ? AND  STMT_N
  1000  -           -             -     FETCH CS0 INTO : SQLSTMT-TBL . SQLSTMT-TEXT
  1000  -           -             -     CLOSE CS0
  3010  -           -             -     JRNL_LN
  3007  PSRECFIELD  -             -     GL750/PSZPS00141  0, GL750/PSEPS00003  6, GL750/PSDPS00004  6, GL750/PSBPS00031  6, GL750/F
  3007  PSDBFIELD   -             -     GL750/PSZPS00048  0, GL750/PSAPS00012  4
  3001  PSRECFIELD  PSZPS00141    I3
  3001  PSDBFIELD   PSZPS00048    I4
  3014  -           -             -     -
  1000  -           -             -     SELECT R.FIELDNAME ,R.FIELDNUM ,F.LENGTH ,F.FIELDTYPE ,F.DECIMALPOS ,R.USEEDIT ,R.EDITTABLE
  1000  -           -             -     FETCH CURSOR_01 USING DESCRIPTOR : SQLDA-SELECT-01
  1000  -           -             -     CLOSE CURSOR_01
******** End of data ********
                                                                                          Bottom
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

3001 Records with Join Data



```
                                   Display Data
                                                              Data width . . . . . . :    1121
Position to line  . . . . .  _____                            Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  Name of     Index       Reason                 Total           Estimated     QQC21  QQC22  QQC23  QQC21  QQ1000
ID      Physical    Name        Code                   Rows            Number of
        File                                                           Joined Rows
  3010  -           -             -                        -                    -   -      -      -      -     JRNL_LN
  3007  PSRECFIELD  -             -                        -                    -   -      -      -      -     GL750/PSZPS00141
  3007  PSDBFIELD   -             -                        -                    -   -      -      -      -     GL750/PSZPS00048
  3001  PSRECFIELD  PSZPS00141    I3                 161,160                    0
  3001  PSDBFIELD   PSZPS00048    I4                  18,956                   32   NL     IN     EQ     NL
  3014  -           -             -                        -                    -   -      IN     -      -     -
  1000  -           -             -                        -                    -   OP     DN     -      OP    SELECT R.FIELDNAME
  1000  -           -             -                        -                    -   FE     NA     -      FE    FETCH CURSOR_01 USI
  1000  -           -             -                        -                    -   CL     NA     -      CL    CLOSE CURSOR_01
********  End of data  ********
                                                                                                                  Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

3001 Record with Index Advisor



```
                                   Display Data
                                                              Data width . . . . . . :    1072
Position to line  . . . . .  _____                            Shift to column  . . . . . .  10
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...1
Record  Name of     Index       Index                  QQI2  Advised
ID      Physical    Name        Advised                      Key
        File                                                 Fields
  3006  *N          -             -                       -  -
  3010  -           -             -                       -  -
  3007  PS_JRNL_LN  -             -                       -  -
  3001  PS_JRNL_LN  PSZJRNL_LN    Y                       5  BUSIN00001, JOURNAL_ID, JOURN00001, UNPOST_SEQ, RATE_MULT, RT_TYPE
  3014  -           -             -                       -  -
  3000  PS_TS00008  -             N                       0
  3014  -           -             -                       -  -
  1000  -           -             -                       0  -
********  End of data  ********
                                                                                                                  Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

3003 Record — Query Sort

The 3003 record shows that the database optimizer has decided to put selected records into a temporary space and sort them. This is either cheaper than alternative indexed methods or it is forced to do so — for example: UNION or order by on fields from more than one file.

Indexes can still be used to select or join records before the sort occurs. This does NOT indicate that the ODP is non-reusable.

The *1000* SQL summary record for the open may have a high elapsed time (QQI6 or QQI4). Sort buffers are refilled and sorted at open time, even in reusable ODP mode. However, high elapsed times may indicate a large answer set. In this case, the sort will outperform index usage (This is the situation in most cases).

If sort seems slow and using an index might be better, try to influence the optimizer away from the sort with better selection indexes. For example, if the answer set is small but the optimizer does not have the right indexes available to know that, creating these indexes can help. This is possible only if the optimizer is not forced to use the sort.

Most commonly used fields:

QQSTIM — Timestamp for start of refill and sort

QQETIM — Timestamp for end of refill and sort

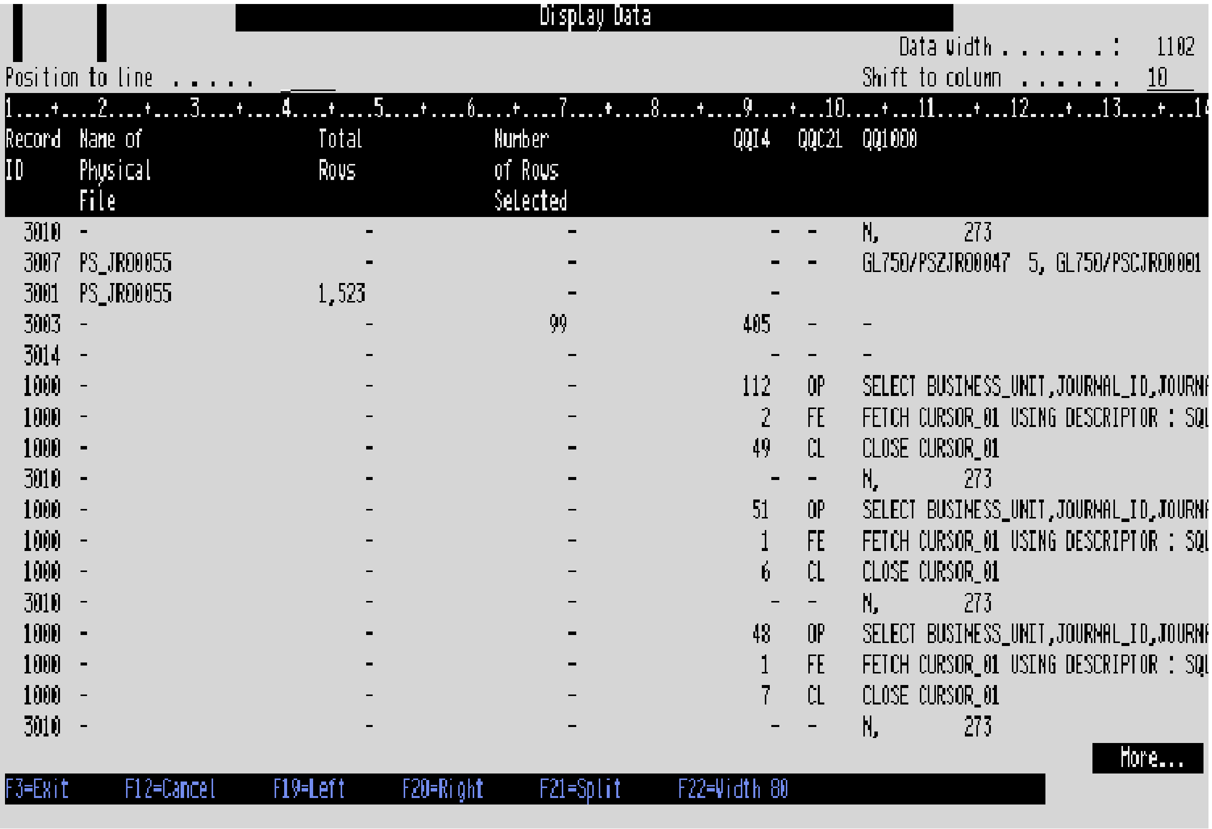
QQRCOD — Reason for choosing query sort

> The QQRCOD field helps to determine whether a sort was required or if it was “costed” this way. Refer to the DB monitor guide for the reason codes.

QQRSS — Number of rows in sort space

> The QQRSS field can be used, along with reason code, to determine if the indexed approach is possible and possibly cheaper (for a small result set). Use the QQI3 value from the corresponding *1000* FE record for this open to determine how many rows were fetched from the sort space. If the QQRSS value is large but the actual number of rows fetched is small, consider adding OPTIMIZE FOR n ROWS to the query to help the optimizer.

3003 Record — Query Sort



```
                                Display Data
                                                         Data width . . . . . . :   1102
Position to line  . . . . .  _____                   Shift to column  . . . . . .  10
1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...10...+...11...+...12...+...13...+...1
Record Name of              Total           Number          QQI4  QQC21 QQ1000
ID     Physical             Rows            of Rows
       File                                 Selected
  3010 -                       -               -              -    -    N,        273
  3007 PS_JR00055              -               -              -    -    GL750/PSZJR00047  5, GL750/PSCJR00001
  3001 PS_JR00055          1,523               -              -
  3003 -                       -              99            405    -    -
  3014 -                       -               -              -    -    -
  1000 -                       -               -            112   OP    SELECT BUSINESS_UNIT,JOURNAL_ID,JOURNA
  1000 -                       -               -              2   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQL
  1000 -                       -               -             49   CL    CLOSE CURSOR_01
  3010 -                       -               -              -    -    N,        273
  1000 -                       -               -             51   OP    SELECT BUSINESS_UNIT,JOURNAL_ID,JOURNA
  1000 -                       -               -              1   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQL
  1000 -                       -               -              6   CL    CLOSE CURSOR_01
  3010 -                       -               -              -    -    N,        273
  1000 -                       -               -             48   OP    SELECT BUSINESS_UNIT,JOURNAL_ID,JOURNA
  1000 -                       -               -              1   FE    FETCH CURSOR_01 USING DESCRIPTOR : SQL
  1000 -                       -               -              7   CL    CLOSE CURSOR_01
  3010 -                       -               -              -    -    N,        273
                                                                                               More...
F3=Exit      F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

3004 Record — Temporary File

The 3004 field show that the database optimizer is forced to store intermediate results/rows in a temporary file due to the nature of the query. Examples are: group by on fields from >1 file or materializing view results. This indicates ODP is non-reusable. This cannot be tuned — consider altering the query.

Most commonly used fields:

QQSTIM — Timestamp for start of fill temporary results table

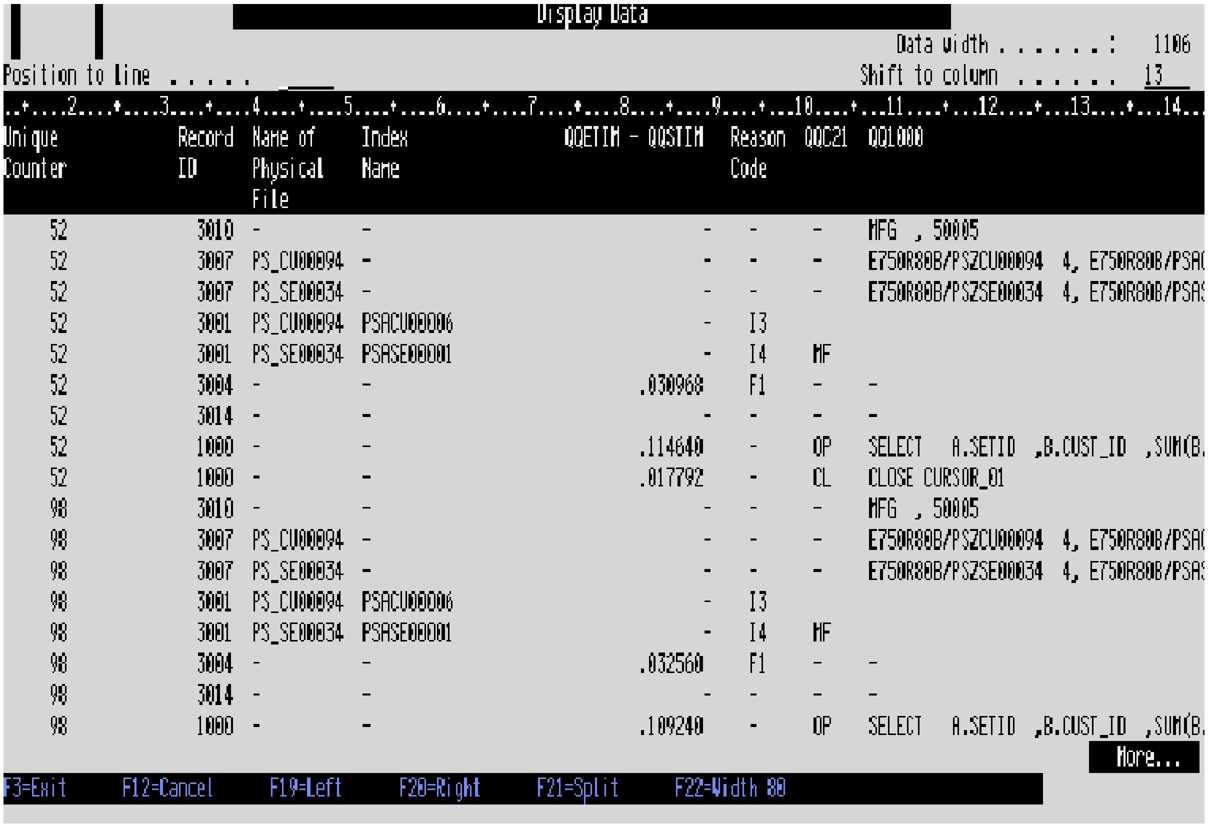
QQETIM — Timestamp for end of fill temporary results table

QQTMPR — Number of rows in temporary table

QQRCOD — Reason for building temporary

Refer to the DB monitor guide for the specific reason codes.

3004 Record — Temporary Results File



```
                                        Display Data
                                                                      Data width . . . . . . :   1106
Position to line  . . . . .  _____                                  Shift to column  . . . . . .  13___
..+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14...
Unique          Record Name of    Index               QQETIM - QQSTIM  Reason QQC21  QQ1000
Counter         ID     Physical   Name                                 Code
                       File
     52           3010 -          -                                  -     -      -    MFG  , 50005
     52           3007 PS_CU00094 -                                  -     -      -    E750R80B/PSZCU00094  4, E750R80B/PSAC
     52           3007 PS_SE00034 -                                  -     -      -    E750R80B/PSZSE00034  4, E750R80B/PSAS
     52           3001 PS_CU00094 PSACU00006                         -     I3
     52           3001 PS_SE00034 PSASE00001                         -     I4     MF
     52           3004 -          -                            .030968     F1     -    -
     52           3014 -          -                                  -     -      -    -
     52           1000 -          -                            .114640     -      OP   SELECT   A.SETID  ,B.CUST_ID  ,SUM(B.
     52           1000 -          -                            .017792     -      CL   CLOSE CURSOR_01
     98           3010 -          -                                  -     -      -    MFG  , 50005
     98           3007 PS_CU00094 -                                  -     -      -    E750R80B/PSZCU00094  4, E750R80B/PSAC
     98           3007 PS_SE00034 -                                  -     -      -    E750R80B/PSZSE00034  4, E750R80B/PSAS
     98           3001 PS_CU00094 PSACU00006                         -     I3
     98           3001 PS_SE00034 PSASE00001                         -     I4     MF
     98           3004 -          -                            .032560     F1     -    -
     98           3014 -          -                                  -     -      -    -
     98           1000 -          -                            .109240     -      OP   SELECT   A.SETID  ,B.CUST_ID  ,SUM(B.
                                                                                                                    More...
F3=Exit     F12=Cancel     F19=Left     F20=Right     F21=Split     F22=Width 80
```

3006 Record — Access Plan Rebuild

The QQRCOD field lists the reason the rebuild of the plan is occurring. It is not present on every full open. It only occurs in cases where the access plan already exists but for some reason it must change. This field can help determine the reason for a full open if other DB monitor records don't show why.

Other DB Monitor Records

The following records are not used as much for performance tuning, but can provide other interesting data.

3014 record — Generic query information

> The 3014 record appears with full open optimization records. In most cases, one *3014* record appears per full open. Multiple *3014* records can appear if the query consists of multiple separately run queries. For example, subqueries with grouping functions or views that need results materialized for use in outer query. Values in this field help identify what type of query this record represents and how long it took to open the cursor for this query. Refer to DB monitor for other field values. See the example on the next page.

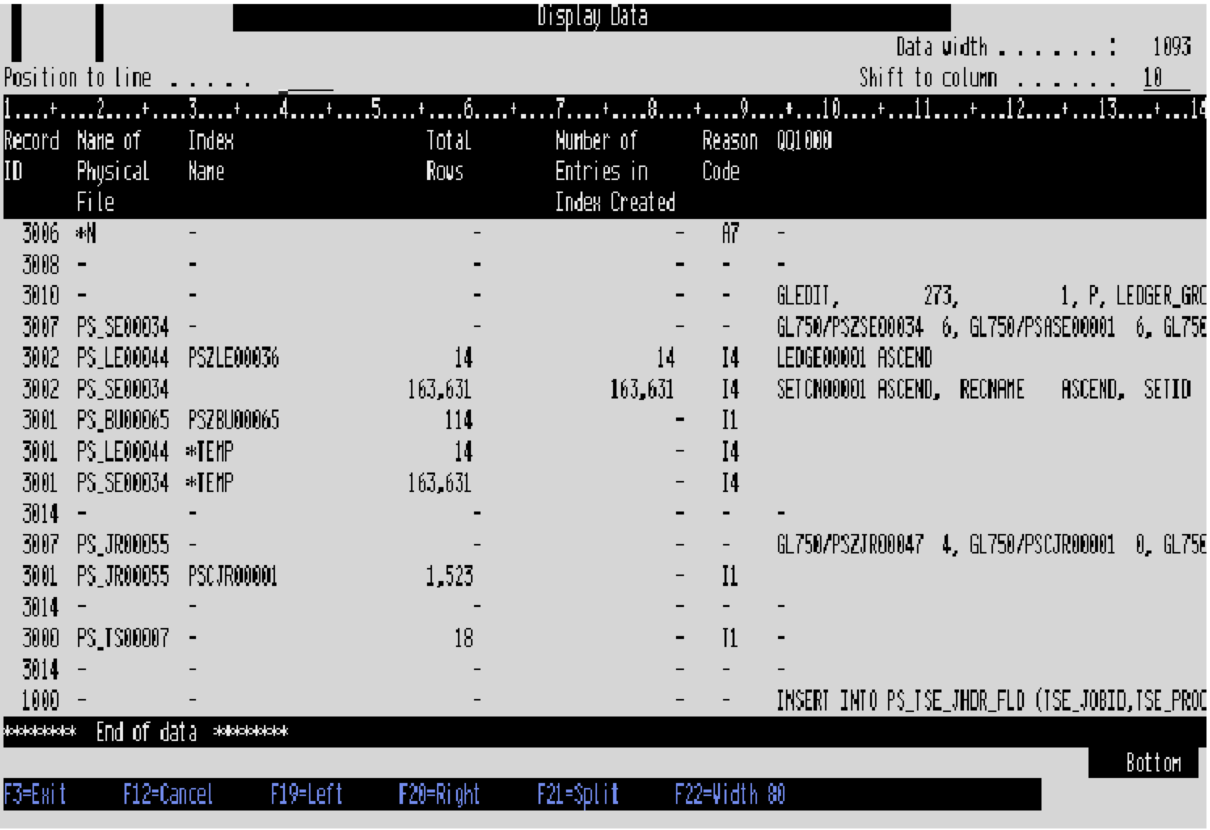
3005 record — Table locked

3008 record — Subquery processing

3018 record — STRDBMON/ENDDBMON information/option

3019 record — Records retrieved detail record (only occurs with TYPE(*DETAIL)

3014 Records



```
                                    Display Data
                                                                 Data width . . . . . . :   1093
Position to line  . . . . .  _____                              Shift to column  . . . . . .  10___
1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+...10....+...11....+...12....+...13....+...14
Record  Name of     Index                  Total           Number of       Reason  QQ1000
ID      Physical    Name                   Rows            Entries in      Code
        File                                               Index Created
  3006  *N          -                          -                   -       A7      -
  3008  -           -                          -                   -       -       -
  3010  -           -                          -                   -       -       GLEDIT,         273,          1, P, LEDGER_GRC
  3007  PS_SE00034  -                          -                   -       -       GL750/PSZSE00034  6, GL750/PSASE00001  6, GL75E
  3002  PS_LE00044  PSZLE00036                14                  14       I4      LEDGE00001 ASCEND
  3002  PS_SE00034                       163,631             163,631       I4      SETCN00001 ASCEND,  RECNAME    ASCEND,  SETID
  3001  PS_BU00065  PSZBU00065               114                   -       I1
  3001  PS_LE00044  *TEMP                     14                   -       I4
  3001  PS_SE00034  *TEMP                163,631                   -       I4
  3014  -           -                          -                   -       -       -
  3007  PS_JR00055  -                          -                   -       -       GL750/PSZJR00047  4, GL750/PSCJR00001  0, GL75E
  3001  PS_JR00055  PSCJR00001             1,523                   -       I1
  3014  -           -                          -                   -       -       -
  3000  PS_TS00007  -                         18                   -       I1      -
  3014  -           -                          -                   -       -       -
  1000  -           -                          -                   -       -       INSERT INTO PS_TSE_JHDR_FLD (TSE_JOBID,TSE_PROC
******** End of data ********
                                                                                                      Bottom
F3=Exit      F12=Cancel      F19=Left      F20=Right      F21=Split      F22=Width 80
```

Database Performance Analysis

Database performance problems that are not easily avoidable

Problems that are generally not simple to rectify include the following:

Full opens due to repetitive query using new literal values each time it's invoked:

If the prepared statement text differs in any way from any previous SQL statement in that job, a full open and new ODP will be needed. SQL will, in most cases, attempt to convert literals to parameter markers to make the repetitive statement appear identical each time. However, literals cannot be converted in the following cases:

- Parameter marker conversion turned off,
- Original SQL statement contains both parameter markers and literals,
- Statement uses special registers (CURRENT DATE, CURRENT TIME), or
- Expressions used in SET or SELECT clause of statement.

Non-reusable ODPs due to:

- Temporary results table created for ODP (3004 record appears in the full open) and/or
- Group by fields on more than one file or view with grouping (materialization).

Although the full opens listed here are not avoidable, they can still possibly be improved. For example, an open might contain a *3004* record but might also contain a costly temporary index build that can be avoided by building a new permanent index. Look at the opens to see what can be done.

Time-consuming queries due to:

- Correlated subqueries run excessive number of times within a query,
- NOT EXISTS forcing poor join order,
- Updates or deletes with poor join performance because the file being changed is forced to be first in join, and/or
- Long running index builds with mapped keys in the build (the *3002* record has *MAP in QQ1000).

Problems that are generally avoidable

Problems that are relatively easy to resolve include the following:

Temporary indexes created for join/order by/group by:

- Individual long-running or repeated index creates (non-reusable ODPs) and/or
- A *3002* record exists in the full open (without *MAP in QQ1000 field).

Time-consuming queries due to:

- Table scans — single long-running or repetitious short-running

  A *3000* record exists in the full open (unless the open is done for a file that is being inserted into).
- Poor join order

  Need to be familiar with what a better join order would be and how to influence the optimizer.
- Temporary sort being done when index approach feasible and faster

  A *3003* record exists in the full open and possibly longer opens in reusable ODP mode.
- Reusable ODPs with a good initial access plan that become worse due to table growth

  Watch the QQI6 or the QQI4 value in the *1000* records to see if the value grows. Try to influence the optimizer to an initial access plan that is not affected by table growth.

Identifying and Tuning Problem Areas

There are many different methods to identify problems and tune the troublesome database statements. One of the most common methods is to identify the most dominating, time-consuming queries and work on each of them individually. Another method is to leverage global information and to use this information to look for indexes that are "begging" to be created.

A set of queries have been included in a following section. These queries have been designed to help produce useful results in most situations. Using these queries will help to understand the data and to learn to construct other queries as well. Query analysis is iterative in nature so try something, run job (with the DB monitor active) or individual query (STRSQL using debug messages) to see if it worked — try again if it did not work.

When using the STRSQL command, ensure that the appropriate settings are used for the “Data refresh” and “Allow copy data” options. These settings can be changed by using the Change Session Attributes option in STRSQL. For example, JDBC uses *FORWARD for Data refresh and *OPTIMIZE for Allow copy data. The defaults are *ALWAYS for Data refresh and *YES for Allow copy data.

It is usually best to first concentrate on repetitious non-reusable ODPs, table scans, and long index builds. Also, look for repetitious short-running queries that are not optimized well. Joins and sorts can be more difficult to analyze. If joins and sorts are accounting for a significant portion of run time, they need to be addressed as well. Fine tuning smaller problems should be done after large problems have been addressed. Generally, indexes will be used to tune most performance problem areas.

Creating Indexes

It is helpful to know how data in the table is populated and how selective certain key fields are. This information can be used to help create indexes that will be used for a large number of queries. It can also help to know why existing indexes were not used in some situations. For example, if a query has WHERE A = ? AND B = ? AND C = ? and there is an index over A, B and C but the database optimizer decides not to use it, it may be because these fields are not very selective. Knowing the data can help to quickly detect this. If the selective of certain fields are not known, the fields can be queried to find out:

```
            SELECT A, B, C, COUNT(*)
FROM TABLEA GROUP BY A, B, C
ORDER BY 4 DESC
```

Try to create indexes that are used more globally. Use selective fields that are commonly used in WHEREs, and where applicable, use them in combination with common join, order by, and group by keys. Remember the tips discussed in the query optimization section on creating these. If index is uniquely keyed, create the index as UNIQUE since this is useful to the database optimizer.

Do not create a lot of permanent indexes trying to cover every combination. Create one or two that are potentially good, run the job again, or run the STRSQL command for a single query and see if they are used. If the indexes used and the run time is noticeably better, consider deploying the indexes for permanent use. If the indexes are not used, delete them and try a different combination. Do not create indexes just to solve a single instance of a full open or query unless it accounts for a significant amount of time.

Each additional index created for a table will cause overhead when:

- Updates to the table include the index keys,
- Rows are inserted or deleted for the table, and/or
- Full opens occur for that file (index evaluation).

For Faster Analysis

The DB monitor file often is large and contains information on many jobs; therefore, running queries on the data can sometimes be slower than desired. Users can try to help this by collecting only the job they want. However, sometimes this is not possible and, even if it is, batch jobs can generate a lot of DB monitor output. Also, using interactive tools such as STRSQL, can result in longer run times on server models. If the response time is slow during the analysis, consider the following tips:

- Create a smaller DB monitor file with only the records you are interested in
  - Build and run a query to pick out only those jobs you're interested in, then copy the records for those job(s) into a separate DB monitor file.
  - CRTDUPOBJ and INSERT w/subselect specifying the QQJNUM value(s)
- Create indexes on the DB monitor file over the common selection and grouping/order by clauses.
  - Examples of some key combinations to use:
    - QQJNUM,
    - QQRID,
    - QQUCNT,
    - QQRID & QQ1000, and
    - QQRID & QQC21.
  - Try other combinations as needed. Remember to combine the selection and grouping/order by clauses.

Altering Insert with Subselect Data in Monitor File

Within the DB monitor file, inserts with subselects actually contain two QQUCNTs. The first one is listed for subselect optimization messages during full open; the second one is for the actual insert statement and each instance of the reusable ODP after that.

This can be a problem when trying to look at the optimization messages for the subselect using the QQUCNT record for the insert operation itself. There is not an easy way to view this. The problem is being addressed, but for now use the following method to correct this problem.

In STRSQL, change the SELECT output value in the session attributes (use PF13, then select option 1) to a value of 3 (“output to file”). For this example, let's choose the file name of FILE01 in QGPL.

Run the following query:

```
      SELECT QQUCNT-1
FROM monitor-file-name
WHERE QQC21 = 'IN' AND QQ1000 NOT LIKE '%VALUES%'
AND QQI5 = 0 AND QQJNUM = job-nbr
```

Output from this query will go to FILE01 in QGPL. Now, change the session attribute back to “display” and enter the following query:

```
      UPDATE monitor-file-name
SET QQUCNT = QQUCNT + 1
WHERE QQUCNT IN(SELECT DISTINCT SEL0001 FROM QGPL/FILE01)
AND QQJNUM = job-nbr
```

DB Monitor Query Examples #1 — #13

DB Monitor Query #1 — *Identify the specific job(s) to analyze.*

```
      SELECT SUM(QQI4), COUNT(*), QQJOB, QQUSER, QQJNUM
FROM DBMON/SQARUNS2

            WHERE QQRID = 1000
            GROUP BY QQJOB, QQUSER, QQJNUM ORDER BY 1 DESC
```

From this list, pick out jobs that use the most run time. Use the QQJNUM value as selection in other analysis steps. If only one job is monitored or one job completely dominates the others, QQJNUM is not needed. If the job cannot be identified from this information, then work with the customer or use other tools, like WRKSYSACT, to determine the job.

DB Monitor Query #2 — *Which type of SQL operations account for the most run time?*

```
      SELECT SUM(QQI4), COUNT(*), QQC21
FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 1000
GROUP BY QQC21 ORDER BY 1 DESC
```

This query shows the number of SQL operations (OP, CL, FE, UP, CM...) used and how much elapsed time is spent by each type of operation. The results from this query can help to know what to concentrate on for the most potential payback (selects, updates, etc.). Total run time for all SQL operations within the job can be obtained by running the above query, only specifying SUM(QQI4) in the select list, and dropping the group by and order by clauses. This value can be useful when trying to determine how much of the job's total run time a given query or set of queries accounts for.

DB Monitor Query #3 — *Which SQL operations account for the most run time (list by text)?*

```
      SELECT SUM(QQI4), COUNT(*), QQ1000
FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 1000
GROUP BY QQ1000 ORDER BY 1 DESC
```

This query shows the text of the SQL operations that account for the most run time, and how many times they were run. Consider using QQC21 or QQUCNT <> 0 to remove the non-ODP operations (prepares, describes, etc.). This query can help to become familiar with particular queries that are using the most run time. Fetches and closes are not correlated to the opens. Also, consider sorting by QQ1000 to group similar statements together. Another method would be to add QQC21 to the select and group by clause and then order by QQC21, 1 DESC (groups opens, inserts, etc. together and by cost).

DB Monitor Query #4 — *Which queries account for the most run time (list by QQUCNT)?*

```
      SELECT SUM(QQI4), COUNT(*), QQUCNT
FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 1000
AND QQUCNT <> 0
GROUP BY QQUCNT ORDER BY 1 DESC
```

This query shows individual query instances and how much time they took, sorted by the run time (largest to smallest) and includes fetch and close time for opens. It does not include non-ODP SQL operations such as prepare or describe (QQUCNT=0). This provides a way to quickly find a query that is taking a large amount of time without having to know the text or anything else about that query. Consider limiting QQC21 values to look at certain query type. If there is not a single dominating query or set of queries, and if DB Monitor Query #3 does not provide the information that is needed, go to DB Monitor Query #5.

DB Monitor Query #5 — *Which queries account for the most run time (using QQUCNT and text)?*

```
      SELECT SUM(QQI4), COUNT(*), QQUCNT, QQ1000
FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 1000 AND QQUCNT <> 0
GROUP BY QQ1000,QQUCNT ORDER BY QQUCNT, 1 DESC
```

The results of this query show all parts of a query grouped together and listed by QQUCNT. Fetches and closes are listed together with their corresponding opens. It does not include non-ODP operations (prepare, describe, commit, etc.). Scroll through the unique queries to see the most expensive ones and use the QQUCNT record to further analyze them (see DB Monitor Query #6 below). The query can be changed to sort by text and run time (4, 1 DESC) to group similar queries together and see which are the most costly, or just by run time to see the most costly overall. It may also be useful to add QQC21 values to select a certain query type.

DB Monitor Query #6 — *How do I determine how an individual query was run?*

```
      SELECT QQRID, QQPTFN, QQIFNM, QQC21, QQI4, QQ1000
FROM DBMON01
WHERE QQJNUM = '195030' AND QQUCNT = query-number
```

The fields used in this query are a good starting point, but there are other fields that can help to determine what the query is doing and the access plan that was used. For example, to find how long an index build within this query is taking, replace QQI4 with QQETIM-QQSTIM. If full open data was collected, optimization records will appear first followed by the *1000* record(s), then repeated instances of query if the ODP was reusable. The text of the query along with the host variable values from the *3010* record can be used to reconstruct the query for debug purposes.

DB Monitor Query #7 — *Which queries are significantly affected by full opens?*

```
      SELECT SUM(QQI4), COUNT(*), QQ1000
FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 1000 AND QQI5 = 0
AND QQC21 IN ('OP', 'DL', 'IN', 'UP')
GROUP BY QQ1000 ORDER BY 1 DESC
```

This query points out queries whose run time is noticeably affected by full opens. Add ORDER BY QQ1000 to see if there are similar queries encountering full opens. Use DB Monitor Query #8 below to find any individual queries that are doing full opens.

DB Monitor Query #8 — *List all queries (sorted by text) that are doing full opens.*

```
      SELECT QQI4, QQUCNT, QQ1000
FROM DBMON/DBMON01
```

```
WHERE QQJNUM = '195030' AND QQRID = 1000 AND QQI5 = 0
AND QQC21 IN ('OP', 'DL', 'IN', 'UP')
ORDER BY 3, 1 DESC
```

This query groups together queries that are doing full opens. Scroll through the list to find repetitive queries, pick QQUCNT from one of these, and use it to look at the query to understand why it is doing a full open. Remember to look closely at the text. If it is a "new" SQL statement, the open cannot be avoided since SQL must create an ODP the first time. Look at certain query types by limiting the QQC21 list and sort by QQI4 to see the most costly opens. Remember that the first full open for each unique query is unavoidable, although it may be possible to improve them in some cases.

DB Monitor Query #9 — *List all queries with full opens that contain a temporary index build***.**

```
      SELECT QQI4, QQUCNT, QQ1000
FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 1000 AND QQI5 = 0
AND QQC21 IN ('OP', 'DL', 'IN', 'UP')
AND QQUCNT IN(SELECT DISTINCT QQUCNT FROM DBMON01
WHERE QQJNUM = '195030' AND QQRID = 3002 AND QQ1000 NOT LIKE '%*MAP%')
ORDER BY 3, 1 DESC
```

This query helps to quickly locate queries that are doing temporary index builds. It provides the QQUCNT values that can be used to look at the query optimization to find more information. First, sort by QQI4 first to find the most expensive opens. This does not necessarily mean that the open can be avoided. There may be other reasons the full open is occurring besides the temporary index build.

DB Monitor Query #10 — *Which index builds are done the most often?*

```
      SELECT QQUCNT, QQETIM-QQSTIM, QQPTFN, QQTOTR, QQRIDX,
            QQRCOD, QQIDXA, SUBSTR(QQ1000, 1, 100),
            SUBSTR(QQIDXD, 1, 100)
      FROM  DBMON01 WHERE QQRID = 3002 ORDER BY 8
```

This query points out commonly occurring index creates and how to create permanent indexes to avoid them. Look for repeated index builds or long index builds first. If an index build only occurs once and is not costly, it may be best to let it occur. The SUBSTR for QQ1000 and QQIDXD should cover most key lists, but it may be necessary to increase the respective values if 100 bytes is not enough.

Use the following methods to determine what keys to use to build a permanent index:

- If QQTOTR = QQRIDX, the keys from QQ1000 for the index build will probably be acceptable.
- If QQTOTR > QQRIDX, additional selection keys should be in the index along with the join, order by, or group by keys from the QQ1000 field. The fields from QQIDXD can be used, if available. However, it may be desirable to add QQI2 to the selection to know the number of primary keys.
- If there is no data in QQIDXD or the keys from QQIDXD do not seem to work, get the most selective keys from the query text itself — use QQUCNT to find the query text.

DB Monitor Query #11 — *Which full opens are not avoidable due to temporary table results?*

```
      SELECT QQUCNT, QQC21, QQI4, QQ1000
FROM DBMON/OM75DBMON
WHERE QQJNUM = '195030' AND QQRID = 1000 AND QQI5 = 0
AND QQC21 IN ('OP', 'IN', 'UP', 'DL') AND QQUCNT IN
SELECT DISTINCT QQUCNT FROM DBMON/OM75DBMON
WHERE QQJNUM = '195030' AND QQRID = 3004)
ORDER BY 4, 3 DESC
```

This query shows all full opens that are unavoidable due to temporary results table and groups them by text and run time. Often it is useful to sum on QQI4 and group on QQ1000 to determine the overall cost. It still may be possible to improve the cost of the open. Use QQUCNT to see if the query can be optimized better or if other tunable items, such as index build, exist.

DB Monitor Query #12 — *Which queries involve arrival sequence?*

```
      SELECT SUM(QQI4), COUNT(*), QQUCNT, QQ1000
FROM DBMON01 WHERE QQJNUM = '195030' AND QQRID = 1000
AND QQUCNT IN (SELECT DISTINCT QQUCNT
FROM DBMON01 WHERE QQRID = 3000 AND QQJNUM = '195030')
GROUP BY QQUCNT,QQ1000 ORDER BY 1 DESC
```

This query lists the most expensive SQL operations that are using arrival sequence. The presence of a *3000* record does not necessarily cause bad performance (inserts, bit maps, and scans of small files). Use the QQUCNT record to look at the most expensive operations to see how much the file using arrival sequence is affecting the query. Look for fetches that are costly per fetch since this can suggest that a table scan is being done on a large file — prime candidate for creating a new index. It may be useful to order by QQUCNT since this lists the specific operations (e.g. fetch) with the corresponding open/close.

DB Monitor Query #13 — *Which queries involve use of a query sort?*

```
      SELECT SUM(QQI4), COUNT(*), QQUCNT, QQ1000
FROM DBMON01 WHERE QQJNUM = '195030' AND QQRID = 1000
AND QQUCNT IN (SELECT DISTINCT QQUCNT FROM DBMON01
WHERE QQRID = 3003 AND QQJNUM = '195030')
GROUP BY QQUCNT,QQ1000 ORDER BY 1 DESC
```

This query lists the most expensive SQL queries that are using a query sort. The presence of a *3003* record does not necessarily indicate poor performance. Use QQUCNT to look at the most expensive opens (selects). Look for opens (selects) that are costly per open. A sort may be required — see the QQRCOD field in the *3003* record. If it is not required, determine if an index would help improve this. It may be useful to order by QQUCNT since this lists the specific operations (e.g. fetch) with the corresponding open/close.

End User Query Tools

Once a basic understanding of the queries that can be used to query the Database Monitor output is gained, the next step is to become efficient at using the queries. A variety of query tools exist that can help access the data from the Database Monitor:

- Operations Navigator: Run SQL Script
- Interactive SQL
- Query/400
- DB2 UDB for AS/400 Query Manager

Of the query tools listed above, DB2 UDB for AS/400 Query Manager (LLP 5769-ST1) is the most flexible for composing, executing and managing reports from queries. The DB2 UDB for AS/400 Query Manager allows user variables in queries so that items like the database filename can be selected when the query is run, instead of when the query is written. DB2 UDB for AS/400 Query Manager Report Forms make formatting output of queries easier by generating column headings, spacing, and field wrapping for more readable end-used reports.

The DB2 UDB for AS/400 Query Manager user guide (SC41-5212-01) can be found on the web at: http://publib.boulder.ibm.com/pubs/html/as400/v4r4/ic2924/info/db2/rbao1mst.pdf

Tools That Can Help

Centerfield Technology has a product, Database Essentials, that can help you with both the analysis and collection of database performance monitor data on AS/400. The advanced toolset from Centerfield Technology features a visual explain toll, graphical formatting, and pre-canned analytical reports that make the database performance monitor data easier to understand and analyze. In addition, Database Essentials provides several easy-to-use graphical wizards and advisors that automate several database performance tuning tasks such as index tuning. For additional information on Centerfield Technology and their Database Essentials product, see their Web site at: http://www.centerfieldtechnology.com

Summary

Tuning SQL and a database structure can be a very demanding exercise. Performance analysis of database problems can be difficult and time-consuming. Performance tuning, particularly when dealing with database operations, is an iterative process but the availability and knowledge of powerful tools allow the performance analyst to narrow-in on a solution much faster.

Knowledge and judicious usage of the OS/400 Database Monitor tool and the detailed queries provided in this document allow the analyst to gain more information about how the application SQL statements are performing and what can be done to correct the problem. The time it takes for the performance analyst to understand the database problems and solve them can be reduced significantly by using this tool.

Additional Information and Author Contacts

Special thanks goes to Dale Weber for the usage of his SQL class foils used to develop this white paper.

IBM Global Services in Rochester, MN, offer technical support for Performance Analysis & Capacity Planning of AS/400 products. Their services are described at: http://www.as400.ibm.com/service/igs/pss.htm

Additional information regarding AS/400 and DB2 UDB for AS/400 and the Database Performance Monitor is available on the AS/400 home page beginning at: http://www.as400.ibm.com/db2

Questions regarding the conclusion reached in this white paper should be addressed to Rick Peterson, rickmp@us.ibm.com , or Richard Odell, rjodell@us.ibm.com .

Trademarks

The following terms are trademarks of the IBM Corporation in the United Stated or other countries or both:

AS/400	IBM
Operating System/400	OS/400
WebSphere	400
DB2 UDB for AS/400	

This publication may have referred to products that are not available in your country.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc., in the United Stated, other countries, or both.

Microsoft, Windows, Windows NT, and Windows logos are trademarks of Microsoft Corporation in the United Stated, other countries, or both.

Intel, Pentium, Xerox are trademarks of Intel Corporation in the United States, other countries, or both.

IBM makes no commitment to make available any products referred to herein.

All other trademarks and registered trademarks are the properties of their respective companies.