SQL

Monitoring

Facility

User's Guide

**Software
Product
Research**

SQL/Monitoring Facility
Release 7.5

1 Functional description

SQL Monitoring Facility ("**SQL/MF**") is a real-time SQL execution monitor for DB2/VM (formerly called SQL/DS).

It provides database administrators with a window into running DB2/VM systems. The facility permanently records the global execution status of the database systems and maintains execution-time information on all active SQL statements, users and programs.

SQL/MF provides following classes of monitoring:

System Monitoring

> The System Monitor records the **global system status** of all monitored DB2/VM databases.

Statement Monitoring

> The Statement Monitor maintains the **execution characteristics for each SQL statement** running in the monitored databases.

Statement Statistics

> For each statement in each executed DB2/VM package, the Statement Monitor records its **resource usage and access path** in a DB2/VM table.

Package statistics

> For each executed DB2/VM **package**, the Statement Monitor records its **resource usage** in a DB2/VM table.

Statement Logging

> An SQL statement is stored in the Monitor Log table, when it exceeds the resource consumption limits, defined by the installation in the Monitor configuration dataset.

Statement Benchmarking

> When a package has been designated for benchmarking, all its SQL statements with their resource usage and access path, are stored in the Monitor Benchmark Log table.

Statement Recording

> When recording has been enabled, all SQL statements executed during the recording period are stored, along with their execution characteristics into a **recorder** file, which is a CMS dataset.

1.1   System Monitoring

The SQL/MF System Monitor runs in a dedicated virtual machine. The facility continually monitors the DB2/VM environment at an installation defined sampling interval and saves the resulting data into the System Monitor tables.

The monitor has two monitoring processes: the **main monitoring process** monitors the entire DB2/VM environment; the **buffer monitoring process** monitors activity within the DB2/VM buffer pool and maintains buffer access distribution information. A separate sampling interval can be provided for each process. A sampling is also performed when DB2/VM initiates a database checkpoint.

The facility is able to monitor up to 255 database machines, 255 storage pools per database and 4096 DBspaces concurrently.

At each monitor sample and for each database monitored, data is obtained and saved into the session datasets. The cumulative counters obtained from DB2/VM are stored as *interval related* values, that is, as indicators of DB2/VM resource consumption during the monitoring interval. Following data are obtained:

General DB2/VM Performance Data

SQL/MF obtains the DB2/VM "COUNTER" data from the DB2 control blocks and computes the resource usage during the interval. Following items are added to the regular DB2/VM statistics:

- the page and directory buffer hit ratios
- the average duration of Block I/O requests
- the average duration of dataspace fault processing
- the average agent dispatch delay (monitors the effect of changes to the DB2 initialization parameter DISPBIAS)
- the VM performance indicators: global CPU utilization, expansion and paging rate

Data Space Monitoring

If the DB2/VM Data Spaces feature has been installed, the performance of each pool will be monitored at each interval. The Monitor stores the number of dataspace buffer lookups, dataspace reads and writes, the target and the current working set and the save interval. Using these data, the dataspace hit ratio (dataspace_faults / (dataspace_reads+dataspace_writes)) and the dataspace access hit ratio (dataspace_faults / buffer_lookups) are computed. The dataspace hit ratio show the efficacy of paging within the dataspace.

Buffer Pool Monitoring

SQL/MF records the usage that individual DBSPACES and DBextents are making of the DB2/VM buffer pool and provides information on the *distribution* of database accesses among individual DBspaces and storage pools. These data are obtained at the "buffer monitoring" interval and written to the session datasets when the "main monitoring" interval expires. Since monitoring very large buffer pools may induce additional overhead when inspecting the DB2/VM buffer control tables, the facility must be specifically enabled.

DB2 Package Cache Monitoring

For each monitor interval, following data are recorded:

- the number of entries defined in the package cache
- the number of package cache entries in use
- the number of package cache entries in use for active agents
- the total number of packages monitored thus far

Dbspace Usage Monitoring

If requested in the configuration file, each DBSPACE in each database is monitored for physical DASD space usage within its header, data and index pages. The related SHOW DBSPACE statements are performed at the day and the time specified in the same configuration file.

Storage Pool Monitoring

At each interval, physical DASD space usage is monitored for each pool and eventual short on storage conditions are recorded.

DB2/VM LOG Usage

At each interval, SQL/MF saves the Logfile space consumption during that interval.

User Activity

At each interval, SQL/MF logs the names and the wait state of all active users.

DB2/VM Lock Monitoring

At each interval, SQL/MF obtains detailed information about users in LOCK wait state and provides information about the locks being held.

Connection Monitoring

At each interval, data is obtained about the state of the DB2/VM connections and agents.

Checkpoint Monitoring

For each monitor interval, following data are recorded, if appropriate:

- the number of checkpoints
- the number of checkpoint delays
- the average duration of a checkpoint
- the average duration of a checkpoint delay

Checkpoint Delay Monitoring

At each interval, SQL/MF checks whether the checkpoint agent waits on termination of long-running DBSS calls and inhibits new LUWs in order to initiate the checkpoint. The list of users active at this time is included in the report.

DB2/VM Session Performance

The DB2/VM performance statistics are kept on an daily basis and retained for a specified number of days. The System Monitor also keeps a **Session** table containing the DB2 counters, summarized per database and per day. This table is never purged. It allows to perform system monitoring on a

long-term basis.

1.2   Statement Monitoring

The Statement Monitor runs partly in the virtual machine of the DB2/VM database server and partly in a dedicated service machine. These components continuously record the execution characteristics of each SQL statement, both dynamic and static (compiled). For each statement in execution, the Statement Monitor maintains following data:

- the DB2/VM agent and userid executing the statement
- the name of the DB2/VM package containing the statement
- the section number of the statement within the package
- the statement type (prepare, open, fetch ...)
- the text of the statement (also for compiled statements)
- the contents of the host variables used by the statement
- the isolation level (repeatable read, uncommitted read or cursor stability)
- the program's blocking attribute
- the statement start date and start time
- the statement end date and end time
- the statement elapsed time
- the name of the DBspace(s) and table(s) being accessed by the statement
- the name of the index(es) being used, if any
- the selectivity of index access
- the CPUtime used by the statement
- the total time spent by the statement in lock wait
- the total time spent in I/O wait
- the total time spent in communications wait
- the number of RDS calls executed by the statement
- the number of DBSS calls executed
- the number of lock escalations during the statement
- the number of lock requests resulting in wait
- the number of deadlocks during the statement
- the number of rows processed by the statement
- the number of buffer lookups performed by the statement
- the number of data pages read and written
- the number of directory buffer lookups
- the number of directory blocks read and written
- the number of log pages read and written
- the total number of I/O requests issued
- the number of dataspace pagefaults (DB2/VM dataspaces only)
- the number of VM BLOCKIO requests (DB2/VM dataspaces only)
- the number of buffer lookups in internal DBspaces
- the number of calls to the DB2/VM dispatcher

The above statistics are kept at 3 levels:

- the **statement** level
- the **section** level (for cursor-based statements)
- the **package** level

The monitor keeps these data in a VM/ESA **shared segment**, which is shared by all databases subject to monitoring and by those SQL/MF components that exploit the monitor's data. The shared segment can accommodate 128 databases, which is the maximum number of databases that can be handled by the Statement Monitor on a single VM/ESA system.

The monitored data in the shared segment are **volatile**: they exist only while statements are executing and they are cleared when the statement completes.

1.3   Statement Statistics

For each section in each DB2/VM package, the monitor automatically maintains the following data in the **SQLCS_SQL_STMNTS** table:

- the name of the database
- the name of the DB2/VM package containing the statement
- the section number of the statement
- the plan number of the access within a section, if the section accesses more than one table or if the statement participates in a referential constraint
- the text of the statement[1]
- the name of all DBspaces, tables and indexes accessed by the statement
- the selectivity of the indexes used
- the statement's execution frequency (times sampled)
- the program's isolation level
- the program's blocking attribute
- the last or total[2] resource usage of the statement:
  - the CPUtime used
  - the total time spent in lock wait
  - the total time spent in I/O wait
  - the total time spent in communications wait
  - the number of RDS calls executed
  - the number of DBSS calls executed
  - the number of lock escalations during the statement
  - the number of lock requests resulting in wait
  - the number of deadlocks during the statement
  - the number of rows processed by the statement
  - the number of buffer lookups by the statement
  - the number of data pages read
  - the number of data pages written
  - the number of directory buffer lookups
  - the number of directory blocks read
  - the number of directory blocks written
  - the number of log pages read
  - the number of log pages written
  - the total number of I/O requests issued
  - the number of dataspace pagefaults, if the DB2/VM dataspace feature has been installed
  - the number of VM BLOCKIO requests, if the DB2/VM dataspace feature has been installed
  - the number of buffer lookups in internal DBspaces
  - the number of calls to the DB2/VM dispatcher

---

[1]The text of the package statements is initially inserted during installation of the Monitor. Subsequent package prepping is sensed by the Monitor and the modified statement text is inserted at the first execution of the modified package. The same applies to new packages.

[2]Depending on the SECTION_STATS statement in the SQLCS CONFIG file.

Notes

(1) The initial SQL/MF product installation process creates the SQL_STMNTS table and populates it with the text of all existing packages, by unloading them into the table. This inserts the statement text of all SQL statements.

(2) The SQL statement text in the SQL_STMNTS table will be automatically updated during DB2/VM functions that modify the statement text, that is, during package prepping and reload.

(3) A "section" represents a single SQL statement, or a group of related cursor-based statements. Moreover, SQL/MF provides a **package initialization** section (numbered -1) to record the processing cost associated with package loading, run privilege checking and eventual auto-reprepping.

(4) If requested, the monitor will maintain the above data for each **modification level** of the program. A new modification level is automatically built when the package is prepped or reloaded. The latest modification level is numbered 0, levels -1, -2 etc. indicate the previous modification levels. The monitor configuration file allows to specify the number of package modification levels to keep. Using the package modification levels, it is possible to monitor the changes in access path and resource usage that result from package modifications.

(5) For performance reasons, the statement statistics are accumulated in memory and stored in the DB2/VM table at regular time intervals, as defined in the STAT_FREQ parameter of the monitor configuration file.

1.4   Package statistics

When a package terminates[3], the Statement Monitor performs *end-of-package logging*. This consists in storing the accumulated statistics for all statements in the package into the Statement Monitor Log table. The end-of-package statistics are maintained in the Log table as one row per day, per user and per program. The package statistics include following data:

- the package execution frequency
- the CPUtime used by the program
- the total time spent in lock wait
- the total time spent in I/O wait
- the total time spent in communications wait
- the number of RDS calls executed
- the number of DBSS calls executed
- the number of lock escalations during the program
- the number of lock requests resulting in wait
- the number of deadlocks during the program
- the number of buffer lookups by the program
- the number of data pages read
- the number of data pages written
- the number of directory buffer lookups
- the number of directory blocks read
- the number of directory blocks written
- the number of log pages read
- the number of log pages written
- the total number of I/O requests issued
- the number of dataspace pagefaults, if the DB2/VM dataspace feature has been installed
- the number of VM BLOCKIO requests, if the DB2/VM dataspace feature has been installed
- the number of buffer lookups in internal DBspaces
- the number of calls to the DB2/VM dispatcher

For performance reasons, the statement statistics are accumulated in memory and stored in the DB2/VM table at regular time intervals, as defined in the STAT_FREQ parameter of the monitor configuration file. The package statistics provide extensive information that can be used for accounting purposes. The statistics can be disabled by specifying the NOPSTATS option during startup of the SQLCS program, as described on page 90.

---

[3]Package termination is sensed when a COMMIT or ROLLBACK statement is issued or when the application is terminated by DB2/VM.

1.5   Additional Monitor Services

In addition to its basic monitoring function, the Statement Monitor provides a number of **monitoring services**, which must be requested by coding appropriate control statements in the Statement Monitor **configuration file**.

1.5.1 Statement Logging

If requested, the Monitor records SQL statements in its exception Log table under the following conditions:

- the statement exceeds a resource consumption limit defined as
    - a maximum number of DB2/VM buffer lookups
    - a maximum number of I/O requests
    - a maximum response (wall-clock) time
    - a maximum lock wait time
- the statement executes using DBSPACE scan
- the statement executes with the repeatable read isolation level
- the statement is in lockwait during a specified period of time
- the agent is idle during a specified period of time
- the agent causes one or more log escalates during its execution
- the agent has been in checkpoint wait during its execution
- the statement ends with an SQLCODE
- the statement executes without blocking
- the statement executes in dynamic mode (QMF, ISQL or DBSU statements for example)

This type of logging is called *exception logging.* Within a single execution, a statement may be logged multiple times, for each exception caused.

Exception logging occurs at the end of the SQL statement or package section. If the package section logged contains multiple SQL statements (such as cursor-based statements), the statistical counters logged represent **package section** values, that is, the accumulated statistics of all preceding statements in the section. For logging purposes, a sequence of **fetch** and/or **put** statements is handled as a single statement and its log entry contains the total consumption by all preceding fetch/put statements.

To avoid excessive logging activity for a given DB2/VM server, following actions are taken:

- As a rule, a static SQL statement (package section) is logged once per day, per user and per logging reason.
- For BUFLOOK, RESPTIME, LUW_TIME and TOTIO exceptions, the monitor keeps the **highest** exception value for a given statement during the session, when LOG HIGHEST ON has been stated in SQLCS CONFIG. Otherwise, all exceptions are logged chronologically.
- The SQLCODE exception is logged unconditionally, but once per LUW only. Continuous and consecutive logging of the same SQLCODE from the same package is also avoided, even when issued from different LUW's.
- Prepping a package clears its "already-logged" status.

1.5.2 Benchmarking

DB2/VM users or programs can be defined as subject to benchmarking. **All** SQL statements executed by these users or programs are automatically stored into the Monitor Log table. The log then provides a detailed description of the 'behaviour' of these programs. This type of logging is called *benchmark logging.* The monitor distinguishes between **specific** (BENCH PROGRAM XYZ) and **generic** benchmarking (BENCH PROGRAM *).

To avoid excessive benchmarking activity in a given DB2/VM server, following actions are taken for **specific benchmark** requests

- A given SQL statement (compiled or dynamic) is benchmarked only once during a DB2/VM session, unless the application is reprepped.
- Benchmarking of a given package can be re-enabled by the MONITOR BENCH statement (see page 200.)
- Dynamic monitor configuration (MONITOR CONFIGURE statement) resets the benchmarking facility to its initial status.

For **generic benchmark** requests, it is the user's responsibility to control the system overhead caused by benchmarking.

When a package eligible for benchmarking calls other DB2/VM packages, the SQL statements in the called packages will be benchmarked as well.

If cursor-based statements are recorded in the benchmark table, the following rules apply:

- for all cursor statements except close, the **statement** resource consumption is stored
- for a cursor CLOSE, the **section** resource consumption is stored, that is, the consumption between PREPARE (if a dynamic statement) or OPEN (for static statements) and CLOSE
- if multiple fetches are executed on the cursor, only the first FETCH statement is recorded

NOTE
Application benchmarking is also possible, with less overhead, using the recording facility, more specifically using the RECORDER START statement (see page 213).

1.5.3 Application Statement Recording

Application Statement Recording is a sophisticated tracing facility that notes the execution characteristics of all executing SQL statements into a CMS **recorder** file. In order to achieve acceptable performance, recording is done into a dataspace. An asynchronously executing component stores the recorded data from the dataspace into the CMS recorder file. When VM/ESA Version 2 Release 1 or higher has been installed, the VM/ESA **Data Compression** facility will be invoked to reduce the disk space required for recording. The Statement Recording facility is described on page 225.

Recording can be enabled on a chronological basis using the RECORDER FROM - TO command or "on demand" using the RECORDER START command (see page 213).

1.5.4 Lockwait Recording

Lockwait Recording is part of the Statement Recording facility. If requested, it registers all lockwait events in the recorder file. The lockwait record stores the name of the locked and locking user and program, the name of the wanted DBSPACE and the characteristics of the lock.

As the facility registers all lockwaits in chronological order, it may help in understanding and correcting locking problems.

1.5.5 Application Tracing

Application tracing is an alternative for application benchmarking and recording. It is primarily intended for application debugging purposes. While benchmarking records the execution of a given statement only once, tracing maintains a complete record of all SQL calls executed by a designated user or program. After execution of the traced application, the trace data are formatted into an SQLTRACE report, which is a CMS file created on the A-disk of the user requesting the trace. For a complete description of the MONITOR TRACE command, please refer to page 215.

1.5.6 Automated Analysis

- When an SQL statement is stored in the Monitor Exception Log, the Analysis option will forward the statement to a designated virtual machine, executing the **SQL/Command Analysis** program product[4] in server mode. The resulting analysis report is routed automatically to a designated VM userid.
- SQL/MF offers a facility that automatically and preventively analyzes packages when it is found, during package prep or reload, that one or more package statements exceed a defined cost and may affect performance at execution time. Analysis is performed by the SQL/Command Analysis server. The analysis report is returned to a designated VM userid or to the user that initiated the prep.
- SQL/MF provides the utilities SQLCSXPL and SQLCSXSA to analyze applications, based on their execution statistics in the monitor tables.

---

[4]SQL/Command Analysis is a tool offered by Software Product Research.

1.5.7 Notification

The notification facility sends a message to a designated VM user (the database administrator for example) when:

- an SQL statement exceeds a resource consumption limit defined as
  - a maximum number of DB2/VM buffer lookups
  - a maximum number of I/O requests
  - a maximum SQL cost, if a dynamic statement
- an SQL statement executes with the repeatable read isolation level
- an SQL statement exceeds a defined response-time
- an SQL session (LUW) is idle for a defined period
- an SQL statement is in the lockwait state for a defined period
- a dynamic statement exceeds a defined SQL cost

Notification does not occur more than once in a given logical unit of work.

Messages issued by notification during a database session are recorded in a CMS dataset, called "database" SQLCSLOG. This dataset is automatically sent to a named user at the end of each day.

1.5.8 Governor

The Governor facility controls the database resource consumption of users and programs. The facility restricts users and programs by issuing a DB2/VM **force** command, whenever a **restricting condition** is encountered. A restricting condition is defined as the maximum amount of system resources that a user or a package section is allowed to consume. Restrictions can be defined in terms of:

- a maximum number of DB2/VM buffer lookups
- a maximum number of I/O requests
- a maximum statement response time
- a maximum lockhold time
- a maximum idle time, while in LUW
- a maximum number of writes to the DB2 log

INCLUDE and EXCLUDE statements define the users and programs, subjected to restricting. The restriction facility can be enabled for all users and all programs, both static and dynamic.

1.5.9 AutoRebind Facility

With the AUTO_REBIND option stated in the SQL/MF configuration file, all CREATE INDEX statements will be recorded and a **package rebind** will be requested (at the end of the monitor session) for all packages that depend on the tables for which a new index has been created. The purpose of this facility is to ensure that existing packages benefit from the newly created indexes.

1.5.10   AutoPrep Facility

The AutoPrep facility reduces the cost of dynamic SQL statement execution. An installation should consider using the facility, when intensive dynamic SQL is performed. PC database access, e-business or ERP applications often execute in dynamic SQL mode only, thus causing a considerable CPU overhead and catalog contention. The AutoPrep facility is described on page 251 of this manual.

1.5.11   Dynamic alias

When compiled DB2 applications wish to execute a statement in dynamic mode, table privileges are required for all users of the application, which is often undesirable. The dynamic alias facility provides a solution. See page 87.

1.5.12   CONNECT Screening

DB2/VM calls a user exit (**ARIUXIT**) whenever a user connects to the database server. However, ARIUXIT must be written in Assembler and be linked with the DB2/VM code whenever created or modified. Moreover, since the exit is called by the DB2/VM Resource Adapter at the client side of the connection, the exit is not fully available in the DRDA environment.

The SQL/Monitoring Facility provides an easier-to-use alternative for ARIUXIT. When a user performs an *explicit* connect (that is, when a package issues the DB2/VM CONNECT statement), SQL/MF invokes the **SQLCSUXT EXEC**, if such file is found on one of the minidisks or directories accessed by the DB2/VM server. Contrarily to ARIUXIT, SQLCSUXT is written in REXX and does not require a DB2/VM relink. SQLCSUXT receives the name of the database and the CONNECT userids as input arguments. The exit can request to cancel the CONNECT by passing a non-zero returncode. A sample SQLCSUXT is delivered with SQL/MF. For details on coding a CONNECT exit, see the chapter "Customizing SQL/MF" on page 279.

1.6   User Interface

The SQL/MF User Interface allows database administrators to examine the data recorded during monitoring. A single, multi-functional program (**SQLCSU**) provides access to all the monitored data.

Using the interface, following data can be obtained:

- The list of **running SQL statements** with their resource consumption, in textual or graphical format. Each running statement can be inspected in detail, and several monitor functions can be invoked for the statement, such as:
  - analyzing the statement (if our SQL/CA program product is in the system)
  - forcing the statement
  - obtaining lock information
  - showing the DB2/VM catalog information for the objects used by the program
- A graphical report of **SQL activity in the current DB2/VM session**, summarized per user, per package or per package section and ordered by descending resource consumption.
- Reports of **global DB2/VM system activity**, in both textual and graphical format.
- Reports of SQL statements **logged** due to their resource consumption.
- Reports of SQL statements **benchmarked** on user request.
- Reports of SQL statements **recorded** on user request.
- Reports of **total resource consumption** by individual users and programs in the current DB2/VM session.
- Reports showing the **resource consumption, the statement text and the access path** for each SQL statement in each DB2/VM program.

1.6.1 User Interface to the Statement Monitor

The **SQLCSU** program allows a database administrator to look into the monitor blocks for the **running** SQL statements. The program provides a continuous display of all statements executing in all monitored databases, with a summary of their characteristics.

```
        SQL/Statement Monitor                        © Software Product Research
-------------------------------------------------------------------------------
Database  VM_name  SQL_name  Started   Elapsed   Package  Type     Stat   SQL_Cost
-------------------------------------------------------------------------------
SQLD2     VSESP    985       10:41:23 00:02:17 DF252     Fetch    Comm        10
TSTADB    CMS14    SQLDBA    10:43:39 00:00:01 CSTEST    Open     DSPF        27




-------------------------------------------------------------------------------
PF  1 Help      2 Resume     3 Quit      4 Detail     5 SQLgraph    6 SQLpulse
    7           8            9 RunStats  10 LogTable  11 SysMon     12 HostCmnd
```

The user may interrupt the display, indicate a particular statement with the cursor and request one of the following functions:

Statement detail

Provides **all** the monitor data available for the statement. In addition, the full text of the statement is shown, with the host variables replaced with their actual contents. For compiled SQL statements, the statement text is obtained from the SQLCS_SQL_STMNTS table. The function also shows the name of the primary index used during execution, unless the statement is executed by DBSPACE scan.

```
 Database TSTADB     VM_name  TSTAST1    SQL_name TSTAST1    Location T25TT262
 Agent   3           Status   Running    Package  SQLCVRFY   Section  1
 Command Fetch       Isolatn  CS         LUW_id   8172387    Protocol DB2/VM
-------------------------------------------------------------------------------
 Started  18:46:39   Curtime  18:46:59   Elapsed  00.00.20   Rowcount 32123
 CPUtime  2.914      Locktime 0.000      IOtime   0.061      Commtime 1.494
 RDScall  354        DBSScall 32123      DispCall 713        Buflooks 32953
 Pagread  831        Pagwrite 0          Logread  0          Logwrite 0
 Dirbufl  787        Dirread  0          Dirwrite 0          Int_DBSP 0
 DSfaults 10         *BlockIO 0          Nr_Locks 10         Escalate 0
-------------------------------------------------------------------------------
 SELECT ITEM,"SQL/DS HELP" FROM SQLDBA.SYSTEXT2



-------------------------------------------------------------------------------
 No index used during access

-------------------------------------------------------------------------------
PF  1 Help      2 Resume     3 Quit      4 Analyze    5 ProgStat   6 LockInfo
    7           8            9           10 CmndText  11 Force      12 ObjList
```

Statement analysis

The statement can be forwarded to the SQL/Command Analysis program product (provided it has been installed on the system) and the analysis report examined interactively.

Lock analysis

When an agent is in the lock wait state, the function shows all information needed to determine the nature of the lock and the name of the lock owner (the so-called DB2/VM *lock graph*). When the agent is not in lockwait, the function shows the locks held by the agent, by issuing the SHOW LOCK USER command. In both cases, the function shows the names of the DBSPACES appearing (by their number) in the DB2/VM command output.

Object Information

Object information shows DB2/VM system catalog information for the DBSPACE, the table, the index used in the current statement and for all indexes created on the table.

Statement force

A designated SQL statement (agent) can be terminated (forced) by the user interface.

Host Commands

Any DB2/VM, CP or CMS command may be executed from the user interface in any of the monitored databases, thus providing a single point-of-control for all monitored databases.

The MONITOR command can be used to suspend or resume monitoring, to start application tracing, to force DB2/VM users online / offline and to reconfigure the monitor without shutdown. (See page 200 for details).

1.6.2 Graphical User Interface to the Statement Monitor

The **SQLPULSE** function, which is invoked from the SQLCSU program, provides a graphical representation of DB2/VM activity by all running statements in all monitored databases. The program shows a bar graph for all running statements with one the following DB2/VM counters:

- number of buffer lookups performed by the statement
- total number of I/O requests performed by the statement
- CPU seconds used by the statement

```
           Tot I/O  ........1K........2K........3K........4K........5K........6K
SQLDBA   ARIISQL    ████████
SQLOPL   MK30A      █████████
SQLOPL   ARIDSQL    ████
TSTADB   SQLCVRFY   ██████████████████
```
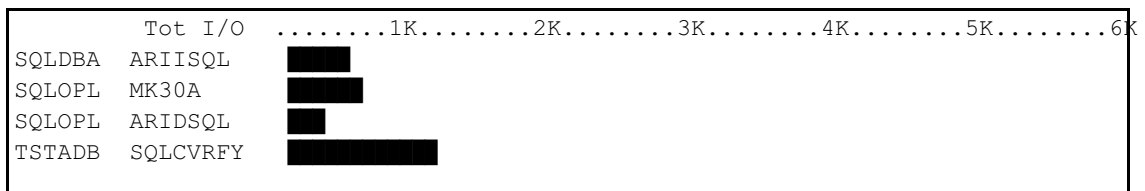
The SQLPULSE **Detail** function provides a graphical representation of resource usage by the SQL statement, pointed to by the cursor. The function shows for the current DB2/VM package section:

- the databasename, userid's, package name and package section nr
- the statement start and elapsed time
- the text of the SQL statement being executed (for both static and dynamic statements)
- the statement's access path (name of the index, if any)
- a graphical representation of the execution statistics

See page 132 for full details.

```
Database  TSTADB           SQL_id   TSTAST1       VM_id   TSTAST1
ProgName  SQLCVRFY         Section  1             Command Fetch
Starttime 15:28:14         Elapsed  00.00.09      Status  Running
Access    No index access noted
SQL_Cmnd  SELECT ITEM,"SQL/DS HELP" FROM SQLDBA.SYSTEXT2
-----------------------------------------------------------------------
CPU       ██  6446
IOwait    █ 97
COMMwait  █ 1740
LOCKwait
NrRows    ████████████████████  51233
RDScall   █ 565
DBSScall  ████████████████████  51296
DISPcall  █ 1139
Buflook   ██████████████████████████████████  53848
Read IO   ████████  1298
Write IO  █ 3
Int DBSP
```

1.6.3 The RunStats Interface

In each database server, SQL/MF keeps the total daily consumption for each package section, that is, for each SQL statement executed. Contrarily to the other statement statistics, which are kept in a table, these statistics always remain in storage for rapid access.

Following counters are kept:

- the total number of buffer lookups performed by the statement
- the total number of I/O's performed by the statement
- the total CPU time consumed by the statement

The **RunStats** function, which is invoked from the SQLCSU program, allows to examine these statistics in an hierarchical manner.

Initially the RunStats utility shows the resource consumption summarized by DB2/VM userid, but summarizing by programname can be requested.

On all RunStats reports, the **Detail** function is used to explore the consumption by a given user or program.

- In a user list, **detail** shows the programs executed by a designated user.
- In a program list, **detail** shows all executed sections (statements) of a designated program.
- In a program section list, **detail** shows the complete statistics, the statement text (if a prepped statement) and the index used for the designated section.

All the above lists are ordered by descending resource consumption. As a default, ordering is by buffer-lookups. But ordering by CPU usage or by I/O load can be achieved using the PFkey interface.

For a detailed description of the RunStats program, please refer to page 127.

```
Page 1.1        SQL/Session Run Statistics per Program      05/13/96 14:07:12
-------------------------------------------------------------------------------
Data = BUFLOOK  .......10K.......20K.......30K.......40K.......50K.......60K
SQLCVRFY        ███████████████████████████████████████     56736
SQLCAXC2        █    178
SQLCLEXP        █    109




-------------------------------------------------------------------------------
PF 1 Help       2 Numerics  3 Quit      4 Detail    5 Users     6
   7 Page <<    8 Page >>   9 CPU      10          11 Tot/IO   12 Totals
```

1.6.4 Saving the RunStats

By coding the RETAIN_RUNSTATS option in SQLCS CONFIG and by creating the saved Runstats table, session runstats can be retained for a user specified number of days. The Runstats user interface is used to display the saved runstats by period.

The table can be consulted using the regular RunStats program, described on page 127.

1.6.5 User Interface to the Statement Monitor Log

The **LogTable** function of the **SQLCSU** program provides access to the Statement Monitor tables. The program is used to consult:

-    the exception log entries
-    the benchmark log entries
-    the statement statistics
-    the package statistics

When invoked, the program displays a **menu** of Monitor reports. From the menu, the user selects the report class (exception log, statement statistics ..) and the particular report within that class. The individual reports show the selected table data:

-    in chronological order
-    by username
-    by package name
-    by index usage
-    by I/O consumption
-    by CPU consumption
-    by lockwait time
-    by logging reason
-    ordered by user defined criteria etc..

All the above reports are generated using a monitor table editor, that comes with the product.

All the reports can be requested for the current day or for a specified chronological range. The user may add his own SELECT clauses to the standard search conditions.

Moreover, the reporting interface has been designed in such a manner, that an installation can easily add its own log reports to the menu.

The following screen shows an entry from the SQL/MF Exception Log table.

```
 SQL/MF Report LCHRONO  System Demo                              Page 65 of 355
 ------------------------------------------------------------------------------
 DATABASE  SQLOPL          USERNAME  VSESP41          SQLID  CICSUSER
    AGENT  7                BEGDATE  1995-01-09      ENDDATE  1995-01-09
  BEGTIME  10.33.40         ENDTIME  10.33.40        ELAPSED  00.00.00
 PROGNAME  BCP007           PROGCREA  BK             SECTION  2
COMM_TYPE  AUXCALL         ISOLATION  RR             BLOCKED  Y
  CPUTIME  3               LOCKTIME  0                IOWAIT  0
 COMMWAIT  1                BUFLOOK  1                 NROWS  1
  RDSCALL  1               DBSSCALL  1              DISPCALL  1
 ESCALATE  0               WAITLOCK  0              DEADLOCK  0
  PAGREAD  0               PAGWRITE  0               DIRBUFL  1
  DIRREAD  0               DIRWRITE  0               LOGREAD  0
 LOGWRITE  0                  TOTIO  0               DSFAULT  0
  BLOCKIO  0               INT_DBSP  0              LOGSTAMP  AC3ECFEB47A90
LOGREASON  DBSPSCAN
   ACCESS  No index access noted
 ------------------------------------------------------------------------------
SELECT STACKDATA, SUBSTR(STACKDATA,10,8),SUBSTR(STACKDATA,24,7),SUBSTR(STACKDATA
,18,6),SUBSTR(STACKDATA,81,1),SUBSTR(STACKDATA,84,7),SUBSTR(STACKDATA,82,2)
FROM BK.STACKSORT WHERE STACKID = 17241990 ORDER BY 2 ASC,3 ASC,4 ASC,5 ASC,6
ASC,7 ASC



 ------------------------------------------------------------------------------
 PF  1 Help      2 Objects   3 Quit      4 Analyze   5 Format    6 Top
     7 Page <<<  8 Page >>>  9 Bottom   10 Search   11 Hardcopy 12 ListMode
```

1.6.6 User Interface to the System Monitor Tables

The **System Monitor** function of SQLCSU provides authorized users with interactive access to the system monitor tables. A menu oriented interface provides access to the following online reports:

-   Session Summary Report
-   SQL COUNTER Reports
    -   Logical Unit of Work counter report
    -   IO counter report
    -   Lock Request Block counter report
    -   VM counter report
    -   Dataspace Counter report
    -   Counter Summary report
-   DBspace Buffer Usage Report
-   Storage Pool Buffer Usage Report
-   DBspace Usage Report
-   Storage Pool Usage Report
-   DB2/VM Log Usage Report
-   User Status Report
-   Lock Contention Report
-   Checkpoint Delay Report
-   Connections Report

Each of the above reports may be requested as:

-   a **detail** report providing data for each monitoring interval,
-   a **totals** report providing summary values for the monitoring session
-   an **averages** reports providing average values for the monitoring session

All the reports are displayed using a monitor table editor, that comes with the product. The following is an example of the **System Counter Summary** report.

```
MONTIME  MONDATE          RDSC       DBSSC       LUWS      TOTIO PAGEHIT
---------------------------------------------------------------------------
17.50.09 1995-08-26       2589       234124         4          0      98
17.50.53 1995-08-26       1708       152202         7          2      98
21.06.25 1995-08-26       2703       238469        16         16      98
21.06.56 1995-08-26       1590       137798        14         14      98
21.07.34 1995-08-26         54          159         6          2      91




---------------------------------------------------------------------------
PF  1 Help      2 Menu      3 Quit      4 CmndLog   5 Format    6 Top
    7 Page <<<  8 Page >>>  9 Bottom   10 View <<< 11 View >>> 12 PageMode
```

A number of control functions is provided to:

- modify the monitoring interval
- print the monitor tables for the current session
- shutdown the system monitor
- temporarily suspend monitoring
- resume suspended monitoring
- execute a DB2/VM operator command (without the need for an agent structure)

1.6.7 Graphical User Interface to the System Monitor

The **SQLGRAPH** function of the SQLCSU program, provides a graphical image of DB2/VM activity in a given database by showing a bar graph for the resource items LUWs, RDScalls, BUFLOOKs and TOTIO. The program can also provide information about the current use of the DB2/VM buffer pool.

```
                                                                   Database
                                                                    SQLDBA


      87890                                                           TOTIO
      81140
      74390
      67640
      60890
      54140
      47390
      40640
      33890
      27140
      20390
      13640
       6890
            06:00-----08:30-----11:00-----13:30-----16:00-----18:30-----
```
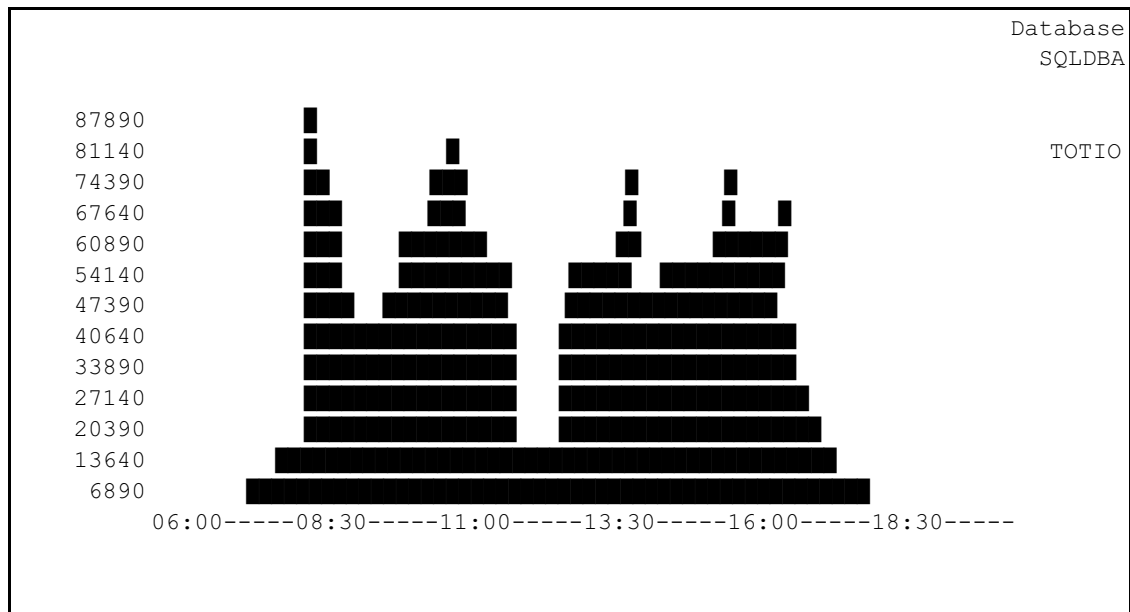
1.6.8 Statement Analysis using the Monitor Tables

If the SQL/Command Analysis program product has been installed on your system, the statement execution statistics recorded by the monitor in the statement statistics and exception tables, can be used as a basis for program analysis.

The SQLCSXPL and SQLCSXA programs acceptsa search clause from the user and applies it to the SQLCS_SQL_STMNTS or SQLCS_LOG tables to extract all statements satisfying the user's clause. These statements are stored in a CMS file, which is subsequently submitted to the SQL/Command Analysis product.

SQLCSXPL and SQLCSXA allow to analyze statements based on their effective execution characteristics such as: the number of I/O's performed, the number of rows fetched, the indexname used and so on. Every column of the SQLCS_SQL_STMNTS or SQLCS_LOG table (or a combination of them) can be used in the search clause.

1.6.9 Printing the Monitor tables

The **SQLCSPRT** program is used to print the following Monitor tables:

- the exception log table entries
- the benchmark log table entries
- the statement statistics

All the reports can be requested for the current day or for another specified chronological range. The user may add his own SELECT clauses to the standard search conditions.

The print interface has been designed in such a manner, that an installation can easily customize the reporting functions.

For a detailed description of the SQL/MF printing facilities, please refer to page 191.

1.7   Customizing

- The operation of the individual monitor components can be controlled by an installation through the monitor **configuration** files.

- The **User Attached Process** facility allows an installation to execute its own REXX programs in real-time at different points in the monitoring process.

- An installation is able to add its own interactive or printed reports to the monitor's **reporting** facility.

- Because the results of monitoring are stored in **DB2/VM tables,** these data are easily accessible to the customer for its own processing needs. (The data in the package statistics table for example, can be used for accounting purposes).

2 SQL/MF Pre-installation tasks

2.1   Determining the monitor database

The monitor database is the database where the SQL/MF tables will reside. The SQL/MF components write to the SQL/MF tables, with a possibly higher checkpoint rate as a result. To avoid conflicts of this kind with the monitored databases, we recommend to create the monitor tables in a dedicated database or in a database that is always active and lightly loaded.

2.2   Preparing the database server virtual machines

In the VM/ESA directory of all database servers to be monitored, perform the following updates:

- Add 4 megabytes to the virtual storage size of the servers.
- The performance of SQL/MF is better when the database server has VM privilege class C or E. These classes allow the monitor to extract VM-related data from the VM control blocks, rather than obtaining them via a CP command.
- To provide for the SQL/MF communications data space:
    - ensure that one additional ALET is available via the XCONFIG ACCESSLIST statement (the VM/ESA default is 62 and is usually sufficient)
    - check that the XCONFIG ADDRSPACE statement arguments MAXNUMBER and TOTSIZE provide for the SQL/MF communications dataspace
    - code the **SHARE** option on the XCONFIG ADDRSPACE statement

    If you intend to use the SQL/MF **recorder** facility:
    - ensure that one additional ALET is available via the XCONFIG ACCESSLIST statement
    - check that the XCONFIG ADDRSPACE statement arguments MAXNUMBER and TOTSIZE provide for the recorder dataspace

2.3   Define the Monitor Log machine

- In the VM directory setup a machine with a name of your choice, an A-disk of 10 3390 cylinders and a virtual storage of 16 Mb at least and. For busy database systems more than 16 Mb may be needed. We suggest to call this machine **SQLCMDM**.
- Define SQLCMDM as an XC machine in the directory.
- Ensure that SQLCMDM has access to the DB2 and SQL/MF product disks.
- Ensure that SQLCMDM is autologged. If you have the CMS Utilities feature installed, you should add SQLCMDM to the AUDITOR CONTROL file.
- In the PROFILE EXEC of the SQLCMDM machine, insert the command **EXEC SQLCS.**

2.4   Define the System Monitor machine

- In the VM directory setup a machine with a name of your choice, with a virtual storage of 8 Mb and an A-disk of 20 3390 cylinders at least. We suggest to call this machine **SQLSYSM**.
- Ensure that **SQLSYSM** has access to the DB2 and SQL/MF product disk.
- Ensure that SQLSYSM is autologged. If you have the CMS Utilities feature installed, you should add SQLSYSM to the AUDITOR CONTROL file.
- In the PROFILE EXEC of the SQLSYSM machine, insert the command **EXEC SQLMFM.**

2.5   Virtual machines for the SQL/MF end user

When using the SQL/MF interface program SQLCSU to browse the monitor tables, a virtual storage size of 8 Mb or more is recommended.[5] Some interactive monitor reports require a screen size of at least 32 lines.

2.6   Define the Monitor Shared Segment

-   Usually, the **location** of the shared segment should be higher than the virtual storage size (actual or future) of any DB2/VM server.[6] The **size** of the shared segment should be computed as **((M * 64K) + (N * 16K))** where M is the number of monitored databases and N the highest number of DB2/VM agents, concurrently active in all monitored databases. The segment size cannot exceed 8 Mb. We suggest a size of 4 Mb.
-   Using **location** and **size**, compute the begin and end address of the segment. For example: a segment located at 32 Mb and 4 Mb long, has begin address 2000000 and end address 23FFFFF.
-   Log on with a VM userid that has privilege class E ("MAINT" for example) and a virtual storage size that is at least 1 MB larger than the end of the shared segment.
-   Issue the command **DEFSEG SQLCSMDS p1-p2 SW**
    (Where **p1-p2** represent the begin and end addresses of the shared segment, expressed as **page numbers.** To define the segment at 32MB with a size of 4Mb, use page numbers **2000-23FF**.)
-   To complete the segment definition and to save the segment, issue the following commands:
    SETKEY 14 SQLCSMDS
    SAVESEG SQLCSMDS
-   The monitor shared segment is for **execution-time data** only: no software code is loaded into it.

2.7   Defining the communications dataspace

The monitor component in the database server and the statement monitor machine SQLCMDM communicate using a dataspace. This has several advantages, because the synchronous IUCV protocol is replaced with an asynchronous communication method. Moreover, dataspace-based communications will significantly increase performance.

To install dataspace-based communications, the following actions should be taken:
-   In the SQLCS CONFIG file, code the **LOG COMQ_SIZE** statement. The statement enables dataspace communications and defines the size of the dataspace in 4K pages or in megabytes (if COMQ_SIZE n MB specified).
-   The database servers and the SQLCMDM machine should have been setup has described above.

---

[5]In order to avoid lockwaits, the SQL/MF table editor reads all the selected rows from the table into a storage list and lets the user browse the list.

[6]This is not strictly necessary however. SQL/MF loads its shared segment using a CMS SEGMENT LOAD, not with a CP Diagnose. CMS SEGMENT LOAD reserves the storage occupied by the segment, so that it cannot be used for other purposes.

2.8　Granting IUCV privileges

- All monitored database servers must have the IUCV privilege to connect to the SQLCMDM virtual machine. Specify the **IUCV ALLOW** statement in the directory entry of SQLCMDM.
- The SQLCMDM and SQLSYSM virtual machines must be able to connect to all monitored database machines as a DB2/VM client.
- Users inspecting the system monitor tables must have the IUCV privilege to connect to the SQLSYSM virtual machine. As an alternative, the IUCV ALLOW statement can be specified in the directory entry of SQLSYSM.
- Users executing the Statement Monitor Interface Program (SQLCSU) must be able to perform an IUCV connect to all monitored databases and to the SQLCMDM and SQLSYSM virtual machines. Moreover, they must be able to connect to the database holding the Statement Monitor tables.

2.9   Sample CP directory entries

2.9.1   Directory entry for a monitored DB2 server

```
USER DB2TEST ****   99M 99M EG
MACHINE XC
XCONFIG ACCESSLIST ALSIZE 1022
XCONFIG ADDRSPACE MAXNUMBER 1022 TOTSIZE 2044G SHARE
ACCOUNT *** ***
OPTION MAXCONN 35 ACCT
IUCV *IDENT DB2TEST GLOBAL
IPL CMS
CONSOLE 009 3215 T
```

2.9.2   Directory entry for the Statement Monitor SQLCMDM

```
USER SQLCMDM ****   32M 32M G
MACHINE XC
OPTION QUICKDSP
ACCOUNT *** ***
IUCV <to all monitored databases>
IUCV ALLOW
IPL CMS
CONSOLE 009 3215
```

2.9.3   Directory entry for the System Monitor SQLSYSM

```
USER SQLSYSM ****   16M 16M G
ACCOUNT *** ***
IUCV <to all monitored databases>
IUCV ALLOW
IPL CMS
CONSOLE 009 3215
```

2.9.4   Directory entry for the optional Recorder Writer SQLCSCRW

```
USER SQLCSCRW ****    16M 16M G
MACHINE XC
XCONFIG ACCESSLIST ALSIZE 1022
XCONFIG ADDRSPACE MAXNUMBER 1022 TOTSIZE 2044G
ACCOUNT *** ***
IUCV <to all monitored databases>
IUCV ALLOW
IPL CMS
CONSOLE 009 3215
```

3 Installing SQL/MF

3.1   Prerequisites

- VM//ESA Version 1 Release 1 or higher
- DB2/VM Release 3.3 or higher
- A CMS minidisk (or SFS directory) of 40 3390 cylinders for the SQLMF **product disk**
- 2 PUBLIC DBspaces of 1024 pages and 1 PUBLIC DBspace of 8192 pages[7] are required in the database that will hold the monitor tables.
- Access to the DB2/VM minidisks in the virtual machine that performs SQL/MF installation.
- The ARITRAC FILEDEF in the DB2 startup should be directed to a disk file, not to tape.

3.2   Loading the software

Access the SQLMF disk or directory for write mode in filemode A.

- If you received the SQL/MF material on a cartridge, issue a **TAPE LOAD * * A**.
- If you received the material on another medium, follow the loading instructions in the README file provided.

3.3   CMS Installation

- Access the SQL/MF product disk or directory in write with filemode A.
- Invoke the installation procedure by typing **SQLCSPI**. On the next panel request CMS install by typing **Y** in the corresponding panel field.
- When the install procedure XEDITs the default SQLCS CONFIG file, specify the name of the database where you will create the monitor tables in the **LOG DATABASE** parameter. Specify or accept the defaults for the configuration parameters **MONITOR** and **LOG COMQ_SIZE** (COMQ_SIZE is to be specified only in a dataspace capable system).
- Other SQLCS CONFIG parameters can be specified now or at a later time. You may want to start with the default configuration supplied with the product. In that case, remove the comment * before the configuration statements.

---

[7]More pages may be needed in this DBspace, depending on the number of packages in the monitored databases.

3.4   DB2/VM Installation

- Access the SQL/MF product disk for write in any filemode.
- When you install SQL/MF in a database for the first time, you will be asked whether all packages should be unloaded into the monitor table SQLCS_SQL_STMNTS. It is suggested that you perform unload at this time. Unload may take a long time to complete, depending on the number of packages in the database. As package unload may cause contention on the DB2/VM catalogs, we recommend that you perform this installation during off-peak hours. In dataspace-based systems, increasing the TARGETWS will considerably speed up the load process. If you decide not to unload the packages at product installation, unload will occur automatically the first time a package is executed.
- If you have installed SQL/MF previously and you want to add a new database, stop the SQLCMDM machine. This will avoid locks and deadlocks when the database's packages are stored in the SQL/MF statements table.

3.4.1 DB2 Installation in the monitor database

- Type **SQLCSPI** and request SQL installation by typing **Y** in the corresponding panel field.
- On the next panel specify:
    - the name of the monitor database **(MONDBASE)**
    - the password of user **SQLDBA**
    - "Y" to create the monitor tables in MONDBASE
    - "Y" to install the SQL/MF packages in MONDBASE
    - "Y" to grant package and table privileges
- Specify the **STORPOOL** parameter in the SQLDBSU job streams that are shown using XEDIT. At this time, you can also modify the NPAGES parameter if you wish to do so.
- When the privileges panel is displayed, specify:
    - the name of the Statement Monitor (SQLCMDM)
    - the name of the System Monitor (SQLSYSM)
    - the name of the user that will use the monitor interface programs
- During initial installation, you will be asked whether the SQLCS_SQL_STMNTS table should be populated. If you reply Y, all packages are unloaded and stored into the SQLCS_SQL_STMNTS table.[8] If you reply N, package unload occurs when the packages are executed for the first time. A new SYSCATALOG index is also created at this time.
- Press PF3 to terminate.

---

[8]During the unload process, **do not disconnect** the virtual machine. Eventual DB2 errors during unload (e.g. SQLCODE -911 due to concurrent preps) may be ignored, since the package affected will be unloaded automatically after its first execution.

3.4.2 DB2 Installation in the monitored databases

- Type **SQLCSPI**.
- Request SQL installation.
- On the next panel specify:
  - the name of the monitored database
  - the password of user **SQLDBA**
  - "Y" to install the SQL/MF packages in the monitored database
  - "Y" to grant package and table privileges
- When the privileges panel is displayed, specify:
  - the name of the Statement Monitor (SQLCMDM)
  - the name of the System Monitor (SQLSYSM)
  - the name of the user that will use the monitor interface programs
- During initial installation, you will be asked whether the SQLCS_SQL_STMNTS table should be populated. If you reply Y, all packages are unloaded and stored into the SQLCS_SQL_STMNTS table. If you reply N, package unload occurs when the packages are executed for the first time. A new SYSCATALOG index is also created at this time.
- Repeat the installation procedure for the next database or press PF3 to terminate.

3.4.3 Adding SQL/MF end users

An SQL/MF user should be added in all monitored databases that may be inspected by that user. Note that the userid may be specified as PUBLIC.

Note
This installation option will issue DB2 grants for the SQL/MF tables and packages to the named users. The SQL/MF authorization definitions, recorded in the SQLMF ACF file (see the SQL/MF Users Guide) allow to issue SQL/MF controlled grants to each SQL/MF function. Even if a GRANT to PUBLIC is performed, access to the SQL/MF applications may still be prohibited using the SQLMF ACF file.

- Type **SQLCSPI**.
- Request SQL installation.
- On the next panel specify:
  - the name of the monitored database
  - the password of user **SQLDBA**
  - "Y" to grant package and table privileges
- On the privileges panel, specify the userid that will use the monitor interface programs.
- Enter the name of the next monitored database or press PF3 to terminate.

3.5   Installing optional SQL/MF components

3.5.1 Installing the Recorder Facility

See page 225.

3.5.2 Installing the AutoPrep Facility

See page 251.

3.5.3 Installing the saved runstats facility

See page 71.

3.6   SQL/MF maintenance

Include the monitor DBspaces SQLCS_LOG and SQLMF in your reorganization schedule. These DBspaces contain highly dynamic tables and need regular reorganization.

The table SQLCS_SQL_STMNTS in the DBspace with the same name, is a more static table. Reorganization will be needed less frequently.

3.7   Timing estimates

The CMS installation procedure will not take more than 15 minutes. The **initial** SQL installation in the MONDBASE database or another monitored database may take longer to complete, due to the unloading of all DB2/VM packages. The actual duration depends on the number of SQL packages in the database, the system load and the hardware in use. It is recommended to perform this operation on a lightly loaded system. Benchmarks show that under these conditions, one package unload will take about one second.

3.8   SQLMFSI Batch installation procedure

The **SQLMFSI** EXEC can be used to perform CMS and/or DB2 installation of SQL/MF in batch mode. This EXEC may be useful when SQL/MF has been installed in many databases.

Prerequisites

- The SQL/MF software components must have been loaded on the product minidisk.
- The SQL/MF product minidisk must have been accessed for write, in filemode A.

Invocation

   [EXEC] SQLMFSI INSTALL <databasename> <password_of_SQLDBA>

Note

- It is possible to invoke CMS and DB2 installation functions separately.
- To perform a CMS installation only, issue "**SQLMFSI CMS_INSTALL**".
- Issuing "**SQLMFSI SQL_INSTALL <databasename> <password_of_SQLDBA>**" will perform a DB2 installation only.

3.9   Verifying SQL/MF installation

When installation has been completed and after the SQL/MF components have been started, you can verify the installation by typing the following command at the CMS prompt:

SQLCVRFY n

where **n** is the number of times SQLCVRFY should read the SYSTEXT2 table (default is 1).

While SQLCVRFY is executing, you can use the SQLCSU program and its functional components to check whether the monitor is functioning as expected.

4 SQL/MF Maintenance

4.1   Reorganizing the monitor DBspaces

During monitoring, new monitor table rows are inserted by timestamp at the end of the tables. On the other hand, automatic purging of the tables by date drops rows at the beginning of the tables. Due to this type of processing, a considerable number of empty pages may exist in the SQL/MF DBspaces SQLCS_LOG and SQLMF. To recover these empty pages, both the system and statement monitoring DBspaces need regular **reorganization**.

The SQLCS_SQL_STMNTS table is a more static table. Rows are deleted and inserted only when programs are prepped and when new databases are included for monitoring. Therefore, the SQLCS_SQL_STMNTS DBspace does not need frequent reorg.

If you installation does not have a reorganization tool, the SQLMFREO EXEC can be used to reorganize the SQLCS_LOG or the SQLMF DBspaces. The syntax of the EXEC is as follows:

SQLMFREO
TABLE { SYSLOG | SYSMON }
PASSWORD nnn
STORPOOL nn
[**STEP** n]

- Specify TABLE SYSLOG to reorganize the SQLCS_LOG table. Specify TABLE SYSMON to reorganize the System Monitor tables.
- For PASSWORD, specify the password of user SQLDBA in the database containing the SQL/MF tables.
- For STORPOOL, specify the storage pool that will contain the reorganized DBspace. (You can use this argument to relocate the DBspace).
- The optional argument STEP to restart reorganization after an error.[9]

---

[9] Reorganization steps:

1	determines table grantees
2	unloads the DBspace
3	drops the DBspace
4	acquires the DBspace, creates table and indexes
5	reloads the DBspace
6	regrants the table

4.2   Moving the monitor DBspaces to another database

4.2.1 Method 1

- Execute **SQLCSCLR** (see page 332) for all databases currently monitored. The EXEC will request the name of the previous log database and the name(s) of the monitored databases.
- Update the LOG DATABASE statement in the SQLCS CONFIG file and re-access the product disk.
- Invoke SQLCSPI. Request monitor table creation in the new log database. Specify the name(s) of all monitored databases. This will unload all packages into the new SQL_STMNTS table (because the SYSCAT_TABID_I1 on SYSTEM.SYSCATALOG has been dropped by SQLCSCLR).

Due to the package unload, this method may take a long time to complete.

4.2.2 Method 2

- Update the LOG DATABASE statement in the SQLCS CONFIG file and re-access the product disk.
- Invoke SQLCSPI. Request monitor table creation in the new log database. Request installation for all monitored databases. No package unload into the SQL_STMNTS table will occur (because the SYSCAT_TABID_I1 on SYSTEM.SYSCATALOG has not been dropped).
- Use SQLDBSU UNLOAD / RELOAD (or an equivalent method), to transfer the SQLCS_SQL_STMNTS table from the previous to the new log database.
- Manually drop the DBspaces SQLMF, SQLCS_LOG and SQLCS_SQL_STMNTS in the previous log database.

Transfer using this method will take less time to complete.

5 Configuring SQL/MF

5.1 Configuring the Statement Monitor

In addition to its basic monitoring function, the Statement Monitor can be requested to perform logging, benchmarking, recording and notification by coding the corresponding control statements in a configuration file, named **SQLCS CONFIG.** The configuration file should be placed on the SQL/MF software disk.

- The configuration file starts with a **global** (unnamed) section that contains the parameters applicable to **all** monitored databases. Parameters that **must** be coded in the global section are: MONITOR, MONSEGMENT and LOG DATABASE. The global section can also contain the configuration parameters identical for all databases.

- Configuration parameters applicable for a single database only, should be placed in the corresponding **database section** of the configuration file. The begin of a database section is indicated by the **SECTION** statement.

With the exception of MONITOR, MONSEGMENT and LOG DATABASE, all configuration statements can be specified in both the global and the database sections. When a given parameter appears in both the global and the database section, the database related parameter prevails.

The configuration file is a fixed or varying length CMS file with a recordsize not exceeding 256 characters. The configuration statements are coded in a free format: the first word of the statement is considered as the function word. All following words are function arguments. They are separated from the function word by one or more blanks. The entire function statement can be coded on a single file line, or multiple lines can be coded, with the function word repeated. If a statement is not coded, the corresponding function is not implemented. A line starting with an asterisk, is considered to be a comment line.

The configuration file is used by the programs SQLCSI (running in the database server), SQLCS (running in the Statement Monitor Log machine) and the user interface program SQLCSU.

- The SQLCS program uses following parameters from the SQLCS CONFIG file

    - LOG DATABASE / LOG LOCKTIME / LOG IDLETIME / LOG RETAIN
    - MODLEVEL
    - MONITOR
    - NOTIFY LOCKTIME / NOTIFY IDLETIME / NOTIFY RESPTIME

- The parameters LOG LOCKTIME, LOG IDLETIME, LOG RETAIN, MODLEVEL, NOTIFY LOCKTIME, NOTIFY IDLETIME and NOTIFY RESPTIME are **not** used by SQLCSI, although they are accepted (and ignored) by this program, when it processes a configuration file containing them.

- The SQLCSU program uses the parameters LOG DATABASE and MONITOR.

5.1.1 List of configuration statements

The configuration file is processed at the start of the SQLCS and SQLCSI programs. Following statements can be coded in the configuration file (see page 77 for a sample file).

```
ANALYZE              SERVER <VM_user> REPORTDEST <VM_user>
AUTO_REBIND
AUX_MONITOR          VMname
BENCHMARK            USER {username | *}
                     PACKAGE {packagecreator | *}.packagename
ENDLINKS
EXCLUDE              {DB2USER|VMUSER|PACKAGE} nnnn
FORCE_RO
FORCE_RW             DBspacename
INCLUDE              {DB2USER|VMUSER|PACKAGE} nnnn
INITIAL              SUSPEND
LOCK                 DIRBUF|PAGBUF|ADDRESS [<address>] <nr_pages>
LOG                  DATABASE databasename
                     BUFLOOK n
                     CHKPT_WAIT n
                     COMQ_SIZE n [MB]
                     CCSID_TRANS
                     DEADLOCKS
                     DYNCOM
                     ESCALATE
                     EXCLUDE { DB2USER | VMUSER | PACKAGE } nnnn
                     HIGHEST { ON | OFF }
                     IDLETIME n
                     ISOLATION RR
                     LOCKTIME n
                     LUW_RESPONSE { hhmmss | ALL }
                     MODLEVEL n
                     NOBLOCK
                     PERIOD FROM hhmm TO hhmm
                     RESPONSE hhmmss
                     RETAIN n m
                     SCAN DBSP
                     SQLCODE n m
                     STAT_FREQ n
                     TOTIO n
   IBUFLOOK          n
   MINPOOL_PAGES     n
   MONITOR           VMname
   MONSEGMENT        Shared_segment_name
   MSGROUTE          VMname
   NOTIFY            NAME VMname
                     BUFLOOK n
                     ESCALATE
                     IDLETIME n
                     ISOLATION RR
                     LOCKTIME n
                     LOGUSAGE n
                     NOFILE
                     NOMSG
                     RESPTIME n
                     TOTIO n
   PERIOD            FROM hhmm TO hhmm
   RETAIN_RUNSTATS   n
   SECTION           Databasename
   SECTION_STATS     {SECTION | SESSION | DAILY}
   SET ISOLATION     {CS | RR | UR} FOR <packagename>
   SMSGROUTE         VMname
```

5.1.2 ANALYZE statement

Purpose

The statement is used in conjunction with the LOG statement. It causes all logged statements to be forwarded automatically to the SQL/Command Analysis program[10], if installed on your system.

Note that:

- SQL/Command Analysis should execute in server mode
- the ANALYZE statement does not apply to benchmarked statements

Syntax:

ANALYZE SERVER VM_username REPORTDEST VM_username

Arguments:

SERVER VM_username
Specify the name of the virtual machine that runs the SQL/Command Analysis program in **server** mode.

REPORTDEST VM_username
Specify the VM userid to which the analysis reports should be routed.

---

[10]SQL/Command Analysis is a tool offered by Software Product Research.

5.1.3 AUTO_REBIND statement

Purpose

The AUTO_REBIND causes all CREATE INDEX statements to be recorded. A **package rebind** will be requested for all packages that depend on tables for which a new index has been created. This ensures that existing packages benefit from the newly created indexes. The rebind is actually performed by the Statement Monitor machine at the end of its session, that is, at midnight by default.

Syntax:

AUTO_REBIND

5.1.4 AUX_MONITOR statement

Purpose

If the Governor and UAP logic is not to be executed by SQLCMDM but by a dedicated machine, specify the name of that machine here. It will cause the UAP8 and UAP9 signals (begin and end of checkpoint) to be sent to the aux_monitor rather than to SQLCMDM. How to define a dedicated Governor is described on page 91.

Syntax:

AUX_MONITOR VM_name

5.1.5 BENCHMARK statement

Purpose

Benchmarking automatically logs all the SQL statements in the designated programs, executed by the designated user. These statements are stored in the log table with the *benchmark* log type, to distinguish them from statements logged *by exception,* as described below.

Multiple BENCHMARK statements can be stated in the configuration file.

Syntax:

BENCHMARK USER {username | *} PACKAGE {packagecreator | *}.packagename

Arguments:

USER {username | *}
Specify the DB2/VM userid subject to benchmarking when it executes the named package.
Specify USER * if the package name alone should be used as a criterion for benchmarking.

PACKAGE {packagecreator | *}.packagename
Specify the creator and name of the package subject to benchmarking.

Note

Both username and packagename can be made generic by appending an * sign, in which case the non-generic part of the name should not exceed 7 characters. Be aware of the significant overhead that generic benchmarking may generate. If your intention is to monitor multiple packages or users, the same results can be obtained with much less overhead, using the Recorder facility.

Example:

BENCHMARK USER USR1 PACKAGE X.PACK1

Logs all SQL statements in the package X.PACK1 when executed by user USR1.

BENCHMARK USER * PACKAGE *.PACK2

Logs all SQL statements in the package PACK2, whatever the executing userid and the package creator.

5.1.6 ENDLINKS statement

Purpose

The statement is an alternative for the MONITOR ENDLINKS command. It requests that the ENDLINKS facility is automatically executed during DB2 shutdown. ENDLINKS will automatically force all active LUWs at shutdown and terminate all active pseudo-agents, by severing their APPC or TCP/IP paths. See also page 209.

Syntax:

ENDLINKS

5.1.7 EXCLUDE and INCLUDE statements

Purpose

To limit the scope of a LOG criterium, INCLUDE and INCLUDE statements can be specified for the following LOG exception criteria:

- BUFLOOK
- CHKPT_WAIT
- CCSID_TRANS
- DEADLOCKS
- DYNCOM
- ESCALATE
- IDLETIME
- ISOLATION RR
- LOCKTIME
- LUW_RESPONSE
- NOBLOCK
- RESPONSE
- SCAN DBSP
- SQLCODE
- TOTIO

The INCLUDE and EXCLUDE statements must immediately follow the defined criterium.

Syntax:

{INCLUDE | EXCLUDE} {DB2USER | PACKAGE | VMUSER} n

- The DB2USER specification includes or excludes the named DB2 userid(s).
- The PACKAGE specification includes or excludes the named DB2 package(s).
- The VMUSER specification includes or excludes the named VM userid(s).

The name of the included or excluded user or package can be made generic by ending it with an * sign. If multiple INCLUDE and EXCLUDE statements are present for a given LOG criterium, their evaluation proceeds until an INCLUDE request is satisfied. If no such request is found, the user or package is excluded if it has matched a previous EXCLUDE request. If no INCLUDE or EXCLUDE statements are present, all users and packages are included by default.

Example
```
LOG DYNCOM
EXCLUDE PACKAGE *
INCLUDE PACKAGE ARI*
LOG TOTIO
EXCLUDE PACKAGE XYZ
```

The first example limits DYNCOM logging to the packages whose name starts with ARI.
The second example excludes package XYZ from TOTIO logging.

Note
The INCLUDE / EXCLUDE facilities described above provide for a more granular exclude than the LOG EXCLUDE { DB2USER | VMUSER | PACKAGE } statement which disables **all** logging criteria for the defined users or packages.

5.1.8 FORCE_RO statement

The statement forces the current database to read-only (except DBspace 1 and the monitor Dbspaces). This statement **must** be coded after a SECTION statement. Multiple databases can be forced to the read-only status.

The FORCE_RO facility can be temporarily disabled or enabled for all or for a named DBspace, using the FORCE_RO and FORCE_RW commands described on page 210.

5.1.9 FORCE_RW statement

Syntax

FORCE_RW DBspacename

When the FORCE_RO statement is in effect for a database, named Dbspaces can be exempted from the read-only restriction by coding their name on a FORCE_RW statement.

There is no limit on the number of FORCE_RW statements that can be coded.

5.1.10   INITIAL statement

Purpose

Start the SQLCSI monitor component in suspended mode. To activate monitoring, the MONITOR RESUME statement must be executed.

Syntax

INITIAL SUSPEND

5.1.11   LOCK statement

Purpose

The statement allows to lock parts of the DB2 buffer pool or a DB2/VM module in real storage, so that no more pagefaults occur when these virtual storage pages are referenced during execution, with improved performance as a result.

Syntax:

LOCK { DIRBUF | PAGBUF | ADDRESS } [<address>] <number_of_pages> [%]

LOCK DIRBUF <number_of_pages> [%] locks the designated number of 4K pages at the beginning of the directory buffer pool. As a directory buffer is 512 bytes, the size of the directory buffer pool equals (NDIRBUF/8) 4K pages. <number_of_pages> may be expressed as a percentage, as explained in usage note 1.

LOCK PAGBUF <number_of_pages> [%] locks the designated number of 4K pages at the beginning of the page buffer pool. Each page buffer is 4K bytes. <number_of_pages> may be expressed as a percentage, as explained in usage note 1.

LOCK ADDRESS <address> <number_of_pages> locks the virtual storage area designated by "address" and "number_of_pages". See usage note 3.

Usage notes

(1) The LOCK DIRBUF and LOCK PAGBUF statements allow to specify the number of pages to lock as a percentage of the total number of pages in the directory or page buffer pool. This makes the LOCK specification independent of any changes in the NDIRBUF and NPAGBUF parameters. A percentage is recognized when the <number_of_pages> value is followed by a % sign, which must be separated from the value by at least one blank.
(2) When initializing and terminating, SQL/MF locks and unlocks the requested pages by issuing the CP LOCK and UNLOCK commands. Both commands require CP privilege class A. If the privilege is unavailable, an error message is issued and operation continues without locking.
(3) The LOCK ADDRESS statement allows to fix any virtual machine page in storage, including pages in the DB2 shared segments. This allows to lock high usage DB2 modules, such as the dispatcher (ARICDSP) in real storage. Use the DB2 link map to determine the load address of the particular module and determine its length by subtracting the module's load address from the load address of the module that follows it in the link map.
(4) Up to 999 LOCK statements can be specified in the SQLCS CONFIG or in a CONFIG SECTION.
(5) The CP LOCK command is performed by the monitor when the first SQL statement of the section is executed. At this time, the locked area is displayed on the console as the starting and ending hexadecimal **page** numbers. To obtain the real address range locked, the page number must be multiplied by 4096, by appending 3 zeroes to the hexa page number.

Examples

-    "LOCK DIRBUF 50 %" locks the first half of the directory buffer pool.
-    "LOCK PAGBUF 256" locks the first 256 4K-pages of the page buffer pool
-    "LOCK ADDRESS 0 1" locks the virtual machine's hardware-software interface page (page zero) in real storage. Page zero is heavily used as all interrupts are reflected by CP to the VM using this page.

5.1.12 LOG statement

Purpose

- Designate the database where the monitor log table has been created.
- Defines communications protocol between SQLCSI in the database servers and SQLCMDM.
- Specify the number of days that log table rows should be kept.
- Define the criteria for statement logging. Logging is performed when a DB2/VM user exceeds the resource consumption limits defined by the LOG statement. This type of logging is called *exception logging*, to distinguish it from *benchmark logging* described above. If a given statement violates several logging limits during a single execution, it is logged on behalf of each exception caused.
- Exception logging can be limited using the LOG EXCLUDE or the INCLUDE and EXCLUDE statements after each exception criterion. For a description of the latter facility, see page 52.
- The statement and package statistics function is activated as soon as the MONITOR and LOG DATABASE statements have been supplied.

Syntax

```
LOG     DATABASE databasename
        BUFLOOK n
        CCSID_TRANS
        CHKPT_WAIT n
        COMQ_SIZE n [MB]
        DEADLOCKS
        DYNCOM
        ESCALATE
        EXCLUDE { DB2USER | VMUSER | PACKAGE } nnnn
        HIGHEST [ON | OFF]
        IDLETIME n
        ISOLATION RR
        LOCKTIME n
        LUW_RESPONSE { hhmmss | ALL }
        MODLEVEL n
        NOBLOCK
        PERIOD FROM hhmm TO hhmm
        RESPONSE hhmmss
        RETAIN n m
        SCAN  DBSP
        SQLCODE n m
        STAT_FREQ n
        TOTIO n
```

Arguments:

DATABASE databasename

Designates the database containing the SQL/MF tables. Several SQL/MF components will automatically perform an SQLINIT to this database.

BUFLOOK n

Logs statements that perform more than N buffer lookups.

CCSID_TRANS

Logs SQL statements for which DB2 performs CCSID translation (with a considerable performance penalty as a result). SQL/MF logs such statements with log reason **CCSIDTRN**.

Since detection of CCSID translation may significantly increase monitor overhead, the CCSID_TRANS option should be requested with discretion. Note that INCLUDE and EXCLUDE statements can be used to limit the monitor overhead to designated users or packages.

CHKPT_WAIT n

Logs statements that had to wait more than **n** seconds for a checkpoint completion. If **n** is set to zero, all checkpoint waits will be logged. The exception log row for a checkpoint_wait event shows the number of milliseconds, the agent has been in checkpoint wait, in the column "comm_wait".

COMQ_SIZE n [MB]

Specify this statement only if dataspace support is available on your system. COMQ_SIZE defines the size of the communications dataspace used to transmit package execution statistics, logging and benchmarking requests from SQLCSI to the statement monitor SQLCMDM. If the statement is omitted or when the size is set to zero, IUCV will be used for communications. For reasons of performance, we strongly recommend to use dataspaces, whenever possible.

The size of the communications dataspace can be expressed in 4K pages (COMQ_SIZE n) or in megabytes (COMQ_SIZE n MB).[11] The minimum allocation is 1 MB (or 256 4K-pages[12]), the maximum 2048 MB (or 524288 4K-pages).

A dataspace of 16 megabytes is usually adequate, unless intensive logging or benchmarking is performed. All entries in the dataspace are 512 bytes long. For logging, benchmarking and prep monitoring, the monitor agent block and the dynamic statement text are also stored. In the average, most log records will fit in one 4K block.

Use the MONITOR CQCOUNT command (page 202) to monitor space usage in the dataspace. If there is no room in the dataspace to forward a request, a message is given and the request is ignored.

Although it is possible to code the statement in an SQLCS CONFIG "section", there is usually no reason to do so. For best performance, code the statement in the global section.

For additional information on the dataspace setup, refer to page 32.

DEADLOCKS

Logs statements during which a deadlock has occurred.

---

[11]Note the blank between n and MB.

[12]If less than 256 pages are requested, the size is automatically and without error message adjusted to 1 MB.

DYNCOM

Logs all SQL statements executed in dynamic mode. Please note that the first data transfer statement (fetch, put..) following the prepare and open will be logged. A given dynamic statement is logged only once per session, that is, the *text* of the dynamic statement is used to determine whether previous logging has occurred.

ESCALATE

Logs all statements during which one or more lock escalations have occurred.

EXCLUDE { DB2USER | VMUSER | PACKAGE } nnnn

Excludes the name DB2 user, VM user or package from exception logging. Up to 999 EXCLUDE requests can be specified.

HIGHEST [ON | OFF]

With LOG HIGHEST ON, the RESPTIME, LOCKTIME, TOTIO and BUFLOOK exception loggings store the highest values encountered thus far for the package section affected. If a package section causes multiple LOCKTIME log events in the database session, the highest LOCKTIME is retained. With LOG HIGHEST OFF (the default), all exceptions are logged chronologically.

IDLETIME n

Logs the users that are in logical unit of work for the specified number of seconds, without issuing SQL statements. Idletime is computed as the difference between the current time and the endtime of the last statement executed.

ISOLATION RR

Logs all statements executing with an isolation level of *repeatable read*.

LOCKTIME n

Logs users that are in the lockwait state for the specified number of seconds.

LUW_RESPONSE { hhmmss | ALL }

Logs DB2/VM sessions not closing their LUW before **hhmmss** time has elapsed. Leading zeroes in the time specification may be omitted. The specification 100 (1 minute) is equivalent to 000100. The LUW_RESPONSE time is calculated as: (time_of_commit) - (start_time of first statement in the LUW).

If LUW_RESPONSE **ALL** is specified, all LUWs will be logged, whatever their duration. Please note that this option may insert a large number of rows in the SQLCS_LOG table.

MODLEVEL n

This parameter specifies the number of package modification levels (including the current level) to keep in the SQLCS_SQL_STMNTS table. A package modification level is created each time a package is prepped or reloaded. For instance: to keep the current and one previous modification level, specify MODLEVEL 2. If the parameter is omitted, no levels other than the current one are kept.

NOBLOCK

Logs the cursor-based statements that execute without blocking. Programs retrieving a large number of rows should normally execute with the blocking option.

PERIOD FROM hhmm TO hhmm

Defines the exception logging period. By default, exception logging is not limited chronologically.

RESPONSE hhmmss

hhmmss represents the statement response time ("elapsed time") that will trigger logging. Leading zeroes may be omitted. The specification 100 (1 minute) is equivalent to 000100.

The response time is calculated as the sum of: CPUtime, lock waittime, I/O wait_time and communications waittime. Response times are computed for the package **section**. For cursor statements for instance, "response time" represents the time elapsed between the opening and the closing of the cursor.

For example :   **LOG RESPONSE 0130** will log all statements with a response time of 1½ minute and higher.

RETAIN n m

Specify **n** as the number of days the exception and benchmarking log table rows should be kept. Rows in excess of this argument are automatically deleted by SQLCS.

Specify **m** as the number of days the package statistics should be kept. Rows in excess of this argument are automatically deleted by SQLCS. If **m** is omitted, the package statistics are not automatically purged.

If the parameter is omitted, the log table is not automatically purged.

SCAN DBSP

Specify if DBspace scans should be logged.

SQLCODE n m

If specified, statements terminating with a negative SQLCODE in the range **N M** (meaning: from **N** up to and including **M)**, will be stored in the Monitor Log table. If **M** is omitted, the single SQLCODE **N** will be logged. An unlimited number of SQLCODE ranges may be specified. However, each range must be coded an a new line.

For example:

To log all SQLCODEs specify:
LOG SQLCODE 1 999

To log all privilege related SQLCODEs and all deadlocks, specify:
LOG SQLCODE 551 564
LOG SQLCODE 911

Please note that only SQLCODEs generated by the database server can be detected by SQL/MF. SQLCODEs generated by the Resource Adapter at the client side (for example -521) are not logged.

STAT_FREQ n

Specifies the frequency of statement and package statistics, that is, the interval after which the Statement Monitor component executing in the database server should transmit its statistics to the Monitor Log virtual machine (the SQLCS program).

Specify the value as:

1 to perform logging every hour
2 to perform logging every 10 minutes
3 to perform logging every minute

The higher the frequency, the sooner the statement and package statistics can be consulted using the statistical reports, but the higher the performance impact of statistics processing. The package statistics are also sent when a package is prepped, because the prep monitor event clears the statistics. If STAT_FREQ is not specified, the default value is set to 1.

TOTIO n

Logs statements that perform more than **n** page I/O's or dataspace page transfers.

5.1.13   IBUFLOOK statement

Purpose

Enable the measuring of agent dispatch delays. Dispatch delay measuring shows the effect of the DB2/VM DISPBIAS startup parameter.

A dispatch delay is the time, in microseconds, between the clearing of a wait condition (posting) and the actual dispatching of the agent by DB2/VM. Dispatch delays are measured for short wait conditions only, that is, for I/O and dataspace pagefault waits. The average dispatch delay during the system monitor interval is stored in the System Monitor COUNTER table. A separate delay counter is provided for interactive and for non-interactive agents. An agent is considered non-interactive by SQL/MF, when it performs more than (IBUFLOOK) buffer lookups during its LUW.

For example, the statement IBUFLOOK 1000, causes agents doing more than 1000 buffer lookups to be considered as non-interactive.

Syntax

IBUFLOOK   n

5.1.14 MAX_PAGES statement

Purpose

Avoid dispatching delays experienced by short (interactive) LUWs while long-running LUWs are in progress.

These delays are due to the fact that DB2 allows a long-running LUW to consecutively process 64 DBspace pages when no wait conditions arise during the retrieval. After 64 pages, the DB2 page retrieval module ARIYI19 forces an exit to the dispatcher. However, while processing the 64 pages, the agent is in the DBSS and effectively monopolizes the database.

The purpose of the MAX_PAGES operand is to reduce the number of pages that can be retrieved without exiting to the dispatcher. Decreasing MAX_PAGES will improve concurrent agent processing.

Syntax

MAX_PAGES n

where **n** specifies the maximum number of consecutive page retrievals allowed for any agent

Note

Reducing MAX_PAGES will somewhat increase the database CPU usage, as more calls are made to the DB2 dispatcher. You should experiment with different values of MAX_PAGES to find the best balance between CPU overhead and processing concurrency.

5.1.15   MINPOOL_PAGES statement

Purpose

Defines the minimum number of free pages to be maintained in each storage pool.

If the statement is supplied, SQLCODE -700 and SQLERRP = 'SQLCSI' will be forced by SQL/MF for all INSERT and UPDATE statements that cause the free pages in any pool to drop below the specified minimum. It ensures that no application can fill all the pages in a pool.

MINPOOL_PAGES may also be used to avoid the DB2/VM system abend that occurs in some cases (after message **ARI0201E**) when no free pages are available in a storage pool.

Syntax

MINPOOL_PAGES n

where **n** specifies the minimum number of free pages to be left in any storage pool

5.1.16   MONITOR statement

Purpose

Designate the name of the virtual machine that will run the SQLCS program. The MONITOR statement is required, unless no logging is desired and the package_unload or object_translation functions for SQLCSU are not required. The default name is SQLCMDM.

Syntax

MONITOR VM_username

5.1.17   MONSEGMENT statement

Purpose

Designates the name of the VM shared segment to be used for storing the monitor data and for passing monitored data between the components of the Monitoring Facility.

If no MONSEGMENT statement is specified, the shared segment name defaults to **SQLCSMDS** and is global to all monitored databases.

Using multiple shared monitor segments on the same physical system allows to logically separate databases for the purpose of monitoring. These databases should be provided with a private configuration file, containing the pertinent MONSEGMENT statement.

In most cases, there is no need to specify the MONSEGMENT statement.

Syntax

MONSEGMENT segment_name

5.1.18   MSGROUTE statement

Purpose

If the MSGROUTE statement is specified, severe SQL/MF messages issued by the components SQLCSI, SQLCMDM and SQLSYSM are routed, using a CP MSG command, to the named VM userid. How to configure the MSGROUTE target is described on page 67.

Syntax

MSGROUTE VM_userid

Notes

1. MSGROUTE is a **global** configuration statement for the SQLCMDM and SQLSYSM components: message routing is done by these components as soon as an MSGROUTE statement is found in SQLCS CONFIG.

2. For the SQLCSI component executing in the database server, MSGROUTE may be local, that is, be coded within a database SECTION. Routing then occurs only for the databases having the MSGROUTE in their section.

3. Due to its implementation, a modified MSGROUTE can be activated in the SQLCMDM and SQLSYSM components only by restarting them. (IPL CMS)

5.1.19   NOTIFY statement

Purpose

The **Notify** facility sends a message to a designated VM user when one of the specified exceptional conditions occur. The message is sent using the CMS **TELL** command. In addition, all messages sent are recorded in a CMS file called <databasename> SQLCSLOG. This file is automatically sent to a designated user at midnight. The target virtual machine of the NOTIFY user should run with CP MSG ON.

The notification criteria are checked by the SQLCS program executing in the Statement Monitor log machine.

Syntax

```
NOTIFY  NAME username
        BUFLOOK n
        DYN_COST n
        ESCALATE
        IDLETIME n
        ISOLATION RR
        LOCKTIME n
        LOGUSAGE n
        NOFILE
        NOMSG
        RESPTIME n
        TOTIO n
```

NAME username

Specify the VM userid that should receive the notification messages. SQL/MF sends the notification using a CMS **TELL** command. Therefore, "username" may also represent a nickname, defined using the CMS **NAMES** statement. Using a nickname that represents multiple userid's, the notification can be sent to several virtual machines.

BUFLOOK n

Generates a notification message when a package section performs more than **N** buffer lookups.

DYN_COST n

Generates a notification message when a dynamic SQL statement is executing, with a PREPARE cost greater than **n**.

ESCALATE

Generates a notification message whenever an agent causes DB2 lock escalation.

IDLETIME n

Generates a notification when a logical unit of work is idle (not executing SQL statements) for more than **N** seconds.

ISOLATION RR

Generates a notification when a package section executes with the *repeatable read* isolation level.

LOCKTIME n

Generates a notification when a statement is in the lockwait state for more than **N** seconds.

LOGUSAGE n

Generates a notification when the DB2 log is **n**% full.[13]

NOFILE

Specifies that notification messages should be not recorded in the SQLCSLOG dataset. By default, all messages are recorded in this dataset.

NOMSG

Requests that notification messages be recorded in the SQLCSLOG dataset but not sent to the user. By default, all notification messages are sent.

RESPTIME n

Generates a notification when a statement has a response-time of more than **N** seconds.

TOTIO n

Generates a notification when a package section has issued more than **N** I/O requests.

---

[13]For users wanting to process the notification in a PROP machine, the format of the message is: **SQLCSI005 LOGUSAGE Notification. LOGMODE x, user xxx, package xxx, section xxx**

5.1.20   PERIOD statement

Purpose

Defines the monitoring period, that is, the chronological window during which monitoring will occur. This allows to reset monitoring during table backup or reorganization procedures, executed off shift.

When monitoring is reset, the host command interface is still enabled. This allows to execute DB2/VM, CP, CMS or MONITOR commands, even if monitoring has been suspended.

Syntax

PERIOD FROM hhmm TO hhmm

Specify the start and end time for monitoring in hours and minutes. The TO time should be greater than the FROM time.

5.1.21 RETAIN_RUNSTATS statement

Purpose

Defines the number of days that should be kept in the saved Runstats table. Data in excess of RETAIN_RUNSTATS are automatically dropped by the SQLCMDM component.

Syntax

RETAIN_RUNSTATS n

where n specified the number of days to be retained

Creating the saved Runstats table

Create the saved Runstats table as follows:

[EXEC] SQLCSCRT PASSWORD n PAGES n STORPOOL n

where:

PASSWORD is the password of user SQLDBA in the SQL/MF database
PAGES is the number of pages to allocate for the saved runstats DBspace
STORPOOL is the storpool where this DBspace is to reside

5.1.22   SECTION statement

Purpose

Indicates the start of "private" configuration parameters for the named database. These parameters complement or overwrite the parameters specified in the global configuration section. A database-specific section ends at the SECTION statement for the next database or at the end of the configuration file.

Syntax

SECTION databasename

5.1.23 SECTION_STATS statement

Purpose

Controls the contents of the statement statistics columns in the SQL_STMNTS table (e.g. RDSCALL, TOTIO etc).

Syntax

SECTION_STATS { <u>SECTION</u> | SESSION | DAILY }

- **SECTION** the columns contain the statistics for the **last** execution of the section
- **SESSION** the columns contain the **total** statistics for all executions of the section during the database **session**
- **DAILY** the columns contain the **total** statistics for all executions of the section during the **day**

If the statement is omitted, SECTION_STATS SECTION will be in effect.

Note

The meaning of the data in the column **SAMPLED** of the SQL_STMNTS table also depends on the SECTION_STATS statement.

- With SECTION_STATS SECTION, the column records the number of times the package section has been executed since it was last prepped (or since it was loaded in the SQL_STMNTS table at SQL/MF install).

- With SECTION_STATS SESSION, the column shows the number of times the package section has been executed during the corresponding database session.

- With SECTION_STATS DAILY, the column shows the number of times the package section has been executed during the day, it was last recorded.

5.1.24   SET ISOLATION statement

Purpose

Force a defined isolation level for specified packages. The isolation level is modified by the monitor in the RDIIN mail block at execution time. The package then runs as if it had been prepped with the specified isolation level.

Syntax

SET ISOLATION { CS | RR | UR } FOR <packagename>

Notes

- a generic packagename is specified by appending an * sign
- package name * means "all packages"
- the SET ISOLATION statement may appear in the global or in a database section
- up to 256 SET statements may be specified per database
- multiple SET statements are processed in the order of specification

Example

SET ISOLATION UR FOR ARI*
SET ISOLATION UR FOR SQL*
SET ISOLATION CS FOR *

The above statements will force isolation UR for all packagenames beginning with ARI or SQL and isolation level CS for all other packages.

5.1.25   SHOW_ALTID statement

Purpose

If an alternate VMID is in effect after a Diagnose D4, specify SHOW_ALTID to use the primary VMID instead of the alternate. Under the SQLDS protocol, the alternate VMID will always be shown.

Syntax

SHOW_ALTID

5.1.26   SMSGROUTE statement

Purpose

If the SMSGROUTE statement is specified, severe SQL/MF messages issued by the components SQLCSI, SQLCMDM and SQLSYSM are routed, using a CP SMSG command, to the named VM userid. How to configure the SMSGROUTE target is described on page 67.

Syntax

SMSGROUTE VM_userid

Note

See the notes for MSGROUTE.

5.1.27 Sample configuration file

* *Global configuration parameters*

MONITOR SQLCMDM
LOG DATABASE database_name
LOG COMQ_SIZE 16 MB
PERIOD FROM 0800 TO 1800
LOG SCAN DBSP

* *Configuration parameters for database DBPRD1*

SECTION DBPRD1
LOG BUFLOOK 5000
LOG TOTIO 100
LOG RESPONSE 30
LOG SQLCODE 1 999
NOTIFY NAME ADMINIST
NOTIFY LOCKTIME 30

* *Configuration parameters for database DBTEST1*

SECTION DBTEST1
LOG BUFLOOK 10000
LOG TOTIO 500
BENCHMARK USER * PACKAGE *.PROG1

5.2   Configuring the Governor Facility

The Governor function is performed by the SQLCS program running in the SQLCMDM machine or the AUX_MON machine, if the latter has been defined (see page 91). The Governor operates using CMS files that define the restricting conditions and the users to be restricted. Restricting a user consists in defining a maximum resource usage limit. When a user exceeds the limit during execution, the governor forces the user off the database. If the governor finds the user provided **SQLMUAPX EXEC** on its search chain, it will invoke the EXEC before restricting. The user exit may request that restricting and forcing be bypassed. The SQLMUAPX interface is described on page 304.

Restrictions are defined for a given database by providing a restrict file named:

> **<databasename>[14] RESTRICT**

If no database related restrict file is found, the **GLOBAL RESTRICT** file is searched for. If not found, restricting is not attempted. The restriction control files should be located on the SQL/MF software disk. Modified restriction files take effect when SQLCMDM is restarted or at its next autostart, which occurs at midnight.

The restrictions are applied at the time interval specified by the INTERVAL argument at the startup of the SQLCS program. If the INTERVAL argument is omitted, it defaults to 10 seconds and the Governor will examine the active users for possible restricting every 10 seconds. A statement that completes before this interval expires, will not be restricted, even if it has violated a restricting rule. When restriction such as MAX_BUFLOOK, MAX_ROWS, MAX_IO or MAX_LOGPAGES are defined, it will usually be necessary to set the SQLCS INTERVAL to a much lower value.

A restriction control file may contain following statements, which can be specified in any order. The INCLUDE and EXCLUDE statements however **must follow the condition to which they apply**. A line starting with an asterisk, is considered to be a comment line.

Defined restrictions (MAX_....) may be defined several times, each time with a different restriction value and a different INCLUDE / EXCLUDE list.

Several RESTRICT FROM blocks may be specified. Each block should define a restriction period and contain specific restriction statements.

---

[14]The name of the database, not the VM_id of the server.

RESTRICT FROM hhmm TO hhmm [WEEKDAYS]

Defines the period during which restriction should be enabled. If the optional WEEKDAYS keyword is specified, restricting will not be attempted on Saturdays and Sundays.

MAX_BUFLOOK n

Requests that an agent be forced when it performs more than **n** buffer lookups.

MAX_CHKP_DELAY n [FORCE_RO | FORCE_RW]

Requests that agents responsible for a system checkpoint delay be forced after **n** seconds. **FORCE_RO** requests that only read agents should be forced. **FORCE_RW** requests that only read-write agents should be forced. When the operand is not specified, all agents responsible for the delay will be forced. Please note that in most cases, checkpoint delay is caused by read-write agents. The DB2/VM design is such that read-only agents will voluntarily leave DBSS when a checkpoint is pending.

MAX_DYN_COST n

Requests that dynamic SQL statements be forced when their cost exceeds **n**. The cost is determined by DB2 during statement PREPARE. The statement will be allowed to start but will be forced at the next SQLCMDM interval (which defaults to 10 seconds).

MAX_IDLETIME n

Requests that an agent be forced when it is idle for more than **n** seconds. An agent is considered idle when it is in LUW, when its previous statement has completed and when it is in the communications wait state (waiting for the next SQL statement).

MAX_IO n

Requests that an agent be forced when it is performs more than **n** I/O operations. **All** I/O (page I/O, directory I/O and logfile I/O) is taken into account in view of restricting. In dataspace-based systems, a transfer of a page between the buffer pool and the dataspace is counted as an I/O.

MAX_LOCKHOLD n [IDLE m] [FORCE_RO | FORCE_RW]

Requests that an agent be forced when it locks a resource for more than **n** seconds, while other agents wait on that resource (are in lockwait). In case of a lock cascade, the first agent not in lockwait will be forced.

The optional **IDLE** keyword requests that the locking agent must be in the communications wait state for at least **m** seconds, before restricting is considered. If IDLE is not specified, the agent exceeding the hold time is forced, whatever its status. Normally, the IDLE time should be inferior to LOCKHOLD time.

The optional keyword **FORCE_RO** requests that only read agents should be forced. **FORCE_RW** requests that only read-write agents should be forced. When the operand is not specified, forcing is done without examining the R/W status.

MAX_LOCKTIME n [FORCE_RO | FORCE_RW]

Requests that an agent be forced when it is in the lockwait state for more than **n** seconds.

The optional keyword **FORCE_RO** requests that only read agents should be forced. **FORCE_RW** requests that only read-write agents should be forced. When the operand is not specified, forcing is done without examining the R/W status.

MAX_LOGPAGES n

Requests that an agent be forced when it writes more than **n** pages on the DB2 log during a given LUW. High log space usage may be the result of executing UPDATE or DELETE statements that affect a large number of rows.

MAX_LUW_RESPONSE n

Requests that an agent be forced when the duration of its LUW exceeds **n** seconds. LUW response time is computed as (current_time - LUW_begin_time).

MAX_RESPONSE n

Requests that an agent be forced when the duration of its current statement exceeds **n** seconds. Statement response time is computed as (current_time - statement_begin_time).

MAX_ROWS n

Requests that an agent be forced when it fetches more than **n** table rows in a single SELECT statement.

INCLUDE {USER | VMUSER | PROGRAM} name

The INCLUDE statement defines the user and package names subjected to the coded restriction. A variable number of INCLUDE statements can follow the restriction definition.

An **INCLUDE USER** statement restricts the named DB2 userid.
An **INCLUDE VMUSER** statement restricts the named VM userid.
An **INCLUDE PROGRAM** restricts the named package (the package creator is not specified). The specified name may be generic by coding an * at the end of the name. (PROGRAM ABC* specifies all package names starting with ABC).

If an INCLUDE is coded, the restriction applies to the named users or programs only. If no INCLUDE is provided, all users and programs are included by default. The scope of an INCLUDE may be reduced by EXCLUDE statements that follow.

EXCLUDE {USER | VMUSER | PROGRAM} name

The EXCLUDE statement defines the user and package names **not** subjected to the coded restriction. A variable number of EXCLUDE statements can follow the restriction definition.

An **EXCLUDE USER** statement unrestricts the named DB2 userid.
An **EXCLUDE VMUSER** statement unrestricts the named VM userid.
An **EXCLUDE PROGRAM** unrestricts the named package (the package creator is not specified). The specified name may be generic by coding an * at the end of the name.

The userid **SQLDBA** is always excluded by default: it cannot be restricted.

Sample Restriction file

```
RESTRICT FROM 0800 TO 1700 (1)
MAX_IDLETIME 60 (2)
INCLUDE USER <userid_1>
MAX_IDLETIME 120 (3)
INCLUDE USER <userid_2>
MAX_LOCKHOLD 60 (4)
MAX_LOGPAGES 1000 (5)
MAX_BUFLOOK 10000 (6)
INCLUDE PROGRAM ARIISQL
EXCLUDE USER ADMUSR1
EXCLUDE USER ADMUSR2

RESTRICT FROM 1700 TO 2300 (7)
MAX_LOCKHOLD 600
MAX_IDLETIME 600
```

This sample control file definesl:

(1) Restriction period for the online window.
(2) Limit MAX_IDLETIME for <user_1>
(3) Limit MAX_IDLETIME for <user_2>
(4) Prevent users to lock a resource for more than 1 minute, if other users are waiting on that
     resource.
(5) Prevent users from writing more than 1000 pages to the DB2 log, without doing a COMMIT.
(6) Disallow ISQL users to perform more than 10000 buffer lookups during a single statement.
     The DB2/VM userid's ADMUSR1, ADMUSR2 and SQLDBA (by default) will not be restricted
     in their usage of ISQL.
(7) Restriction period for the off-shift window.

Cases 4 and 5 use the default INCLUDE and EXCLUDE lists, that is, all users except SQLDBA
are included.

5.3   Configuring the System Monitor

During SQL/MF initialization, monitoring control parameters are read from a CMS file named **SQLMFIN CTL.** This file is accessed using the standard CMS search order. It contains the following control statements :

BUFMON {YES | NO}

Specify NO if buffer monitoring should not be performed. **BUFMON YES** enables the buffer monitoring interval specified in the **INTERVAL** operand. Default is NO.

INTERVAL mi bi

Specify **mi** as the main monitoring interval **in seconds.** Specify **bi** as the buffer monitoring interval **in seconds**. If mi is not a multiple of bi, it will be set to BI*INT(MI/BI). Setting this parameter requires that **BUFMON YES** is specified too.

MONPERIOD from_hhmm to_hhmm

Specify the monitoring period as a start and end time in the format HHMM. If not specified, MONPERIOD is set to 0800 1700. If a monitoring period has been defined for the statement monitor in the SQLCS CONFIG dataset, system statistics are collected and made available to the system monitor, during this period only.

MONVMID xxxxx

Specify the name of the virtual machine that runs the system monitoring facility.

PRINT {YES | NO | LOG}

Specify **YES** to print all System Monitor tables and the Statement Monitor Logtable at the begin of a new monitoring session (day).[15] Specify **LOG** to print the statement monitor log table only, at the begin of a new monitoring session. The default is YES.

RETAIN n

Specify the number of monitoring sessions (days) to be kept in the monitoring tables. Session rows in excess of RETAIN are deleted at the end of the monitoring session. When the parameter is omitted, no automatic delete occurs.

---

[15]The DBSU file SQLMFPRT SYSIN performs the automatic print of the system monitor table rows for the previous day. You may alter this file to meet your needs. The statement monitor Log table is printed using the statement SQLCSPRT EXCPLOG. The log listing is ordered by descending I/O consumption. You may customize the listing by modifying the control file **EXCPLOGP CSP**.

SHOWDBSP hh:mm [dayname]

Specify the time as **hh:mm** when the system monitor should issue the SHOW DBSPACE command for all Dbspaces in the monitored databases. Specify **dayname** as the day-of-week when SHOW DBSPACE should be performed. If hh:mm is specified alone, the SHOW DBSPACE is done every day. If no SHOWDBSP statement is supplied at all, no SHOW DBSPACE commands are issued and the resulting information is not available in the system monitor tables.

To designate the databases to be monitored, two methods are available:

1. Code the following two statements for each database:

    -   **DATABASE xxxxxxx**   The name of the database.
    -   **VMID xxxxxx**   The name of the virtual machine running the database server.

2. Specify **DATABASE AUTODETECT** to select **all** monitored databases, that is, all databases in whose startup the EXEC SQLCSI command is present.

Sample SQLMFIN

```
MONPERIOD 0800 1700
MONVMID SQLSYSM
BUFMON YES
INTERVAL 300 300
PRINT YES
RETAIN 7
SHOWDBSP 02:00 SUNDAY
DATABASE AUTODETECT
```

Notes

1. The **SQLCSI** program should have been included in the startup procedure of all databases appearing in the SQLMFIN configuration file.

2. If SHOWDBSP is specified with a dayname, the rows in the DBSP_USE table will have the date stamp of that day. When consulting DBspace usage via the online interface, you must specify that date (or a period including that date). You may also have to adjust the SQLMFIN **RETAIN** parameter.

5.4   Configuring the SQL/MF print dataset

The SQLCSPRT program and the hardcopy function of the SQLCSU table editor write their output to the QSAM dataset **SQLMFPRT**, using CMS/OS simulation. The physical destination of that print dataset is determined by the FILEDEF command coded in the **SQLMFPFD** EXEC.

The distributed EXEC routes the SQLMFPRT dataset to your virtual printer. To route the print output to a CMS file, you can specify a FILEDEF such as:

FILEDEF SQLMFPRT DISK SQLMF LISTING (DISP MOD

The SQLMFPRT records written have a record length of 81 and start with an ASA control character.

5.5   Defining dynamic user aliases

Compiled DB2 applications sometimes wish to use dynamic SQL. Dynamically building an SQL SELECT during execution may provide end-users with more flexible and sophisticated data selection capabilities. Also dynamic SQL is sometimes faster than static SQL, because the dynamic statement predicate specifies actual selection values rather than host variables.

However, executing dynamic SQL requires that all users of the package have DB2 privileges on all tables used by the dynamic statement. In many cases this is undesirable, as installations grant end-users on DB2 packages, not on DB2 tables.

The dynamic alias facility offers a solution by replacing (during SQL PREPARE[16]) the executing DB2 userid with a userid that has table privileges (the alias). The alias userid's and the packages eligible for aliasing are defined in a CMS file named **<databasename> DYNALIAS**. The file should be accessible to the SQL/MF component SQLCSI that runs in the DB2 server.

Each record in the DYNALIAS file has the format:

PACKAGE <packagename> **ALIAS** <userid>

meaning: if <packagename> performs a dynamic SQL PREPARE, substitute the current DB2 userid with the userid specified as the ALIAS. The alias should have table authorities, either because it is the table creator or has DBA authority.

Alias substitution occurs during the dynamic PREPARE statement only. All static statements and the dynamic statements other than PREPARE, execute under the normal userid.

Note
Generic packagenames cannot be specified in the DYNALIAS file.

---

[16]Privileges are checked during PREPARE.

6 Initiating SQL/MF

6.1 Initiating Statement Monitoring in a Database Server

To enable monitoring in a DB2/VM database server, update the PROFILE EXEC and the DB2/VM startup procedure in the virtual machine running the database server.

- Include the CP and CMS commands necessary to access the minidisk or directory containing the SQL/MF software material.

- Before the **EXEC SQLSTART** command in the DB2 startup stream, insert the following command **EXEC SQLCSI DBNAME databasename**.

- If the DBNAME argument is omitted on the EXEC SQLCSI command, the databasename will be taken from the global CMS variable SQL/DS.DBNAME, which is kept by DB2 in the LASTING GLOBALV file. If, for some reason, the GLOBALV file has been removed from the A-disk, the GLOBALV DBNAME will not be found. SQLCSI then uses the server's VM-id as the databasename (after a warning). Therefore, it is recommended to specify the DBNAME argument on the EXEC SQLCSI.

- If it is not possible to alter the startup stream, the EXEC SQLCSI can be inserted in the profile exec or in any other exec called during database startup. In this case it should be ensured that the SQLCSI exec is executed each time the database is started (during database shutdown, the actions taken by SQLCSI are undone). You should follow this procedure when using IBM's Control Center to start the database.

- There is a potential SQL/MF performance improvement when the database server has VM privilege class C or E. This allows the monitor to extract VM/ESA-related data directly from the VM control blocks, rather than obtaining them using a CP command.

Note

- If you have created a dedicated database for the monitor tables (the LOG DATABASE in SQLCS CONFIG), there is no need to initiate monitoring in this database, since only monitor components connect this database.

- Do not start monitoring when operating in single user mode.

6.2   Initiating SQLCMDM

In the PROFILE EXEC of the virtual machine (suggested name is SQLCMDM) that will perform statement logging and the other functions provided by the SQLCS program, insert the command:

EXEC SQLCS
[RESTART {hh:mm|00:00}] [INTERVAL {n|10}] [NOPSTATS] [DSPOLL {n|5}] [REBIND]

Where:

RESTART
Specifies the time when the SQLCS program should perform its daily restart logic, which includes a CMS IPL. Default restart time is midnight.

INTERVAL
Defines the agent scan interval in seconds. The notification facility, the Governor facility and UAP initiation are executed at the scan interval. When the argument is omitted, a scan interval of 10 seconds is used. When a zero interval is specified, the notification, UAP and governor functions are not implemented.

DSPOLL
If a communications dataspace is used, the argument defines the dataspace polling interval in seconds. If omitted, polling for new requests is done every 5 seconds. DSPOLL must be less than INTERVAL. If DSPOLL is specified, it will also affect the INTERVAL parameter, as the polling logic determines when the scan interval expires: if INTERVAL is not a multiple of DSPOLL, INTERVAL will be rounded to the next higher multiple of DSPOLL.

NOPSTATS
When specified, the package statistics are not stored in the SQL/MF tables. Statement statistics are always stored. If you do not need the package statistics (which are the sum of all statement statistics for the executed programs), specify this option.

REBIND
When specified, the SQL/MF packages SQLCS and SQLCSXCP will be rebound at the every start of SQLCS. The argument is provided for compatibility with previous releases. The REBIND option is currently set by default.

When starting, SQLCS will attempt to connect the log database named in the corresponding statement of the SQLCS CONFIG file. If that connect fails, an error message is issued. Connect is re-attempted at the first request from a client. The program uses the SQLCS CONFIG file also to obtain the following configuration parameters: MODLEVEL, LOG LOCKTIME, LOG IDLETIME, LOG RETAIN, NOTIFY LOCKTIME, NOTIFY IDLETIME, NOTIFY RESPTIME.

6.3   Initiating an auxiliary SQLCMDM

The SQLCS program writes to the Monitor tables and, at its scan interval, performs the notification and the governor functions and calls the UAP's, if defined.

If the interval-related functions are highly time-critical and should not be delayed by possible waits induced by DB2/VM requests (such as a lockwait or a long checkpoint wait), an installation may decide to assign the table-related and the interval-related functions to 2 different virtual machines.

- The first machine (SQLCMDM) will perform logging into the SQLCS_LOG table and record the monitor statistics into the SQL_STMNTS table. This machine is defined as the **MONITOR** machine in SQLCS CONFIG. To disable the interval related functions, an **INTERVAL 0** argument is coded during SQLCS startup.

- The second machine will perform the notification, the governor and the UAP-call functions at the scan interval. Therefore, a non-zero **INTERVAL** must be specified at startup. Moreover, the machine should be defined as the **AUX_MONITOR** in SQLCS CONFIG (see page 49). As a result, it will not perform statement logging or statistics recording.

- Both machines should be setup in the same way and should have the same IUCV and DB2/VM privileges. The GRANT function of the SQL installation EXEC may be used to issue the DB2 grants.

6.4   Initiating the System Monitor

The system monitor runs in a dedicated virtual machine (suggested name is SQLSYSM) . The PROFILE EXEC of that machine should link to the SQL/MF product disk and execute the command **EXEC SQLMFM** to start the monitor.

Monitoring parameters are retrieved from the **SQLMFIN CTL** configuration file, as described on page 83.

To terminate the machine, use the **Shutdown** control command on the **SysMon** main menu (cf. page 169) or logoff the virtual machine manually.

7 Using SQL/MF: Monitor Data Items

7.1   Description

The monitor reporting functions described in the following chapters, show the following data columns:

Database
The name of the database where the statement executes.

VM_name
The VM name of the user executing the statement.[17]

SQL_name
The DB2/VM name of the user executing the statement.

Agent
The number of the DB2/VM agent.

Agent-status
The execution status of the DB2/VM agent. See table on page 97.

LUWid
The DB2/VM LUW number.

Location
For VM/ESA users: the VTAM terminalname (or "DSC" for a disconnected user). For CICS users: the CICS terminalnumber prefixed with "CICS". For terminal-independent applications the CICS taskid is shown, prefixed with "TID".

CICStid
The CICS task number.

Package
The name of the package containing the statement

Section
The section number of the package containing the statement. SQL/MF maintains a dummy package section **-1** to record the cost involved in package initialization (package loading, privilege checking and eventual automatic reprepping).

Plan
1 for the first table accessed by the statement, 2 for the next table and so on. (Access to an internal DBspace is also considered as a plan).

Statement
The type of SQL statement executed. See table on page 98.

Isolation

---

[17]If an alternate VMID is in effect (via a Diagnose D4), the alternate ID is shown, unless the SHOW_ALTID option has been coded in SQLCS CONFIG, in which case the primary VMID is always shown, if the protocol is DRDA. With Diag D4 in effect under PROTOCOL SQLDS, the primary VMID cannot be shown, as it is not present in the DB2/VM control blocks.

The isolation level of the statement as **CS** (cursor stability), **RR** (repeatable read) or **UR** (uncommitted read).

Blocked
The blocking attribute of the statement (**Y**es or **N**o). For non-cursor statements blocking is always set to Y.

Startdate
The start date of the statement.

Stopdate
The stop date of the statement.

Starttime
The start time of the statement.

Stoptime
The stop time of the statement, that is, the starttime of the next statement..

Elapsed
The response time of the statement or statement section as hh:mm:ss. Note that for cursor related statements, starttime is the starttime of the section, that is, the time the cursor was opened. Elapsed time is the time elapsed since the opening of the cursor.

CPUtime
The total CPU usage time for executing the statement, in microseconds. The CPU time shown is equivalent to the VM TOTCPU value.

UserCPU
CPU time used by the DB2 client virtual machine. Available only when the SQLCSI program in the database server has VM privilege class E.

Lockwait
The total time spent in lock wait, in milliseconds.

Iowait
The total time, in milliseconds, spent in BLOCKIO wait and, if the dataspaces feature has been installed, the time spent in dataspaces page fault wait.

Commwait
The total time spent in communication wait, in microseconds.

CHKPtime
The total time spent in checkpoint wait, in milliseconds.

RDScall
The number of calls to the Relational Subsystem.

DBSScall
The number of calls to the Database Subsystem.

DispCall
The number of dispatcher calls during the statement. (A dispatcher call is made whenever a DB2/VM agent must wait on an *event*).

Nrows
The number of table rows fetched, inserted, deleted or updated by the statement.[18]

Tot_rows
The total number of table rows processed by the entire package.

Buflooks
The number of buffer lookups performed.

Waitlock
The number of times a lock request resulted in wait.

Nr_Locks
The number of locks currently held by the agent.

Escalate
The number of times a lock request resulted in lock escalation.

Pagread
The number of page reads during the statement, that is, the number of times a request for data could not be satisfied from the buffer pool. The statistic includes reads from the dataspaces if applicable.

Pagwrite
The number of page writes during the statement. The statistic includes writes to the dataspaces if applicable. [19]

Logread
The number of reads to the DB2/VM log during the statement.

Logwrite
The number of writes to the DB2/VM log during the statement.

---

[18]For blocked, cursor-based fetch statements, the statistic shows the number of rows in the block passed by the database server to the Resource Adapter in the client's machine. Although available due to the statement's search conditions, all these rows are not necessarily processed by the application program.

[19]During output statements such as UPDATE or INSERT, the write request changes the state of the corresponding buffer pool page to "modified". An I/O request is not immediately performed. Modified buffer pool pages are written to DASD at the end of the LUW or when the buffer pool page is re-allocated to another agent. In the latter case, the page write is accounted to that agent by the monitor.

Dirbufl
The number of directory buffer lookups performed.

Dirread
The number of directory page reads during the statement. The statistic includes reads from the directory dataspace if applicable.

Dirwrite
The number of directory page writes during the statement. The statistic includes writes to the directory dataspace if applicable.

IntDBsp
The number the number of buffer lookups in internal DBspaces during the statement.

DSfaults
The number of dataspace pagefaults during the statement, if DB2/VM dataspaces have been installed. A pagefault occurs when a requested dataspace page is not in main storage and must be retrieved from DASD (by CP). The statistic is not available when no dataspaces have been installed.

*BlockIO
The number of VM IUCV BlockIO requests submitted by the database manager to CP during the statement. Since a single BlockIO request usually reads or writes several pages at once, the number will normally be lower than PAGREAD+PAGWRITE. The statistic is not available when no dataspaces have been installed.

Tot_Elaps
Total elapsed time in the package, as the sum of CPU time and all wait times.

Max_Elaps
Maximum elapsed time in the package during the STAT_FREQ period.

Access
The name of the index used to access the first table in the statement or "No index access noted" if no index is being used. For indexed access, the first 40 characters of the index column definition are shown.

Selective_Index
A YES/NO flag indicating the selectivity of the index used.

SQL Statement
The text of the SQL statement executed with the host variable names replaced with the host variable contents.. For static statements, the statement text is obtained from the SQLCS_SQL_STMNTS table or by unloading the package if it is not in the above table.

7.2   Agent status description

Label	Meaning
RUNNING	agent is processing
READYRUN	agent is ready for processing
IOWAIT	agent is waiting for completion of an I/O operation or a dataspace pagefault
LOCKWAIT	agent is waiting on a locked resource
COMMWAIT	agent is waiting for next SQL statement from client
CHKPWAIT	agent is waiting for completion of checkpoint
PBUFWAIT	agent is waiting for a page buffer in the buffer pool
DBUFWAIT	agent is waiting for a directory buffer
DSPFWAIT	agent is waiting for completion of a dataspace page fault (VMDSS only)
SLDWAIT	agent is waiting for a Save List Definition block (VMDSS only)

7.3 SQL statement types description

Label	Meaning
CLOSE	close a cursor
Statement	"AUXcall" (single SELECT, INSERT and UPDATE)
COMMIT	commit work
CONNECT	connect a database
EXECUTE	perform an EXECUTE IMMEDIATE statement
EXT_EXEC	execute an extended dynamic statement
FETCH	fetch using a cursor
OPEN	open a cursor
OPERCMND	execute a DB2/VM operator command
PUT	insert using a cursor
PREPARE	prepare a dynamic statement
PREPCALL	perform a package prep
PROGINIT	load (or auto-prep) a package
PROGLOAD	perform an SQLDBSU package reload
ROLLBACK	perform rollback

8 Using SQL/MF: General considerations

8.1 Authorizing access to the User Interface

Access to the SQL/MF user interface is controlled by means of the **SQLMF ACF** file, which **MUST** reside on the SQL/MF product disk. The authorization component inspects the authorization file on this disk only and does not use the CMS search chain, to avoid that users create an authorization file on their own minidisk.

The default SQLMF ACF delivered with SQL/MF provides public access to all components of the user interface. If this is not your intention, modify the file to suit your needs.

8.1.1 Authorization scheme

The authorization component uses following concepts:

Application
Each SQL/MF component that can be invoked individually is considered an SQL/MF application.

Group
Authorization to execute a named SQL/MF application is granted to a named group.

User
To authorize a user on a specific SQL/MF application, the user must be connected to a group that has the required authorization.

Private access
Authorization distinguishes between private and public access. Users with private access (e.g. developers) have access only to the monitor data or table rows that pertain to their userid. Users with public access (e.g. a DBA) have unlimited access to the monitor data.

8.1.2 Defined SQL/MF Applications

Application name	Application function	Public access applicable
COMMAND_ANALYSIS	Performs an SQL/CA analysis for an SQL statement.	No
COMMAND_LIST	Shows the list of all SQL statements in one or more databases.	Yes
COMMAND_RECORDER	Accesses the Statement Recorder file.	Yes
HOST_STATEMENTS	Issues DB2, CP, CMS and MONITOR statements.	No
LOCK_INFO	Shows the locks held and wanted by an agent.	No
LOCK_RECORDER	Accesses the Lock Recorder file.	Yes
LOG_TABLE	Accesses the Exception Log table.	Yes
OBJECT_LISTS	Provides access to the DB2 catalog tables.	No
RUNSTATS	Shows DB2 session statistics by package section.	No
SQLGRAPH	Graphically shows global DB2 session statistics.	No
STATEMENTS_TABLE	Accesses the Statements table.	Yes
SYSMON_TABLES	Accesses the System Monitor tables.	No

8.1.3   Authorization statements

8.1.3.1   GRANT statement

Purpose

Grants access on a named SQL/MF application to a named group.

Syntax

GRANT { <u>PRIVATE</u> | PUBLIC } *application* TO *group*

> **application**
> The name of an SQL/MF application listed in the above table.
>
> **group**
> The name of the group that must be granted on the application. The length of the groupname should not exceed 32 characters. If group is specified as **PUBLIC**, the application is available to all users. A CONNECT statement is not necessary in this case.
>
> **PRIVATE**
> When accessing monitor data, users can inspect only the data produced by their own SQLID. This is the default option.
>
> **PUBLIC**
> Users can inspect all monitor data.

8.1.3.2   CONNECT statement

Purpose

Connects a named VM userid to a named authorization group.

Syntax

CONNECT *userid* TO *group*

> **userid**   The VM userid.
> **group**   An authorization group, previously defined in SQLMF ACF.

8.1.3.3   Comment

A line starting with an asterisk is considered as a comment line.

8.1.3.4   Names

All groupnames and userids should have distinct values. A group cannot have the same name as a connected user.

8.1.4 Automated SQL/MF table GRANTS

The DB2 grants on the SQL/MF tables SQLCS_LOG and SQLCS_SQL_STMNTS will be issued automatically, using the authorization file, by the Statement Monitor SQLCMDM at its daily restart. SQLMCMDM will revoke all existing grants on the above tables and then issue a GRANT SELECT for those users (VM_id's) that appear in the SQLMF ACF table.

Please note that only the grants issued by SQLCMDM itself are revoked. Grants issued by other grantor are not affected and should be revoked manually, if needed.

8.1.5 Distributed authorization file

```
GRANT COMMAND_ANALYSIS TO PUBLIC
GRANT PUBLIC COMMAND_LIST TO PUBLIC
GRANT PUBLIC COMMAND_RECORDER TO PUBLIC
GRANT HOST_COMMANDS TO PUBLIC
GRANT LOCK_INFO TO PUBLIC
GRANT PUBLIC LOCK_RECORDER TO PUBLIC
GRANT PUBLIC LOG_TABLE TO PUBLIC
GRANT OBJECT_LISTS TO PUBLIC
GRANT RUNSTATS TO PUBLIC
GRANT SQLGRAPH TO PUBLIC
GRANT PUBLIC STATEMENTS_TABLE TO PUBLIC
GRANT SYSMON_TABLES TO PUBLIC
```

The default authorization file is applied, the first time you start the SQLCMDM component after SQL/MF installation.

8.1.6 Sample authorization file

```
*
*   The DBA_USER group has unlimited access
*
GRANT COMMAND_ANALYSIS TO DBA_USER
GRANT PUBLIC COMMAND_LIST TO DBA_USER
GRANT PUBLIC COMMAND_RECORDER TO DBA_USER
GRANT HOST_COMMANDS TO DBA_USER
GRANT LOCK_INFO TO DBA_USER
GRANT PUBLIC LOCK_RECORDER TO DBA_USER
GRANT PUBLIC LOG_TABLE TO DBA_USER
GRANT OBJECT_LISTS TO DBA_USER
GRANT RUNSTATS TO DBA_USER
GRANT SQLGRAPH TO DBA_USER
GRANT PUBLIC STATEMENTS_TABLE TO DBA_USER
GRANT SYSMON_TABLES TO DBA_USER
*
*   The DEVELOPER group has selective access
*
GRANT COMMAND_ANALYSIS TO DEVELOPER
GRANT PUBLIC COMMAND_LIST TO DEVELOPER
GRANT PRIVATE COMMAND_RECORDER TO DEVELOPER
GRANT PRIVATE LOCK_RECORDER TO DEVELOPER
GRANT PRIVATE LOG_TABLE TO DEVELOPER
GRANT PRIVATE STATEMENTS_TABLE TO DEVELOPER
* Users connected as DBA_USER
CONNECT USERA TO DBA_USER
CONNECT USERB TO DBA_USER
*   Users connected as DEVELOPER
CONNECT DEV1 TO DEVELOPER
CONNECT DEV2 TO DEVELOPER
CONNECT DEV3 TO DEVELOPER
```

8.2   Task-oriented Description

8.2.1 Examining running SQL statements

Type **SQLCSU** at the CMS prompt. This gives you a list of all statements executing in all monitored databases. By default, the list is refreshed every 3 seconds and ordered by database and statement resource usage. Therefore, the statements at the top of the list usually require your attention. Also note the **SQL Cost** column in the list. It is computed as TOTIO + (DBSSCALL÷3).

If you want to examine a statement more closely, press ENTER. Position the cursor on the screen line displaying the statement and press PF4 (the **Detail** key). On the detail screen, you can perform a number of functions (such as **Analyze**) by PFkey. Press PF3 to return from statement detail to the running statement list and press the **Resume** key.

8.2.2 Examining running SQL statements graphically

Start **SQLCSU**, press ENTER and PF6 (**SQLpulse**). This provides a bargraph representing the resource consumption by all programs running in all monitored databases. By default, the bargraph shows the number of DB2/VM buffer lookups for each package (you can also request a bargraph by CPU or I/O usage). The bargraph is refreshed every 3 seconds by default.

If you want to examine a package more closely, press ENTER. Position the cursor on the bargraph line displaying the package and press PF4 (the **Detail** key). This will continuously display the monitored data for the statement in bargraph format. Again, press ENTER to interrupt the display. Use the PFkey interface to perform functions (such as **analyze**) for the statement shown.

8.2.3 Examining the Run Statistics for a DB2/VM server

Start **SQLCSU**, press ENTER and PF9 (**RunStats)**.

The RunStats program will ask the name of the database server to be inspected. After you enter the name, the RunStats utility shows the resource consumption during the DB2/VM session summarized by DB2/VM userid. Use PF5 to summarize the consumption by programname.

Using the **Detail** function you can examine the consumption for a given user or program. To do so, place the cursor on the user- or programname and press the PF4 key.

- In a user list, you will receive the list of programs executed by that user.
- In a program list, you will receive the list of package sections.
- When in a program section list, you can use the Detail key again to obtain the complete statistics, the statement text (if a prepped statement) and the index used for a given section.

All the above lists are ordered descending on the current resource consumption counter. By default, this is the number of buffer lookups. But you can also order by CPU usage or by the I/O load, using the PFkey interface.

8.2.4 Examining the System Statistics for a DB2/VM server

Start **SQLCSU**, press ENTER and PF11 (**SysMon)**. On the next panel, choose function number 1 (*Show SQL Counters*), type the name of the server and eventually a date / time period you want to examine.

On the next panel, choose function number 5 (*Counter Summary*). For each monitor sample you can see the number of RDS and DBSS calls during the sample, the number of LUWs and I/O requests and the hit ratio for the directory and page buffers.

The System Monitor interface has other functions for inspecting the usage of the storage pools, the DB2/VM log, the DB2/VM connections and more.

To return to the SQLCSU screen, press PF3 repeatedly.

8.2.5 Examining the status of a DB2/VM server graphically

Start **SQLCSU**, press ENTER and PF6 (**SQLgraph)**. On the next panel, confirm or modify the name of the database to inspect. SQLGRAPH then shows a bargraph for the resource consumption during the last 60 minutes. Press ENTER to enable the PFkey interface. Use PF7 and PF8 to browse the graph backwards and forwards. PFkeys are also available to alter the format of the bargraph, to switch between numeric/graphic mode etc.

8.2.6 Examining statement usage statistics

During execution, the monitor maintains the execution frequency, the access path and the latest resource consumption for all SQL statements executed in the SQLCS_SQL_STMNTS table. You can inspect this table as follows:

Start **SQLCSU**, press ENTER and PF10 (**LogTable)**. This provides the **SQL/MF Report Menu**. Choose "Statement Statistics". On the next panel:

-   Using PF4, invoke the **SCUSE1** report to show the statement statistics for a package (or for several packages if you use a generic package name).

-   Using PF4, invoke the **SCUTIME** report to show the statistics for the packages that executed during a specified date and time period. The report is sorted by descending DB2/VM buffer lookup, so that the statements with the highest cost appear at the top of the list.

-   Using PF4, invoke the **PCOST** report to show the statistics for all packages executed in the current monitoring session. The report is ordered by descending package cost and within each package, by descending statement cost. (The cost value is computed as **TOTIO + (DBSSCALL ÷ 3)**).

8.2.7 Examining package usage statistics

During execution, the monitor maintains the total resource consumption for all static (prepped) packages. You can inspect this table as follows:

Start **SQLCSU**, press ENTER and PF10 (**LogTable)**. This provides the **SQL/MF Report Menu**. Choose "Package Statistics". On the next panel:

-   Using PF4, invoke the **PPROG** report to show the **total** statistics for a package (or for several packages if you use a generic package name).

-   Using PF4, invoke the **PDETAVG** report to show the **average** package statistics i.e. total_statistics ÷ LUWS column

-   Using PF4, invoke the **PUSER** report to show the package statistics summarized by invoking DB2/VM Userid.

Note   The package execution statistics are suitable for **accounting** purposes.

8.2.8 Examining the statement exception Log

You can specify a number of exception values in the SQLCS CONFIG file (see page 45), and cause an **exception log** event for SQL statements exceeding the exception value. For instance, you can request to log statements that perform more than N buffer lookups. To inspect the exception log, proceed as follows:

Start **SQLCSU**, press ENTER and PF10 (**LogTable)**. This provides the **SQL/MF Report Menu**. Choose "Statement Exception Log". On the next panel:

- Using PF4, invoke the **LCHRONO** report to show the statement exceptions in chronological order.

- Using PF4, invoke the **LREASON** report to show the statement exceptions by logging reason.

- The remaining **L** reports show the statement exceptions in different order.

8.2.9 Examining the benchmark Log

When requested, the Monitoring Facility benchmarks a package by storing all the statements it executes in the LOG table. To inspect the benchmarking results, proceed as follows:

Start **SQLCSU**, press ENTER and PF10 (**LogTable)**. This provides the **SQL/MF Report Menu**. Choose "Benchmark Reports". On the next panel:

- Using PF4, invoke the **BCHRONO** report to show the benchmarked statements in chronological order.

- The remaining **B** reports allow you to inspect the benchmark log in a different order.

Note   Benchmarking shows the **actual resource consumption** by the benchmarked statements. Comparable statement results appear as **resource consumption** in the SQLCS_SQL_STMNTS table. In addition, the latter table contains the statement execution frequency, which cannot be seen in the benchmarking report. To inspect the SQLCS_SQL_STMNTS table, refer to *Examining statement usage statistics* above.

8.2.10   Printing monitor reports

- When displaying one of the above logging or statement statistic reports, the **Hardcopy** function of the editor can be used to print the entire report or one report page.

- Alternatively, you can use the **SQLCSPRT** program for printing purposes. For a complete description of this program, please refer to page 191 (and following).

8.2.11   Executing Host Commands

The Monitoring Facility allows you to execute a DB2/VM, a CP, a CMS or a MONITOR command in one of the monitored databases (provided you have an IUCV link to this database).

To do so, start **SQLCSU**, press ENTER and PF12 (**HostCmnd)**. On the next panel, type the name of the target database and the text of the command, prefixed with **CP** (if a CP command), **CMS** (if a CMS command) or **MONITOR** (if an SQL/MF command).

For more details, please refer to page 195.

8.3   Generalized hardcopy function

By pressing PA3 in any SQL/MF output list, the current screen can be hardcopied to the disk file **SQLMF HARDCOPY A**. The current hardcopy is always appended to this file. You are responsible for file cleanup.

9 Using SQL/MF: Inspecting runtime data

9.1 Inspecting running SQL statements using SQLCSU

Preliminary notes:

1. The SQLCSU program directly reads the Monitor shared segment SQLCSMDS to display he monitor data. Therefore, the machine running SQLCSU should not execute in 370 mode, when the shared segment has been defined above the 16 Mb line.

2. The SQLCSU program and the functions callable from SQLCSU, heavily use virtual storage. For a smooth operation of these interface programs, a virtual storage size of at least 8 Mb is recommended in the virtual machine invoking them.

9.1.1 Statement List

The SQLCSU program allows you to inspect all running SQL statements and to take actions on any of them. The syntax of the SQLCSU command is as follows:

SQLCSU [interval] [database]

where:

interval
is the statement list refresh interval in seconds (default is 3)

database
is the name of the database for which running statements must be shown; if omitted all statements running in all databases are shown.

The program continuously shows the list of all running SQL statements in all monitored databases on the screen, at the specified display interval. The statement list is ordered by databasename and descending SQL cost, so that the statements consuming most resources in each database, appear at the top of the list. The list shows all users that are in logical unit of work, even if they are not effectively running SQL. The SQL_Cost item shown is computed in the same way as the DB2/VM Query Cost Estimate, namely: total I/O requests + (number of DBSS_calls / 3). Statement lines with a cost greater than 1000 will be highlighted.

```
      SQL/Statement Monitor                     © Software Product Research
--------------------------------------------------------------------------
Database   VM_name   SQL_name  Started   Elapsed  Package  Type    Stat SQL_Cost
--------------------------------------------------------------------------
SQLD2      VSESP     985       10:41:23 00:02:17 DF252    Fetch   Comm       10
TSTADB     CMS14     SQLDBA    10:43:39 00:00:01 CSTEST   Open    DSPF       27




                                                                          _
--------------------------------------------------------------------------
PF  1 Help      2 Resume    3 Quit      4 Detail    5 SQLgraph   6 SQLpulse
    7           8           9 RunStats 10 LogTable 11 SysMon    12 HostCmnd
```

9.1.2 List Statements

To interrupt the running statement list, press the ENTER key. The bottom lines of the screen now show the PFkey labels that you can use to invoke a list statement. The Detail and SQLGraph functions operate on the statement indicated with the cursor (the current statement). Therefore, before pressing the corresponding PF key, you should place the cursor on the screen line showing the statement.

PF1   Help
Displays the SQLCSU help file. The help file provides for column help or for procedural help. Column help provides a short description of each data field displayed.

PF2   Resume
Resumes interval driven display of the running statement list.

PF3   Quit
Terminates the SQLCSU program.

PF4   Detail
Shows full details of the current statement. See the paragraph Statement Detail on page 115.

PF5   SQLgraph
Invokes the SQLGRAPH program for the database named the current statement. If the cursor is not on a statement, SQLGRAPH is executed for the default database. For a complete description of the SQLGRAPH program, please refer to page 123.

PF6   SQLpulse
Invokes the SQLPULSE program. For a complete description of the SQLPULSE program, please refer to page 132.

PF7   Page <<<
Displays the previous page of the running statement list, if the list has more than one screen page.

PF8   Page >>>
Displays the next page of the running statement list, if the list has more than one screen page.

PF9   RunStats
Invokes the RunStats program to show the Session Run Statistics for a designated database. For a description of the RunStats utility, please refer to page 127.

PF10   Log Table
Invokes the SQLCSLST table editor to display the SQL/MF Log Tables. See page 135 and following for more details.

PF11   System Monitor
Invokes the SQLMF program to display the System Monitor tables. See page 167.

PF12   Host Command
This function allows you to execute DB2/VM, CMS, CP and MONITOR commands interactively. See the paragraph Host Commands on page 195.

9.1.3 Statement Detail

```
Database TSTADB     VM_name  TSTAST1   SQL_name TSTAST1   Location T25TT262
Agent    3          Status   Running   Package  SQLCVRFY  Section  1
Command  Fetch      Isolatn  CS        LUW_id   8172387   Protocol DB2/VM
--------------------------------------------------------------------------
Started  18:46:39   Curtime  18:46:59  Elapsed  00.00.20  Rowcount 32123
CPUtime  2.914      Locktime 0.000     IOtime   0.061     Commtime 1.494
RDScall  354        DBSScall 32123     DispCall 713       Buflooks 32953
Pagread  831        Pagwrite 0         Logread  0         Logwrite 0
Dirbufl  787        Dirread  0         Dirwrite 0         Int_DBSP 0
DSfaults 10         *BlockIO 0         Nr_Locks 10        Escalate 0
--------------------------------------------------------------------------
SELECT ITEM,"SQL/DS HELP" FROM SQLDBA.SYSTEXT2




--------------------------------------------------------------------------
No index used during access
_____
PF  1 Help      2 Resume    3 Quit      4 Analyze   5 ProgStat  6 LockInfo
    7 Page <<<  8 Page >>>  9          10 CmndText 11 Force    12 ObjList
```

The statement detail function shows for a given statement all monitor data columns, described on page 93.

Note

When a sequence of identical statements is being performed (a sequence of cursor based fetches for example), the statistics reflect the consumption of all statements in the sequence. Moreover, cursor related fetch and put statements are considered identical for the purpose of monitoring.

9.1.4 Statement Detail functions

When the statement detail screen is shown, following actions may be taken on the statement, by pressing the corresponding PFkey.

PF1   Help

Displays the help file for the statement detail function.

PF2   Resume

Resumes display of the running statement list.

PF3   Quit

Terminates the statement detail function and returns to the statement list, without resuming it.

PF4   Analyze

Invokes SQL/Command Analysis for the current statement, if our SQL/CA product has been installed on your system.

PF5   Package Statistics

Shows, for the current statement, the statistics recorded for the entire package in the SQL_STMNTS table. These statistics show:

- the resource consumption for each statement in the package
- the primary access path noted for each statement
- the SQL statement text for each statement in the package

PF6   Lock Info

When the agent is in the lock wait state, PF6 displays the Lock Graph for the agent. (The lock graph shows the lock hierarchy between all agents involved in a lock).

When the agent is not in lockwait, PF6 shows the locks currently held by the agent, by issuing the DB2/VM command Show Lock User.

In both cases, the function shows the DBspace names for all DBspace numbers appearing in the command output (using an IUCV transaction with the Statement Monitor machine).

PF7   Previous Page

Shows the detail screen for the previous agent in the running agent list.

PF8   Next Page

Shows the detail screen for the next agent in the running agent list.

PF9   Graphic

Shows agent detail in graphical format, by calling the SQLPULSE function described on page 132.

PF10   CmndText

Displays the full and formatted SQL statement text. Long statements may take several pages to display. Use PF7 and PF8 to browse through the pages. Press PF3 to terminate the text display.

PF11   Force

Issues the DB2/VM FORCE ROLLBACK command for the running agent. Before forcing, the function will send the confirmation prompt Continue force? to which you must reply Y for forcing to proceed.

When you issue your force request, the agent may actually no longer be active, or may be executing another package. Therefore, your force request is rejected:

- when the agent is no longer in work
- when another DB2 or VM userid runs on the same agent structure
- when the agent executes a package different from the one appearing on the detail screen

PF12   Objects

The function shows the DB2/VM catalog information for the current Dbspace, table, columns or index, as described on page 144. The Objects function can also invoke the Catalog Navigator, which is described on page 146.

9.1.5 Host Commands

This function lets you execute a DB2/VM, a CP, a CMS or a MONITOR command in a designated database server. To execute the command, you should have the necessary IUCV privileges to connect to the server machine.

On the host command screen specify:

the virtual machine name (not the databasename) of the desired database server
the text of the command, prefixed by the word

- **CP** if a VM command is issued
- **CMS** if a CMS command is issued
- **MONITOR** if an SQL/MF MONITOR command is issued

On completion of the command, its output is displayed on your terminal in CMS full screen mode. Use the CMS full screen PFkeys to examine the command output. Press PF3 to terminate the full screen session. Press PF3 to quit the Host Command function and to return to the running statement list.

For a description of the host command interface, please read page 195. For a description of the MONITOR command, goto page 200.

9.1.6 Configuring SQLCSU

By providing an SQLCSU CONFIG dataset, you can modify the operation of SQLCSU, in particular, the presentation of the running agent list.

When starting, SQLCSU looks for an SQLCSU CONFIG dataset, using the standard CMS search order. Following statements can be coded in the CONFIG file:

DATABASE <name>
Includes the named database only in the running agent list. Up to 16 databases can be specified.

INTERVAL <n>
Defines the running agent list refreshment interval, in seconds.

NOCICSLINK
Suppresses the display of the initial CICS links (package ARIRSQL) which are activated at CICS startup.

ORDER BY {SQLID | VMID | ELAPSED | START | COST | PACKAGE} {ASC | DESC}
Modifies the sort order of the agent list (the default order is descending SQL cost). The specified item defines the secondary sort column: the primary sort column is always the database name.

SQLID: sort on DB2/VM username
VMID: sort on VM username
ELAPSED: sort on statement elapsed time
COST: sort on SQL cost, that is, on TOTIO + (DBSSCALL ÷ 3)
PACKAGE: sort on package name

SHOW LOCATION
Shows the CICS terminal-id in the agent list, instead of the DB2/VM-id.

SHOW LUWTIME
Shows the begintime of the LUW, instead of the statement begintime.

9.2   Using SQLGRAPH

The SQLGRAPH function provides a graphical representation of activity within a given DB2/VM Database server, by showing a bar graph with the following counters:

- number of RDS calls
- number of buffer lookups
- number of LUWs
- total number of I/O requests

When started, SQLGraph continually displays the DB2/VM activity counters for the last hour, with a display resolution of one minute.

To interrupt the automatic display, the ENTER key must be pressed, which will show the PFkey function definitions at the bottom of the screen.

```
                                                              Database
                                                              SQLDBA


    87890          █
    81140          █            █
    74390          █            █            █
    67640          █            █            █       █
    60890         ███          ███          █       █
    54140         ███         █████        ███     ███
    47390        █████        █████        ███     ███
    40640       ███████      ███████      █████   █████
    33890       ███████      ███████      █████   █████
    27140       ███████      ███████      █████   █████
    20390       ███████      ███████      █████   █████
    13640      █████████████████████     ███████████████
     6890     ████████████████████████████████████████████
          06:00-----08:30-----11:00-----13:30-----16:00-----18:30-----



 PF 1=Help        2=Resume     3=Quit       4=Numbers   5=Period    6=
    7=Back        8=Forward    9=Server    10=Save     11=Load     12=Columns
```
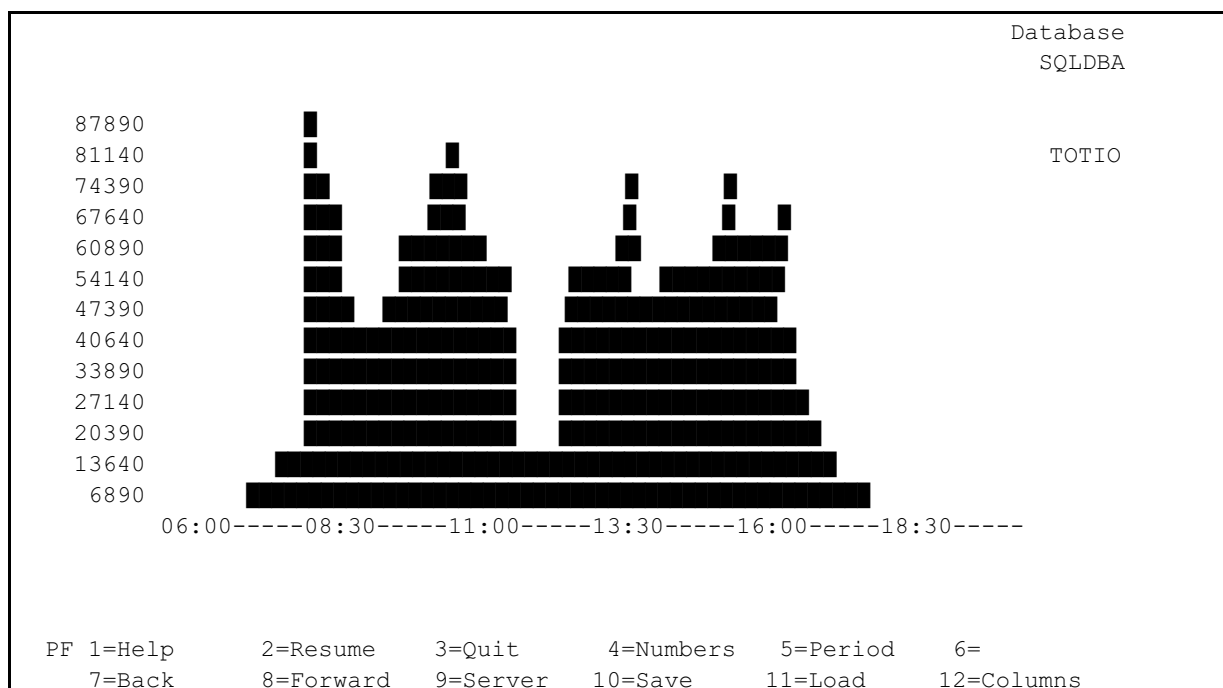
9.2.1 Invocation

The SQLGRAPH function is invoked from SQLCSU through the SQLgraph function key PF5. If the cursor is on an active statement, the program shows the activity within that Database. Else, the activity in the default Database is shown.

9.2.2 SQLGRAPH Functions

SQLGRAPH provides a PFkey driven interface to perform the following functions:

PF1
Displays the SQLGRAPH help file.

PF2
Returns to the automatic output display mode.

PF3
Terminates the SQLGRAPH utility and returns to SQLCSU.

PF4
Toggles between graphical and numeric output mode.
The graphical mode shows the DB2/VM counters as a bar graph.
The numeric mode shows the DB2/VM counters as numbers, with 20 periods on one screen.

PF5
Toggles between the Period and the Session display modes.
The "period" mode shows the DB2/VM activity for one hour, with a display resolution of one minute.
The "session" mode shows the DB2/VM activity for an entire session, from 6 AM to 8 PM, with a resolution of 15 minutes.

PF6
Shows the DB2/VM Buffer Pool usage graph. This function is described on page 126 of this manual.

PF7
Displays the previous period, that is, the previous hour in graphical mode, or the previous 20 minutes in numeric mode

PF8
Displays the next period, that is, the next hour in graphical mode, or the next 20 minutes in numeric mode

PF9
Inspects another database server, by accepting the VM userid of that database server.

PF10
Saves the current graphical counters to a CMS file, with the following default name:

CMS filename = G$yymmdd   (Where yymmdd is the current date)
CMS filetype = databasename

Before saving, you have the opportunity to modify the default CMS filename and filetype.

PF11
Loads and shows the graphical counters from a previously saved CMS file. A panel is displayed to enter the CMS name of the file to be loaded.

PF12
Selects the DB2/VM counters to be shown.
By default, all counters (RDSCALL, BUFLOOK, LUW, TOTIO) are displayed.
The PF12 key function displays a panel where one or more counters can be enabled (by entering Y) or disabled (by entering N).

The selections made on this panel are saved as a CMS file and remain in effect for all subsequent SQLGRAPH sessions, until modified by another PF12 request.

Notes:

1. A colour 3270 terminal is required to display the bar graph.

2. The absolute values of the DB2/VM counters are reduced to fit on the 20 screen positions that constitute the y-axis. A large reduction factor may cause loss of precision when displaying small sampled values. To view the real counter values, the numeric mode can be toggled.

3. For a given sampling interval, the individual counters are superposed with different colours. If different counters are within the same range, one or more counters may be invisible. To view all counter values, the numeric mode can be toggled.

9.2.3 Buffer Pool usage graph

The buffer pool usage function shows how individual DBspaces are making use of the DB2/VM buffer pool.

The function counts the number of page and directory buffers allocated to each DBspace and displays the 19 DBspaces that own the highest number of buffers. The first line in the report shows the number of unused buffers.

The number of used buffers is expressed as a percentage of the total number of buffers in use.

The panel shows the page buffer usage to the left and the directory buffer usage to the right of the screen.

A special entry is made for the buffers owned by the DB2/VM log and the internal DBspaces (if it belongs to the high-use entries).

Following functions key are provided on the buffer pool usage panel:

PF2
Refreshes the graphical output for the current database.

PF3
Quits the function and returns to the main SQLGRAPH panel.

PF4
Allows to get the buffer pool usage for another database server.

9.3   Invoking the RunStats Function

In each database server, SQL/MF keeps a number of statistics for each package section, that is, for each SQL statement executed. These statistics are maintained for the duration of the entire DB2/VM session and can be saved to a DB2 table using the SQLCS CONFIG RETAIN_RUNSTATS option.

Following statistical counts are provided:

- the number of times a section has been executed
- the section's CPU usage expressed in milliseconds
- the number of buffer lookups performed by the section
- the number of I/O requests performed by the section

The RunStats function lets you examine these statistics in an hierarchical manner. It is invoked by pressing PF9 in the SQLCSU program.

At entry into the function, specify the name of the database you want to inspect. If the RETAIN_RUNSTATS option has been enabled, the function request screen allows you to enter the following additional selection arguments:

Show Runstats by period
Enter Y if you wish to consult the saved runstats table. Enter N if you wish to consult the current session runstats.

Packagename
Specify a package name to be examined. If left blank, all packages are shown. Package names are generic by default, that is, all packages beginning with the specified string will be included.

Date_from
Enter the start date for selection as yyyy-mm-dd.

Date_to
Enter the end date for selection as yyyy-mm-dd.

Time_from
Enter the start time for selection as hh:mm:ss.

Time_to
Enter the end time for selection as hh:mm:ss.

Initially, RunStats graphically shows the resource consumption in all sections summarized by DB2/VM userid. Using PF5, you can request a resource consumption report, summarized by packagename.

On these reports, use the Detail function to explore the consumption by a given user or package. To do so, place the cursor on the user- or packagename and press the PF4 **Detail** key.
- In a user list, you receive the consumption in the packages executed by that user.
- In a program list, you receive the consumption in the package sections, i.e. the individual SQL statements
- In a program section list, you obtain the complete statistics, the statement text (if a prepped statement) and the access path for a given section. This information is obtained from the SQL/MF "SQL_STMNTS" table. If the package section is dynamic and if the LOG DYNCOM option has been specified in SQLCS CONFIG; the package section text is obtained from the SQLCS_LOG table.

All the above lists, except the section detail, are ordered by descending resource consumption count. By default, the consumption is shown in terms of buffer lookups. But you can order by CPU usage or by I/O load, using the PFkey interface. By default, the total resource consumption is shown, that is, the consumption during one statement execution. A PFkey allows you to swap between the "totals" and "averages" counters.

When inspecting the saved runstats table, the output report will also contain the date and the time of the selected table rows.

Note   All options selected during a RunStats session are saved to a CMS file and represented as the default options at the next execution of RunStats.

Pfkey functions

PF1
Displays this help file.

PF2
Toggles between numeric and graphical output mode.
In numeric mode, the report shows the value of the counters and the number of executions. In graphical mode, the number of executions is not displayed.

PF3
Terminates the RunStats utility program.

PF4
Provides detailed information for the report line, pointed to by the cursor.

- In the user list, PF4 shows the packages executed by a user.
- In the program list, PF4 shows all executed sections (statements) within a designated package.
- In the program-section list, PF4 shows the section information from the SQL/MF "Statements" or the exception log table.

PF5
Swaps between the "Users" and "Programs" consumption reports.

PF6
When in the package section list, PF6 will invoke the statement recorder for the section pointed to by the cursor, provided that the statement recorder has been started.

PF7
Shows the previous page of the report.

PF8
Shows the next page of the report.

PF9
Shows the CPU usage counters.

PF10
Shows the buffer-lookup counters.

PF11
Shows the I/O counters.

PF12
Swaps between "Averages" and "Totals" counters.
The average counters show the total consumption, divided by the number of package executions.

Sample Runstats

The following figures show:

- the runstats for all executed packages
- the runstats for a selected package
- the statistics for a selected package section

Runstats per package

```
Page 1.1          SQL/Session Run Statistics per Program      05/13/96 14:07:12
--------------------------------------------------------------------------------
Data = BUFLOOK  .......10K.......20K.......30K.......40K.......50K.......60K

SQLCVRFY        ████████████████████████████████████████  56736
SQLCAXC2        ██   178
SQLCLEXP        ██   109


--------------------------------------------------------------------------------
PF 1 Help       2 Numerics  3 Quit       4 Detail    5 Users    6
   7 Page <<    8 Page >>   9 CPU       10          11 Tot/IO   12 Totals
```

Runstats per package section

```
Page 1.1          SQL/Session Run Statistics per Section      05/13/96 14:07:12
--------------------------------------------------------------------------------
Data = BUFLOOK  .......10K.......20K.......30K.......40K.......50K.......60K

SQLCVRFY (1)    ████████████████████████████████████████  56715
SQLCVRFY (2)    ██   21

--------------------------------------------------------------------------------
PF 1 Help       2 Numerics  3 Quit       4 Detail    5          6
   7 Page <<    8 Page >>   9 CPU       10          11 Tot/IO   12 Totals
```

Package section detail

```
 DATABASE  TSTADB          PROGCREA  SQLDBA        PROGNAME  SQLCVRFY
  SECTION  1                SAMPLED  4                 ISOL  CS
  CPUTIME  7800            LOCKTIME  0               IOWAIT  335
 COMMWAIT  2432             RDSCALL  610            DBSSCALL  55321
 DISPCALL  1238            ESCALATE  0              WAITLOCK  0
 DEADLOCK  0                  NROWS  55319           BLOCKED  Y
  BUFLOOK  56715           PAGREAD  1396            PAGWRITE  22
  DIRBUFL  1389            DIRREAD  21              DIRWRITE  0
  LOGREAD  0               LOGWRITE  0                 TOTIO  1440
  DSFAULT  33              BLOCKIO  0              INT_DBSP  0
LAST_DATE  1996-05-13     LAST_TIME  09.31.21
   ACCESS  No index access noted
------------------------------------------------------------------------------
SELECT ITEM,"SQL/DS HELP" FROM SQLDBA.SYSTEXT2



------------------------------------------------------------------------------
PF 1 Help      2 Numerics  3 Quit       4 Detail    5           6
   7 Page <<   8 Page >>   9 CPU       10          11 Tot/IO   12 Totals
```

9.4   Using SQLPULSE

The SQLPULSE function provides a graphical representation of DB2/VM activity by all running statements in all monitored databases. SQLPulse shows a bar graph for all running statements with one the following counters:

-   number of buffer lookups performed by the statement
-   total number of I/O requests performed by the statement
-   CPU used by the statement

Each position in the bar graph represents:

-   1 second of CPU usage
-   100 I/O requests
-   1000 buffer lookups

When started, the program continually displays the current DB2/VM activity with a display resolution of 3 seconds. The output data is ordered by descending resource consumption. To interrupt the automatic display, press the ENTER key which displays the PFkey interface.

9.4.1 Invocation

The function is invoked from the SQLCSU program using the SQLpulse Pfkey (PF6).

9.4.2 SQLPULSE Functions

SQLPulse provides a PFkey driven interface to invoke the following functions:

PF1
Displays the help file.

PF2
Resumes interval driven display after its interruption by a previous PFkey function.

PF3
Terminates SQLPULSE and returns to SQLCSU.

PF4
Provides a detail graph for the statement on the screen line pointed to by the cursor. See SQLPULSE Detail below.

PF6
Selects the resource item to be shown. By default, the BUFLOOK information is displayed. The PF6 key lets you toggle the selectable columns, in the following order: BUFLOOK, TOTIO, CPUTIME. Your latest selection is saved on file and re-used for subsequent SQLPULSE sessions. PF6 causes an automatic resume.

9.4.3 SQLPULSE Detail

The detail function provides a graphical representation of resource usage by the SQL statement, pointed to by the cursor. The function shows for the current SQL package section:

- the databasename, userid, package name and package section number
- the statement start and elapsed time
- the text of the SQL statement being executed (for both static and dynamic statements)
- the statement's access path
- a graphical representation of the following counters:

  - CPUtime used in milliseconds
  - Iowait time in milliseconds
  - Commwait time in milliseconds
  - Lockwait time in milliseconds
  - Number of RDS calls
  - Number of DBSS calls
  - Number of DB2/VM dispatcher calls
  - Number of locks
  - Number of lock escalations
  - Number of page and directory buffer lookups
  - Number of read IO's
  - Number of write IO's
  - Number of accesses to internal DBspaces
  - Number of dataspace pagefaults

The graphics are refreshed every 3 seconds. If the logical unit of work terminates, the global screen is automatically re-activated.

To stop the automatic refresh, press ENTER. This freezes the bar graph and enables the following PFkeys:

PF1
Displays this help file.

PF2
Resumes the time-driven display.

PF3
Terminates SQLPULSE detail and returns to the global display.

PF4
Invokes the SQL/Command Analysis program product for the current statement.

PF5
Displays the full and formatted SQL statement text. Long statements may take several pages to display. Use PF7 and PF8 to browse through the pages. Press PF3 to terminate the text display.

PF7
Shows SQLPULSE detail for the previous agent in the list.

PF8
Shows SQLPULSE detail for the next agent in the list.

Sample SQLPULSE session

The pulse output for all running packages:

```
          Tot I/O  ........1K........2K........3K........4K........5K........6K

TSTADB  SQLCVRFY  ███████████
SQLOPL  MK30A     ██████
SQLDBA  ARIISQL   ███
SQLOPL  ARIRSQL
```

The pulse output for a selected package:

```
Database  TSTADB            SQL_id   TSTAST1           VM_id    TSTAST1
ProgName  SQLCVRFY          Section  1                 Command  Fetch
Starttime 15:28:14          Elapsed  00.00.09          Status   Running
Access    No index access noted

SQL_Cmnd  SELECT ITEM,"SQL/DS HELP" FROM SQLDBA.SYSTEXT2


--------------------------------------------------------------------------
CPU       ██ 6446
IOwait    █ 97
COMMwait  █ 1740
LOCKwait
NrRows    ███████████████████ 51233
RDScall   █ 565
DBSScall  ███████████████████ 51296
DISPcall  █ 1139
Nr_Locks
Escalate
Buflook   ████████████████████████████████ 53848
Read IO   ███████ 1298
Write IO  █ 3
Int DBSP
```

10   Using SQL/MF: Table Editor

All data recorded by SQL/MF in its monitoring tables are accessed using the SQL/MF Table Editor.

To start the table editor, run the SQLCSU program and the PF10 (Logtable) key.
A menu is now displayed, allowing you to choose one of the following report classes:

  (1) the statement exception reports (statements logged due to their resource consumption)
  (2) the statement statistics reports (last recorded statistics for all statements in all DB2/VM packages)
  (3) the package statistics reports (last recorded statistics for all DB2/VM packages)
  (4) the benchmark log reports (statements logged due to a benchmarking request)
  (5) the statement recording reports (if the facility has been enabled)
  (6) the lockwait recording reports (if the facility has been enabled)
  (7) the last recorder extract report, to redisplay the last extract table created by option (5)
  (8) the last lockwait extract report, to redisplay the last extract table created by option (6)

Select one of the report classes by placing the cursor on the corresponding screen line. Then press the <Enter> key.
All reports available for the selected report class are shown.. Reports are provided to display the log in a predefined order: in chronological order, by username, by package name, by CPU usage, by I/O usage .., as shown in the report descriptor line on the screen.
You can write your own log reports, which will be added to the menu automatically. For details on this subject, please read page 275.
To select a particular report, place the cursor on the corresponding screen line and press PF4, the Execute key.

```
                         SQL/MF Report Menu
--------------------------------------------------------------------------
Reportname  Description

--------------------------------------------------------------------------
LCHRONO     Exception Log in chronological order
LCPU        Exception Log by CPU consumption
LINDX       Exception Log by index name, with non-indexed accesses at top
LLOCK       Exception Log by lockwait time
LLOGR       Exception Log by log reason
LPACK       Exception Log by package name
LTOTIO      Exception Log by I/O counts
LUSER       Exception Log by username
LUSERDEF    Exception Log in user-defined order
LWAITIO     Exception Log by I/O wait time



--------------------------------------------------------------------------
PF  1           2           3 Quit     4 Execute   5           6 Top
    7 Page <<<  8 Page >>>  9 Bottom   10          11          12
```

All reports contain variable symbols, which you can assign on the screen that is displayed next. Most variables have default values, but you may alter the displayed default values. Your input should be a syntactically valid SQL expression. Following variable symbols can be assigned:

&SEL_DATE_FROM
&SEL_DATE_TO

SEL_DATE_FROM and SEL_DATE_TO specify the date range for the log table rows to be displayed. The default value for both variables is CURRENT DATE, which will display today's log table entries. You may enter any valid SQL duration expression to specify another range, for example:

```
SEL_DATE_FROM   CURRENT DATE - 7 DAYS
SEL_DATE_TO     CURRENT DATE
```

to display the log entries of the past week.

&OPT_CLAUSE

If desired, you can specify an expression on one or more columns of the log table. For example COMM_TYPE='OPEN' to display SQL OPEN statements only. For a description of the log table, please read page 259 and following.

&ORDER_BY

Defines the default order of the report, which you may modify or complement.

```
     Assign placeholders for macro BCHRONO
---------------------------------------------------
Placeholder        Substitute value
---------------------------------------------------
&SEL_DATE_FROM     CURRENT DATE
&SEL_DATE_TO       CURRENT DATE
&OPT_CLAUSE
&ORDER_BY          LOGSTAMP
```

The Report function keeps the latest values assigned to each report variable in a CMS file, named <Reportname XPARMS>. When the same report is subsequently invoked, these values are represented as defaults.

10.1 Report Display

When you have completed your selection, the selected log table rows are displayed on the screen. For a description of the data columns appearing in the report, refer to page 93.

The report is initially displayed in list mode. The screen then contains as many log rows as fit, each row taking one screen line. Since a log table row cannot be displayed on 80 columns, a viewing window concept is used.

Alternatively, the report may be displayed in page mode. In this mode, a screen contains one table row only.

Please note the following:

-   all statistic columns in the log table show package section related values: the total consumption by the section is shown in the last statement of that section

-   all chronological statistics in the log table (CPUtime, IOWAIT time etc.) are expressed in milliseconds

Note:

To prevent locks on the log table during browsing, the table editor maintains the selected log table rows in storage and performs browsing from storage. When requesting large reports, you may get "Unable to obtain storage" messages. You may receive the same error when invoking auxiliary functions (such as Command Analysis) from the report. To work comfortably with the table editor, a virtual storage size of 8 Mb or more is recommended. Short-on-storage conditions detected by the editor, will cause an automatic CMS IPL.

10.2 Acting upon the report

The report is shown by default in list mode, as follows:

```
SQL/Monitor Report                                              Page 1 of 1
-----------------------------------------------------------------------------
DATABASE USERNAME  AGENT PROGCREA PROGNAME SECTION COMM_TYPE BEGDATE   BEGTIME
-----------------------------------------------------------------------------
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP       7 AUXCALL   1995-06-16 10.49.1
TSTADB   TSTAST1      5 SQLDBA   CSTEST         2 COMMIT    1995-06-16 10.49.2
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP      10 PREPARE   1995-06-16 10.49.4
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP      10 DESCRIBE  1995-06-16 10.49.4
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP      10 OPEN      1995-06-16 10.49.4
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP      10 FETCH     1995-06-16 10.49.4
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP      10 CLOSE     1995-06-16 10.49.4
TSTADB   TSTAST2      4 SQLDBA   SQLCLEXP       1 COMMIT    1995-06-16 10.49.4
TSTADB   TSTAST2      6 SQLDBA   SQLCLEXP       7 AUXCALL   1995-06-16 10.52.1
TSTADB   TSTAST2      6 SQLDBA   SQLCLEXP       1 COMMIT    1995-06-16 10.52.1


-----------------------------------------------------------------------------
PF  1 Help      2 CmndText  3 Quit      4 Analyze   5 Format     6 Top
    7 Page <<<  8 Page >>>  9 Bottom   10 View <<< 11 View >>> 12 PageMode    13
ReportSw 14 R_Report
```

The page mode format is as follows:

```
 SQL/MF Report LCHRONO  System Demo                              Page 65 of 355
 -----------------------------------------------------------------------------
 DATABASE  SQLOPL           USERNAME  VSESP41           SQLID   CICSUSER
    AGENT  7                 BEGDATE  1995-01-09       ENDDATE  1995-01-09
  BEGTIME  10.33.40          ENDTIME  10.33.40         ELAPSED  00.00.01
 PROGNAME  BCP007           PROGCREA  BK                SECTION  2
COMM_TYPE  AUXCALL          ISOLATION RR               BLOCKED  Y
  CPUTIME  3                LOCKTIME  0                 IOWAIT   0
 COMMWAIT  1                 BUFLOOK  1                  NROWS   1
  RDSCALL  1                DBSSCALL  1               DISPCALL   1
 ESCALATE  0                WAITLOCK  0               DEADLOCK   0
  PAGREAD  0                PAGWRITE  0                 DIRBUFL  1
  DIRREAD  0                DIRWRITE  0                LOGREAD   0
 LOGWRITE  0                   TOTIO  0                DSFAULT   0
  BLOCKIO  0                INT_DBSP  0               LOGSTAMP   AC3ECFEB47A90
LOGREASON  DBSPSCAN              SEL
   ACCESS  No index access noted
 -----------------------------------------------------------------------------
SELECT STACKDATA, SUBSTR(STACKDATA,10,8),SUBSTR(STACKDATA,24,7),SUBSTR(STACKDATA
,18,6),SUBSTR(STACKDATA,81,1),SUBSTR(STACKDATA,84,7),SUBSTR(STACKDATA,82,2)
FROM BK.STACKSORT WHERE STACKID = 17241990 ORDER BY 2 ASC,3 ASC,4 ASC,5 ASC,6
ASC,7 ASC
 -----------------------------------------------------------------------------
 PF  1 Help      2 Objects   3 Quit      4 Analyze   5 Format    6 Top
     7 Page <<<  8 Page >>>  9 Bottom   10 Search   11 Hardcopy 12 ListMode
```

The table list is manipulated using a PFkey driven interface. Some statement keys operate on the current row. (The row that is highlighted on the screen).

PF1
Displays the editor help file. The help file provides for column help or for procedural help. Column help provides a short description of each data field displayed in the current list.

PF2
Invokes the Objects function. See Object Editor Function on page 144.

PF3
Terminates the table editor.

PF4
Forwards the current statement to the SQL/Command Analysis tool, if installed on your system.

PF5
Performs report formatting functions. See Formatting the report on page 150.

PF6
Shows the first page of the log report.

PF7
Shows the previous page of the log report.

PF8
Shows the next page of the log report.

PF9
Shows the last page of the log report.

PF10
In list mode, moves the viewing window to the left.
In page mode, calls the report search function described on page 152.

PF11
In list mode, moves the viewing window to the right.
In page mode, copy current screen to the hardcopy dataset. The hardcopy printer file is closed when the table editor is terminated.

PF12
Swaps between "list" and "page" display mode.

PF13
Invokes the report switching function to show another report, without leaving the current one. The report switching function is described on page 142.

PF14
Invokes the report switching function to show a related report for the current package, that is, for the package appearing on the highlighted report line. The report switching function is described on page 142.

Note   It is possible for a user to alter the above PFkey function assignments. See the section "PFkey remapping" on page 153.

10.3 Report Switching

Report switching provides for concurrent viewing of multiple (up to 8) table editor reports.

When viewing a report, report switching permits to initiate a new report. The current report becomes suspended and the new report becomes the current one. The editor maintains a list of suspended reports. When a report is suspended, it is appended at the end of the current suspended list. Therefore, the list reflects the chronological order of report initiation.

Invoking the report switching function using PF13 or PF14 shows a panel where you can enter the class for a new report or resume a report from the "suspended reports" list.

```
                    Report Switching
          _____


   Enter class of new report _         (see Note)
  or select suspended report LCHRONO
                             SCTEXT
                             R2CHRONO




  Note:  Class L = log reports
         Class S = statement statistics
         Class P = package statistics
         Class B = benchmark reports
         Class R = recorder reports
         Class W = lockwait reports
```

To initiate a new report, enter one of the indicated classes. The usual report selection menu will then be displayed to select a report in the specified class.

To resume a suspended report, position the cursor on its name and press ENTER.

Notes

When you invoke a related report using PF14, the normal switching rules apply. However, on the report panel that defines the monitor table selection arguments, the databasename and the package name from the current line of the invoking report will be displayed as default values. This allows to more easily display different reports related to the same package.

Up to 8 reports can be active concurrently.

The same report cannot be invoked more than once. (For example, you cannot initiate a new LCHRONO report if there is already such a report in the suspended list.)

When the primary report (the first report in the suspended list) is terminated via PF3, all secondary reports, if any, are terminated automatically.

When a secondary report is terminated via PF3, control is transferred to its parent report, that is, to the report preceding it in the suspended list.

10.4 Object Editor Function

The Objects function displays the object selection menu. To select one of the menu items, place the cursor on the corresponding screen line and press ENTER.

```
        Statement Objects
_____

    _  Statement Text
       Plan Index
       Table Indexes
       Table
       Table Columns
       DBspace
       Catalog Navigator


Select object with the cursor
       and press ENTER.

       Press PF3 to quit.

Current table is .............
```

When the Statement Statistics or Log table is edited, all the items shown are available, provided that the statement is a static one.

When the Log table is edited and an SQLCODE log entry is being processed, the additional item SQLCA Map is available.

Statement Text
Displays the full and formatted SQL statement text. Long statements may take several pages to display. Use PF7 and PF8 to browse through the pages. Press PF3 to terminate the text display.

Plan Index
Shows the SYSINDEXES row for the primary index used by the statement.

Table Indexes
Shows the SYSINDEXES rows for all indexes created on the table accessed by the statement.

Table
Shows the SYSCATALOG row for the table accessed by the statement.

Table Columns
Shows the SYSCOLUMNS rows for the columns of the table accessed by the statement.

Dbspace
Shows the SYSDBSPACES rows for the DBspace accessed by the statement.

SQLCA Map
For an SQLCODE log event, this object request formats all fields of the SQLCA at the time of the SQLCODE. Moreover, the DB2/VM help information for this SQLCODE is included from the DB2/VM Help tables. The SQLCA Map may contain several pages. Therefore, it is presented as a list and browsed using the standard PFkey interface.

Catalog Navigator
Browses the DB2/VM catalog tables. See page 146 for details.

The catalog object lists are displayed using the Table Editor. To manipulate the catalog lists, use the PFkey interface. Exiting the catalog list will return to the Object Menu.

The current table from the current statement is shown and may be modified, thus allowing catalog access for other objects in the database.

To exit the Object menu, press PF3.

10.5 The Catalog Navigator

10.5.1   Catalog Lists

The Catalog Navigator provides its users with a comprehensive interface for interactive, fullscreen access to the DB2 catalogs.

Within Catalog Navigator the user may select one of the following catalog lists:

- the package list
- the DBspace list
- the table list
- the table column list
- the table index list
- the table key constraint list
- the view list
- the synonym list
- the user list
- the owning user list

The user selects the desired list by entering:

- an **object class** from the list that appears in the upper right corner of the screen (A for package list, T for table list etc.)
- a **DB2 userid** representing the creator of the object or a privileged userid
- an **object name** designating the name of the table, index etc. Userid and objectname may be generic. A generic name will obtain information for multiple catalog objects. If userid or objectname is omitted, % (all entities) is assumed.
- an optional **WHERE** clause allows to perform additional selection on any column of the target catalog table
- an optional **ORDER BY** option requests a specific list order

Using the PF2 key, any database where SQL/MF has been installed can be connected under the default DB2 userid.

When building the catalog list, the Navigator scans the DB2 catalogs for rows satisfying the supplied search criteria and stores the selected rows in an internal list. When all information has been extracted, the path to DB2 is closed and the internal list is displayed on the screen. The user may then browse the list or execute a command on an object in the list. Since browsing is done from the Navigator, no locks exist on the DB2 catalogs while the data is being examined.

10.5.2   Related Catalog Lists

The DB2 catalogs contain hierarchically structured information. To obtain complete information about a given SQL object, it is usually necessary to examine several catalog tables. For example: DB2 holds the basic characteristics of a table in SYSCATALOG, the description of the table columns in SYSCOLUMNS, the description of the table indexes in SYSINDEXES and so on.

With the **related list** concept, the user can easily navigate through the catalog hierarchy. To obtain related information for a given object in a base catalog list:
- point the cursor to the base objectname
- press PF4
- on the related object selection screen, enter:
    - the class of the desired related object
    - eventually a userid and or object name
    - an optional WHERE or ORDER BY clause

Nested related lists are supported: in a related list, you can invoke a new related list for any object in the current related list. There is no limit on the list nesting depth.

Following related lists that can be obtained for each base class:

Packages
- all DBspaces used by the package
- all indexes used by the package
- all users having run privileges on the package
- all tables referenced in the package
- all views referenced in the package

Columns
- all users having update authority on a given column
- the table where the column belongs to
- the view where the column belongs to

DBspaces
- all packages using tables in the DBspace
- all tables residing in the DBspace

Indexes
- all indexes used in the package

Owners
- all packages created by the user
- all DBspaces acquired by the user
- all indexes created by the user
- all tables created by the user
- all views defined by the user

Privileges
- the connect privileges of the user
- all packages which the user may execute
- all columns on which the user has update authority
- all table and view privileges of the user

Tables
- all packages containing references to the table
- the columns in the table
- the DBspace where the table resides
- the indexes created on the table
- the privileges granted on the table
- the synonyms defined on the table
- the views defined on the table
- the parent table in a referential constraint
- the dependent table(s) in a referential constraint
- the primary and foreign keys in a referential constraint

Views
- all packages containing references to the view
- the columns in the view
- the privileges granted on the view
- the synonyms defined for the view
- the tables referenced in the view
- the views defined on the view

10.5.3   Processing the catalog list

Following PFkeys are used to process the list:

PF1
Displays the Navigator helpfile.

PF2
When in a table or view list, invokes the table browse function.
On the browse selection screen, you may specify a WHERE or ORDER BY clause, used when performing the table SELECT.
The table list is processed as a catalog list.

PF3
Exits the Catalog Navigator.

PF4
Invokes a related list.

PF5
Perform catalog list formatting (see page 150). The Navigator however has no support for the hardcopy and window move functions. If needed, invoke the general hardcopy function using PA3.

PF6
Goes to the top of the list.

PF7
Displays the previous list page.

PF8
Displays the next list page.

PF9
Goes to the bottom of the mist.

PF10
Shifts display window to the left, if in listmode.

PF11
Shifts display window to the right, if in listmode.

PF12
Toggles between page and list mode. Pagemode displays one table row on the screen, listmode displays multiple rows on one screen.

10.6 Formatting the report

When invoked, the report format function displays a function menu. Select the desired function by cursor positioning or enter the code corresponding to the desired formatting function, as follows:

F   to hide, unhide, sort or displace report columns
H   to take a copy of the list on a CMS file
L   to move the viewing window to the left margin
R   to move the viewing window to the right margin
G   to enable graphical mode display for numerical columns
N   to disable graphical mode display for numerical columns

List reformatting

The F function shows all columns in the report, preceded by a + sign if the column is currently displayed or by a - sign if the column has been hidden previously. You may override the + or - sign in the following manner:

Enter **+**   to unhide a previously hidden column, so that the column is displayed again.

Enter **-**   to hide a column so that it is no longer displayed, although it remains in the report.

Enter **<**   to sort the report on this column in ascending sequence (low to high). Only one ordering column can be designated.

Enter **>**   to sort the report on this column in descending sequence (high to low). Only one ordering column can be designated.

Enter **1 - 9** to move the column to the corresponding position in the report. The column previously on that position will take the position of the column moved. Placing a column on position 1 of the report will freeze it: the editor always leaves the first column on the screen when moving the viewing window.

```
+DATABASE           +USERNAME           +AGENT              +PROGCREA
+PROGNAME           +SECTION            +COMM_TYPE          +BEGDATE
+BEGTIME            +ENDDATE            +ENDTIME            +ELAPSED
+ISOLATION          +INDCREA            +INDNAME            +CPUTIME
+LOCKTIME           +IOWAIT             +COMMWAIT           +RDSCALL
+DBSSCALL           +DISPCALL           +ESCALATE           +WAITLOCK
+BUFLOOK            +PAGREAD            +PAGWRITE           +DIRBUFL
+DIRREAD            +DIRWRITE           +LOGREAD            +LOGWRITE
------------------------------------------------------------------
PF  1 Help      2             3 Quit    4           5           6
```

The new report layout is saved as a "template" file and reused when the same report is invoked later. If you wish to return to the standard report layout, erase the template file, which is named <reportname> TEMPLATE A.

List hardcopy

The H function writes the entire list, in page mode format, to the SQLMFPRT dataset. The physical destination of the hardcopy is determined by the FILEDEF command stated in the SQLMFPFD EXEC, described on page 86. The default SQLMFPRT FILEDEF routes the hardcopy dataset to your virtual printer. The printer file is closed when the table editor is terminated. Note that a single-screen hardcopy function is provided when the list is processed in page mode.

Skip to left margin

When the display window has been moved, you can return to the first column in the list, by executing the L function.

Skip to right margin

When the display window has been moved, you can return to the last column in the list, by executing the R function.

Enable graphical mode

With graphical mode enabled, numerical columns are shown in bargraph format when the report is in pagemode. Moreover, the highest column value in the list is highlighted.

Disable graphical mode

Resets graphical mode.

PF key definitions:

PF1   Requests help
PF3   Terminates reformat.

10.7 Searching the report

The report search function is called by pressing PF10 when the list is displayed in page mode.

The search function displays the following search criteria panel:

```
        Search column .......
         Search value .......
    Search direction >
```

Search column
Enter the name of the list column to be searched. Use the column name that appears as column header on the display screen.

Search value
Enter the value to find in the search column.
The specified search value can be partial. The search column is examined over the length of the specified value only.
By default, an equal search is performed. By prefixing the search value with one of the logical operators < > <= >= ^= you can perform a not-equal, a lower-than or a greater-than search. Blanks may, but need not, intervene between the operator and the value.
You can also specify MIN or MAX as the search value, to find the minimum or maximum value of the search column in the list.

Search direction
Enter the > sign to search the list in forward direction, which is the default.
Enter the < sign to search the list in backward direction.

Notes

(1)   Except for a MIN or MAX search, which always scans the entire list, searching starts at the position of the current report line + 1. When the search is productive, the current line is positioned on the list entry found. This allows to repeatedly apply the same search arguments.

(2)   The last search criteria entered are redisplayed on the next search panel.

(3)   Logical operators and MIN/MAX functions can also be applied to non-numerical list columns.

PF key definitions:

PF1   requests help
PF3   exits search

10.8 PFkey Remapping

You may need a PFkey remapping facility, if you are using software systems, such as terminal session managers, that reserve some PFkey for their own use. Such PFkeys are intercepted by that software and not passed to the guest application, such as the SQL/MF table editor.

The remapping facility allows you to define a PFkey that will behave like and take over the functions of the "dead" PFkey.

You must define the alternate PFkeys in a CMS file named SSPCONS PFREMAP. This file is looked up, using the CMS search order, by the 3270 support module that is part of the Monitoring Facility (and of other SPR program products). If the file is found, the requested key remapping is performed.

For each alternate PFkey, a line must be coded in the remapping file, specifying

- the number of the PFkey to be remapped
- the number of the PFkey to be used as an alternate

Both numbers should be separated by at least one blank.

Example

If you are using a session manager that reserves PF13 and PF14 and you decide to use PF23 and PF24 as their alternates, code following lines in an SSPCONS PFREMAP file:

```
13 23
14 24
```

10.9 Configuring the Table Editor

By providing an SQLCSTXT CONFIG dataset, you can modify the operation of the table editor.

Following statement can be coded in the CONFIG file:

UNFORMATTED_TEXT

By default, the table editor formats the text of an SQL statement, by starting a line for the major SQL clauses. (Formatting is not attempted for 24-line screens). By coding the statement UNFORMATTED_TEXT, SQL texts are displayed in a unformatted manner.

11 Using SQL/MF: Inspecting the monitor tables

11.1 Examining the Exception Log table

The exception logs are recorded in the SQLCS_LOG table and the SQLCSU table editor is used to display them. Please refer to page 135 and following for a complete description of the Table Editor.

To start the program, run the SQLCSU program, press the PF10 key("LogTable") and select the Statement Exception Log menu entry. On the Log Report menu, choose one of the exception reports by pressing the Execute PFkey.

For a description of the data columns appearing in the report, refer to page 93.

11.2 Exception Log Reports available

LCHRONO
Exception log entries in chronological order

LCPU
Exception log entries by CPU usage

LDBSCALL
Exception log entries by DBSSCALL/RDSCALL ratio

LDEADLCK
Exception log entries related to deadlocks

LIDBSP
Exception log entries by internal dbspace access

LINDX
Exception log entries by index name

LLAST
Exception log entries in the last hour

LLOCK
Exception log entries by locks

LNSIA
Exceptions logs for non-selective index access

LPACK
Exception log entries by package and section

LREASON
Exception log entries by logging reason

LROWCOST
Exception log entries by row access cost (BUFLOOK/NROWS)

LTOTIO
Exception log entries by number of I/O's

LUSER
Exception log entries by username

LUSERDEF
Exception log entries by user defined criteria

LWAITIO
Exception log entries by IO wait time

11.3 Examining the Benchmark table

The benchmarking logs are recorded in the SQLCS_LOG table and the SQLCSU table editor is used to display them. Please refer to page 135 and following for a complete description of the Table Editor.

To start the program, run the SQLCSU program, press the PF10 key("LogTable") and select the benchmark log reports menu entry. On the Benchmark Report menu, choose one of the benchmark reports by pressing the Execute PFkey.

For a description of the data columns appearing in the report, refer to page 93.

11.4 Benchmark Reports available

BCHRONO
Benchmarked statements in chronological order

BCPU
Benchmarked statements by CPU usage

BDBSCALL
Benchmarked statements by DBSSCALL/RDSCALL ratio

BIDBSP
Benchmarked statements by internal DBspace access

BINDX
Benchmarked statements by index name

BLAST
Benchmarked statements in the last hour

BLOCK
Benchmarked statements by locks

BNSIA
Benchmark logs for non-selective index access

BPACK
Benchmarked statements by package and section

BTOTIO
Benchmarked statements by number of I/O's

BUSER
Benchmarked statements by username

BUSERDEF
Benchmarked statements by user defined criteria

BWAITIO
Benchmarked statements by IO wait time

11.5 Examining the Package Statistics

The package statistics are contained in the Log table and the SQLCSU table editor is used to display them. Please refer to page 135 and following for a complete description of the Table Editor.

To start the program, run the SQLCSU program, press the PF10 key("LogTable") and select the package statistics menu entry. On the Log Report menu, choose one of the package statistic reports by pressing the Execute PFkey.

For a description of the data columns appearing in the report, refer to page 93.

In addition to the package's resource consumption, the reports provide a column named LUWS, which shows the package execution frequency, that is, the number of times program statistics have been recorded.

The report is shown in list mode by default, as follows:

```
SQL/Monitor Report                                           Page 2 of 13
--------------------------------------------------------------------------
PROGNAME PROGCREA DATABASE    CPUTIME      RDSCALL     DBSSCALL     DISPCALL
--------------------------------------------------------------------------
DD98A     PA       SQLOPL           2            2           11            8
DD99A     PA       SQLOPL          23           16           71           51
DD99A     PA       SQLOPL           1            1            2            4
ELORXSQL  SQLDBA   SQLOPL         205           29          639          111
FZESQLDS  SQLDBA   SQLOPL        1042          532         2450         1536
GF02P     SG       SQLOPL        1666         1651         4953         3305
GF02P     SG       SQLOPL        1475         1474         4422         2948
GS80A     GS       SQLOPL           9            6           21           21
GS90A     GS       SQLOPL          18           18           27           19
IB51A     IB       SQLOPL          10            8           23           31
KW62A     KWM      SQLOPL           5            5           15           20
MO04A     KS       SQLOPL          19            6           41           19
--------------------------------------------------------------------------
PF  1 Help      2 CmndText  3 Quit       4 Analyze   5 Format    6 Top
    7 Page <<<  8 Page >>>  9 Bottom    10 View <<< 11 View >>> 12 PageMode
```

The page mode report format is as follows:

```
SQL/Monitor Report                                            Page 3 of 5
-----------------------------------------------------------------------------
PROGNAME  SQLCAXU2          PROGCREA  SQLDBA          DATABASE  TSTADB
 BEGDATE  1995-10-04            LUWS  4                CPUTIME  183
LOCKTIME  0                   IOWAIT  254            COMMWAIT  253
 RDSCALL  63               DBSSCALL  353             DISPCALL  157
ESCALATE  0                WAITLOCK  0               DEADLOCK  0
 BUFLOOK  424               PAGREAD  14              PAGWRITE  0
 DIRBUFL  246               DIRREAD  6               DIRWRITE  0
 LOGREAD  0                LOGWRITE  3                  TOTIO  23
INT_DBSP  4
  Access  Not applicable



-----------------------------------------------------------------------------
PF  1 Help      2 Objects   3 Quit      4 Analyze   5 Format    6 Top
    7 Page <<<  8 Page >>>  9 Bottom   10 Search   11 Hardcopy 12 ListMode
```

11.6 Package Statistic Reports available

PCOST

Shows for the current session, the packages with the highest resource consumption and, within a package, the package section with the highest consumption.

PDETAIL

Total package resource consumption by user and package during the requested date range.

PDETAVG

Average package resource consumption by user and package during the requested date range.

PPROG

Total package resource consumption by package during the requested date range.

PUSER

Total package resource consumption by user during the requested date range.

Notes

The PCOST report joins the package statistics from SQLCS_LOG with the statement statistics from SQLCS_SQL_STMNTS. It computes their SQL cost as: (TOTIO+(DBSSCALL/3)). The package cost is not necessarily equal to the summarized cost of all package statements, because the package statistics show the actual consumption for the current session, while the statement statistics show the consumption during the last execution.

The PDETAVG report computes the average consumption as (total_consumption ÷ number_of_LUWs_executed)

11.7 Examining the Statement Statistics

The statement statistics are kept in the SQL_STMNTS table and the SQLCSU Table Editor is used to display them.

To start the program, run the SQLCSU program, press the PF10 key("LogTable") and select the statement statistics menu entry. On the Log Report menu, choose one of the statement statistic reports by pressing the Execute PFkey.

Please refer to page 135 and following for a complete description of the SQLCSU Table Editor.

For each statement in the selected package(s), the statement statistics allow to view:

- The statement resource usage (for a description of those data columns, please refer to page 93).
- The access path of the statement. If a statement accesses multiple tables or participates in a referential constraint, multiple plan rows will be presented, each showing the name of the DBspace, table and index accessed. Moreover, the SEL column will show whether a selective or a non-selective index access has been performed. The resource consumption of the statement section is always contained in the first plan row.
- The text of the statement.

The list mode report format is as follows:

```
SQL/Monitor Report                                              Page 1 of 2
-------------------------------------------------------------------------------
DATABASE PROGCREA PROGNAME SECTION     SAMPLED    CPUTIME    LOCKTIME      IOW
-------------------------------------------------------------------------------
TSTADB   SQLDBA   CSTEST        1          1          8          0
TSTADB   SQLDBA   CSTEST        2          1       5017          0           1
TSTADB   SQLDBA   CSTEST        3          1       1371          0
TSTADB   SQLDBA   SQLCAXC2      4          2         13          0
TSTADB   SQLDBA   SQLCAXC2      5          2          4          0



-------------------------------------------------------------------------------
PF  1 Help      2 CmndText  3 Quit     4 Analyze   5 Format    6 Top
    7 Page <<<  8 Page >>>  9 Bottom   10 View <<< 11 View >>> 12 PageMode
   13 ReportSw 14 R_Report
```

The page mode report format is as follows:

```
 SQL/MF Report SCUSE1   System Demo                            Page 2 of 25
 ------------------------------------------------------------------------------
 DATABASE  TSTADB          PROGCREA  SQLDBA          PROGNAME  SQLCAXC2
  SECTION  6                   PLAN  1                SAMPLED  44
     ISOL  CS              CPUTIME  7               LOCKTIME  0
   IOWAIT  35              COMMWAIT  51              RDSCALL  3
 DBSSCALL  11              DISPCALL  5              ESCALATE  0
 WAITLOCK  0               DEADLOCK  0                 NROWS  1
  BLOCKED  Y                BUFLOOK  21              PAGREAD  3
 PAGWRITE  0                DIRBUFL  8               DIRREAD  2
 DIRWRITE  0                LOGREAD  0              LOGWRITE  0
    TOTIO  5                DSFAULT  2               BLOCKIO  0
 INT_DBSP  0                    SEL  Y             LAST_DATE  1996-07-15
 LAST_TIME  11.00.40



   ACCESS   Index SQLDBA.SQLCA_IX02 (+FILENAME
 ------------------------------------------------------------------------------
 SELECT QUERYNO FROM SQLCA_INDEX WHERE USERID=:CHAR8 AND FILENAME=:CHAR8 AND
 FILETYPE=:CHAR8 AND DBASE=:CHAR8



 ------------------------------------------------------------------------------
 PF  1 Help      2 Objects   3 Quit      4 Analyze   5 Format    6 Top
     7 Page <<<  8 Page >>>  9 Bottom   10 Search   11 Hardcopy 12 ListMode
```

11.8 Statement Statistic Reports available

SCAVGUSE
The average resource consumption of designated package sections. Should be used only when SECTION_STATS = 'SESSION' or 'DAILY', in which case SCUSE1 shows the total consumption per session or day.

SCDBASE
The average, maximum or minimum consumption counts for a designated database.

SCIO
Package sections with a high I/O usage, ordered by descending IO usage.

SCMPATH
The package sections having multiple access plans because they access multiple tables or internal DBspaces.

SCNBLOCK
The access path information and resource usage for all package sections that execute without blocking.

SCNROWS
The access path information and resource usage for all selected package sections, ordered by descending "number of rows processed".

SCNSIA
The statement statistics for all sections executing by non-selective index scan.

SCPATH1
The access path information for the last modification level of the selected statements.

SCPATH2
The access path information for all modification levels of the selected statements.

SCPATHNI
The package name and statement text for those statements that execute without index access.

SCRDBSP
The access path information and resource usage for all package sections that refer to tables in the designated DBspace.

SCRINDEX
The access path information and resource usage for all package sections that execute using the designated index (or indexes if a generic indexname is used).

SCRTABLE
The access path information and resource usage for all package sections that refer to the designated table (or tables if a generic tablename is used).

SCTEXT
The SQL statement text for all sections of the selected package(s).

SCUSE1
The resource usage for the last modification level of the selected statements.

SCUSE2
The resource usage for all modification levels of the selected statements.

SCUTIME
The resource usage for selected statements during a specified time period.

SROWCOST
Statement statistics by descending row access cost (BUFLOOK/NROWS)

11.9 Accessing the System Monitor tables

```
┌─────────────────────────────────────┐
│ SQL Monitoring Facility             │
│                                     │
│ Summary Report -----------> 0       │
│ Show SQL Counters --------> 1       │
│ Show Buffer Usage --------> 2       │
│ Show DBspace Usage -------> 3       │
│ Show Storage Pool Usage --> 4       │
│ Show Log Usage -----------> 5       │
│ Show Users ---------------> 6       │
│ Show Lock Contention -----> 7       │
│ Show Checkpoint Delays ---> 8       │
│ Show Connections ---------> 9       │
│ Execute Control Function -> C       │
│ Execute User Report ------> U       │
│                                     │
│ Function Number ==========>         │
│ Database Name ============>         │
│ Monitoring Date From =====>         │
│ Monitoring Date To =======>         │
│ Monitoring Time From =====>         │
│ Monitoring Time To =======>         │
│                                     │
│ Help --------------------> PF1      │
│ Exit --------------------> PF3      │
└─────────────────────────────────────┘
```

Pressing the SYSMON function key (PF11) in SQLCSU will display the above function selection screen. The control screen allows to:

-   Select a system report by function number
-   Select a monitored database for report inspection
-   Specify an inspection period through a combination of the DATE_FROM, DATE_TO, TIME_FROM and TIME_TO screen fields.

For a complete description of the interactive System Monitor reports, read page 171 and following.

Selecting a System Report

Select a system monitor report by entering one of the following report codes in the screen field Function Number:

0   the SUMMARY Report provides performance related data per monitoring session (i.e. per day).

1   Basically, the COUNTER Reports show the statistics that are usually obtained by the DB2/VM COUNTER * command and add various other statistics computed by the Monitor.

When the COUNTER reports are selected, another menu screen is presented to select one of the following COUNTER subreports:

- LUW Counters
- I/O Counters
- LRB (Lock Request Block) Counters
- VM Counters
- Summary Counters
- Dataspace Counters
- Extended Counters (average IO duration, dispatch delays ...)

2   the BUFFER USAGE Report shows the distribution of the DB2/VM buffers among DBspaces and storage pools.

3   the DBSPACE USAGE Report shows the page and free space utilization for all existing DBspaces. Due to its performance implications (it implies execution of the SHOW DBSPACE command), this report is optional: its availability depends on the SHOWDBSP parameter of the SQLMFIN configuration file.

4   the STORAGE POOL USAGE Report shows the page utilization for all defined storage pools per monitoring interval.

5   the LOG USAGE Report shows the usage of DB2/VM Logfile space during the monitor interval and other items obtained from the DB2/VM SHOW LOG command.

6   the USERS Report provides a list of the users active at each interval

7   the LOCK CONTENTION Report is a Users report including the entries for users in LOCK wait state only; it shows additional information on held locks (such as the lock owner).

8   the CHECKPOINT DELAY Report lists the occurrences of checkpoint delay and the users active at that time.

9   the CONNECTIONS Report lists for each monitoring interval the number of active and waiting DB2/VM agents and IUCV connections.

C executes an SQLMF control function. The control function screen is as follows:

```
     SQL/MF Control Functions

     SQL Operator Command
     Shutdown SQLMF
     Modify Monitor Interval
     Load Monitor Tables
     Print Monitor Tables
     Resume Monitoring
     Suspend Monitoring
```

U Executes a user defined report on the monitoring tables. The 'user report' screen allows to specify a variable ORDER BY and SELECT WHERE clause for the user SQL statement specifications.

Additional Report Selections

After the initial report selection, the next screen allows:

- To request a detail, a total, a maxima or an averages report. The detail report produces one line per monitor interval, the other reports produce one line per monitor session (i.e. per day).

- To modify the default ORDER of the generated report by adding column names or the ASC / DESC specification to the ORDER columns displayed.

- Database and date or time selections generate a SELECT WHERE to which the user may add his own specifications on the WHERE screen line. Normal SQL syntax is used to formulate the condition.

Report Display

When you have completed your selection, the selected monitor table rows are displayed on the screen in list mode, using the SQLCSU Table Editor described on page 135 and following.

Sample report

```
MONTIME  MONDATE           RDSC        DBSSC         LUWS       TOTIO PAGEHIT
-----------------------------------------------------------------------------
17.50.09 1995-08-26        2589        234124           4           0      98
17.50.53 1995-08-26        1708        152202           7           2      98
21.06.25 1995-08-26        2703        238469          16          16      98
21.06.56 1995-08-26        1590        137798          14          14      98
21.07.34 1995-08-26          54           159           6           2      91



-----------------------------------------------------------------------------
PF  1 Help      2 Menu      3 Quit      4 CmndLog   5 Format    6 Top
    7 Page <<<  8 Page >>>  9 Bottom   10 View <<< 11 View >>> 12 PageMode
```

Acting upon the report

The table list is manipulated using a PFkey driven interface. Some command keys operate on the current row. (The row that is highlighted on the screen).

PF1 Displays the table editor help file. The help file provides for column help or for procedural help. Column help provides a short description of each data field displayed in the current report.

PF2 Recursively invokes the SQLMF menu, to generate another report without terminating the current one. There is no limit, other than available storage, to the number of recursive report menu calls.

PF3 Terminates the table editor.

PF4 Views the Statement Monitor Log without terminating the current report.

PF5 Performs report formatting functions. See Formatting the report on page 150.

PF6 Shows the first page of the log report.

PF7 Shows the previous page of the log report.

PF8 Shows the next page of the log report.

PF9 Shows the last page of the log report.

PF10 Moves the viewing window to the left.

PF11 Moves the viewing window to the right.

PF12 Swaps between "list" and "page" display mode.

11.10   System Monitor Reports

11.10.1   Session Summary Report

This report displays for the specified database and the specified period, the following session statistics :

- name of the database monitored
- monitoring date
- RDS calls during the database session
- DBSS calls during the database session
- LUWs during the database session
- Rollbacks during the database session
- Checkpoints during the database session
- Long locks during the database session
- Lock escalations during the database session
- Nr of Locks resulting in wait
- Deadlocks during the database session
- Page buffer lookups during the database session
- Page reads during the database session (dataspace reads if VMDSS installed)
- Page writes during the database session (dataspace writes if VMDSS installed)
- Directory buffer lookups during the database session
- Directory page reads during the database session
- Directory page writes during the database session
- Log page reads during the database session
- Log page writes during the database session
- DASD reads during the database session
- DASD writes during the database session
- DASDR + DASDW during the database session
- Dataspace pagefaults during the database session
- Read ratio i.e. the percentage of (PAGRD+PAGWR) that is PAGRD.

11.10.2   LUW COUNTER Reports

The detail report displays following items for each monitoring interval :

- name of the database monitored
- monitoring date
- monitoring time
- number of calls to the Relational Data System during the interval
- number of calls to Database Storage Subsystem during the interval
- RDS/DBSS ratio (giving a complexity approximation of the average SQL access during the interval)
- number of LUWs started during the interval
- number of LUWs rolled back during the interval
- number of checkpoints taken during the interval
- number of failing lock escalations during the interval
- number of successful lock escalations during the interval
- number of lock requests resulting in wait during the interval
- number of deadlocks detected during the interval

The totals report provides the sum of the above items per monitoring session for the selected period.

The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.3 IO COUNTER Report

The detail report displays following items for each monitoring interval:

- name of the Database monitored
- monitoring date
- monitoring time
- number of page buffer lookups during the interval
- number of data pages read during the interval
- number of data pages written during the interval
- data page hit ratio during the interval (see page 376)
- directory buffer lookups during the interval
- directory pages read during the interval
- directory pages written during the interval
- directory page hit ratio during the interval (see page 376)
- number of log pages read during the interval
- number of log pages written during the interval
- total number of DASD reads as PAGRD+DIRRD+LOGR
- total number of DASD writes as PAGWR+DIRWR+LOGW
- total number of DASD I/O's as DASDRD+DASDWRT
- dataspace hit ratio for all pools (see page 376)
- dataspace access hit ratio for all pools (see page 376)

The totals report provides the sum of the above items per monitoring session for the selected period.

The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.4   Lock Request Block COUNTER Report

The detail report displays following items for each monitoring interval:

-   name of the Database monitored
-   monitoring date
-   monitoring time
-   number of calls to RDS during the interval
-   number of calls to DBSS during the interval
-   number of LUWs started during the interval
-   number of lock request blocks (LRBs) defined within SQL
-   number of lock request blocks in use at monitoring time
-   maximum number of LRBs used by any application at monitoring time

The totals report provides the sum of the above items per monitoring session for the selected period.

The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.5 VM COUNTER Report

The detail report displays following items for each monitoring interval :

- name of the Database monitored
- monitoring date
- monitoring time
- number of calls to RDS during the interval
- number of calls to DBSS during the interval
- number of LUWs started during the interval
- CPU time (in seconds) used by the Database server during the interval
- number of VM page reads for the Database server during the interval
- number of VM page writes for the Database server during the interval
- the storage working set (as computed by VM/ESA) of the Database server at the interval
- smoothed CPU utilization percentage at monitoring time
- smoothed VM paging rate at monitoring time
- smoothed storage utilization percentage at monitoring time
- smoothed VM expansion factor at monitoring time

The totals report provides the sum of the above items (the maximum values for the VM related items) per monitoring session for the selected period.

The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.6   COUNTER Summary Report

The detail report displays following items for each monitoring interval :

- name of the Database monitored
- monitoring date
- monitoring time
- number of calls to RDS during the interval
- number of calls to DBSS during the interval
- number of LUWs started during the interval
- total IO count during the interval
- page buffer hit ratio during the interval
- directory buffer hit ratio during the interval
- CPU time (in seconds) used by the Database server during the interval
- number of VM page reads for the Database server during the interval
- number of VM page writes for the Database server during the interval
- the storage working set (as computed by VM/ESA) of the Database server at the interval
- smoothed CPU utilization percentage at monitoring time
- smoothed VM paging rate at monitoring time

The totals report provides the sum of the above items (the maximum values for the VM related items) per monitoring session for the selected period.

The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.7　Dataspace Counter Report

The detail report displays following items for each monitoring interval and for each pool:

- name of the Database monitored
- monitoring date
- monitoring time
- pool number
- number of buffer lookups
- number of dataspace reads
- number of dataspace writes
- number of dataspace pagefaults
- number of Block I/O requests
- dataspace hit ratio (see page 376)
- dataspace access hit ratio (see page 376)
- target working set
- current working set
- save interval

The totals report provides the sum of the above items per monitoring session for the selected period. For the hit ratios, the maximum and the minimum hit values are shown.

The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.8   Extended Counter Report

The detail report displays following items for each monitoring interval :

- name of the Database monitored
- monitoring date
- monitoring time
- average duration of a Block I/O in microseconds
- average duration of a dataspace pagefault in microseconds
- number of checkpoints during the interval
- average duration of a checkpoint in milliseconds
- number of checkpoint delays during the interval
- average duration of a checkpoint delay in milliseconds
- number of times dataspace pages were saved during interval as a result of the SAVEINTV parameter
- average number of pages saved per request
- number of times Database had to wait for a save block
- number of dataspace page prefetches during the interval
- average number of pages on the prefetch list
- number of times dataspace pages were released during interval as a result of the TARGETWS parameter
- average number of pages released per request
- maximum number of slots in the DB2 package cache
- number of packages loaded
- number of packages loaded for active agents
- total number of packages monitored in the DB2 session
- maximum number of agents active during the monitor interval
- dispatching delay for interactive agents in microseconds
- nr of times a dispatching delay for interactive agents was detected
- dispatching delay for non inter-active agents in microseconds
- nr of times a dispatching delay for non-interactive agents was detected

The totals report provides the maximum value the above items per monitoring session for the selected period. The averages report provides the average value for the above items per monitoring session for the selected period.

11.10.9 Dbspace Buffer Usage Report

The detail report displays following items per monitoring interval for each Dbspace in the selected Database :

- name of the Database monitored
- monitoring date
- monitoring time
- dbspace name
- number of page buffers used by this dbspace at monitoring time
- number of directory buffers used by this dbspace
- percentage of page buffers used by this dbspace
- percentage of directory buffers used by this dbspace
- number of page buffers used by all dbspaces at monitoring time
- number of directory buffers used by all dbspaces
- the NPAGBUF DB2/VM initialization parameter
- the NDIRBUF DB2/VM initialization parameter

The maxima report provides the maximum values per session for the above items in the selected monitoring period.

The averages report provides the average values of these items per monitoring session.

11.10.10 Storage Pool Buffer Usage Report

The detail report displays following items per monitoring interval for all storage pools in the selected Database :

- name of the Database monitored
- monitoring date
- monitoring time
- storage pool number
- number of page buffers used by this storage pool at monitoring time
- number of directory buffers used by this storage pool
- percentage of page buffers used by this storage pool
- percentage of directory buffers used by this storage pool
- number of page buffers used by all storage pools at monitoring time
- number of directory buffers used by all storage pools
- the NPAGBUF DB2/VM initialization parameter
- the NDIRBUF DB2/VM initialization parameter

The maxima report provides the maximum values per session for the above items in the selected monitoring period.

The averages report provides the average values of these items per monitoring session.

11.10.11 DBSPACE Usage Report

The report displays following items per monitoring session :

- database name
- monitoring date
- dbspace name
- number of data pages defined in the dbspace
- number of data pages currently in use
- percentage of data pages currently in use
- current data page freespace percentage
- number of index pages defined in the dbspace
- number of index pages currently in use
- percentage of index pages currently in use
- current index page freespace percentage
- number of header pages defined in the dbspace
- number of header pages currently in use
- percentage of header pages currently in use
- current header page freespace percentage

11.10.12 DBEXTENT Usage Report

The detail report displays following items for each monitoring interval

- Database name
- monitoring date
- monitoring time
- storage pool number
- number of pages defined in the storage pool
- number of storage pool pages currently in use
- number of storage pool pages currently free
- percentage of storage pool pages currently in use
- if the short on storage condition did occur for this storage pool during the interval

The maxima report provides the maximum values per session for the above items in the selected monitoring period.

The averages report provides the average values of these items per monitoring session.

11.10.13 DB2/VM Log Usage Report

The detail report displays following items for each monitoring interval

- database name
- monitoring date
- monitoring time
- number of bytes defined for the logfile
- number of bytes written on the logfile during the last monitoring interval
- percentage of logfile space used at monitoring time
- number of logfile pages remaining before log overflow, if archive disabled
- number of logfile pages remaining before implicit log archive, if archive enabled
- number of logfile pages remaining before a checkpoint will be taken

The maxima report provides the maximum values per session for the above items in the selected monitoring period.

The averages report provides the average values of these items per monitoring session.

11.10.14 User Status Report

This report displays following items for each user active during the monitor interval:

- database name
- monitoring date
- monitoring time
- name of the user
- DB2/VM wait state of this user (agent) as:
  - CKPT waiting to perform a checkpoint
  - COMM in communication wait: agent is waiting for an SQL request
          from the user
  - I/O   waiting for input/output completion
  - LOCK waiting for a database resource
  - OUTB waiting for a directory buffer
  - OUTP waiting for a page buffer
- number of locks held by this user
- number of long locks held by this user

The following columns are filled in only when the user is in the lock wait state :

- type of the lock wanted as:
  - DB   database lock wanted
  - DBSP   DBspace lock wanted
  - IKEY   index key lock wanted
  - IPAG   index page lock wanted
  - PAGE   data page lock wanted
  - ROW   table row lock wanted
  - SYS   system lock wanted
  - TABL   table lock wanted

- dbspace name of the wanted lock
- name of the user holding the wanted lock
- request mode of the lock held as :
  - SIX   share and intention exclusive lock
  - IS   intention share lock
  - IX   intention exclusive lock
  - S   share lock
  - X   exclusive lock

- duration of the held lock as :
  - INST   state of a lock is being tested
  - LONG   lock held until end of LUW
  - MED   lock held until end of LUW or explicit release
  - SHORT lock held until end of DBSS call

11.10.15 Lock Contention Report

Selecting the Lock Contention report provides a User Status Report including those users only that have been found in lock wait state. The report provides the same data as the User Status Report, described above.

11.10.16 Checkpoint Delay Report

The detail report displays following items for each occurrence of a long (more than 3 second) checkpoint delay. A checkpoint delay is the situation where the DB2/VM checkpoint system agent waits for the completion of long-running DBSS calls in order to initiate a checkpoint.

- database name
- monitoring date
- monitoring time
- name of user active at checkpoint delay time and possibly causing it
- wait state of the user
- users agent number
- set to Y when the user runs a read-write application; N when the application is read-only.
- name of package executed
- package section number executed

The totals report shows the number of checkpoint delays during each monitor session.

11.10.17 Connections Report

The detail report displays following items for each monitoring interval

- database name
- monitoring date
- monitoring time
- number of users currently connected to DB2/VM i.e. holding an IUCV connection
- number of users currently in LUW and holding a DB2/VM agent structure
- number of users waiting for a DB2/VM agent structure to become available
- number of users not in LUW but holding an IUCV connection
- number of agents structures currently available
- number of IUCV/APPC connections currently available

The maxima report provides the maximum values per session for the above items in the selected monitoring period.

The averages report provides the average values of these items per monitoring session.

12 Using SQL/MF: Printing the monitor tables

12.1 Printing the System Monitor Tables

Printing of the system monitor tables is performed by the System Monitor, using an SQLDBSU command stream.

At the begin of each day, all monitor data for the previous day are automatically printed, if requested by means of the PRINT parameter in the SQLMFIN CTL configuration file.

The print control function of the SQLMF program can be used to print all monitor data for the current day.

12.2 Printing the Statement Monitor Tables using SQLCSPRT

The SQLCSPRT program provides a printed report facility for the benchmark log, the exception log, the package statistics and statement statistic tables for a given Database. The contents and the selection criteria for the reports produced by SQLCSPRT are defined in report control files. These files have the reportname as their CMS filename and "SQLCSPRT" as filetype. As for the SQLCSU table editor, you can easily create your own report layouts. How to do so is fully explained on page 278.

SQLCSPRT produces its report on the SQLMFPRT dataset, using OS/QSAM. The physical routing of this dataset is determined in the SQLMFPFD EXEC, as described on page 86.

The invocation syntax of the SQLCSPRT program is as follows:

```
[EXEC] SQLCSPRT
       reportname
       [DATABASE databasename]
       [DATE_FROM begin_date]
       [DATE_TO end_date]
       [WHERE selection clause]
       [ORDER BY ordering clause]
```

The reportname is the only required argument. If the optional arguments are omitted, default values are taken from the report definition. The following table shows the distributed reports, their purpose and their default argument values.

Report name	Purpose	Default Database	Default Date_from	Default Date_to	Default Where	Default Order_by
BENCHLOG	Print the benchmark log	All	CURRENT DATE	CURRENT DATE	none	Database, Logstamp
EXCPLOG	Print the exception log	All	CURRENT DATE	CURRENT DATE	none	Database, Logstamp
EXCPLOGU	Print the exception log by usage	All	CURRENT DATE	CURRENT DATE	none	Database, BUFLOOK DESC
PRGSTATS	Print the package statistics	All	CURRENT DATE	CURRENT DATE	none	Database, Date, Package Name
CMDSTATS	Print the statement statistics per package	All	CURRENT DATE	CURRENT DATE	none	Database, Date, Package Name, Section
CMDUSAGE	Print the statement statistics by usage	All	CURRENT DATE	CURRENT DATE	none	Database, Date, BUFLOOK DESC

The following is a sample print output page:

```
DATABASE : SQLOPL         USERNAME : VSESP41       SQLID : CICSUSER
   AGENT : 7                BEGDATE : 1995-09-09   ENDDATE : 1995-09-09
 BEGTIME : 09.47.54         ENDTIME : 09.47.54     ELAPSED : 00.00.00
PROGNAME : FZESQLDS         PROGCREA : SQLDBA       SECTION : 4
COMM_TYPE : FETCH          ISOLATION : CS          BLOCKED : Y
 CPUTIME : 7               LOCKTIME : 0             IOWAIT : 0
COMMWAIT : 1                BUFLOOK : 55             NROWS : 1
 RDSCALL : 2               DBSSCALL : 26          DISPCALL : 3
ESCALATE : 0               WAITLOCK : 0           DEADLOCK : 0
 PAGREAD : 0               PAGWRITE : 0            DIRBUFL : 0
 DIRREAD : 0               DIRWRITE : 0            LOGREAD : 0
LOGWRITE : 0                  TOTIO : 0            DSFAULT : 0
 BLOCKIO : 0               INT_DBSP : 0           LOGSTAMP : AC3EC5B07FA10
LOGREASON : DYNCOM
  ACCESS : Index PA.DDUSERX1 (+TERMINAL
 CMDTEXT : SELECT PERSNR
           FROM PA.DDUSER T1
           WHERE TERMINAL = '9622'
```

12.3 Using SQLCSPRT interactively

As with the SQLCSU table editor, you can specify the print parameters for SQLCSPRT interactively.

To do so, type SQLCSPRT at the CMS prompt.

This will display a menu of the available print reports, from which you select one by means of the PF4 key.

The parameters needed for the report chosen will then be prompted for.

13   Using SQL/MF: Host Command Interface

13.1 Issuing Host Commands

Host commands can be issued:

- from the DB2/VM console
- using the CP SEND command from a secondary DB2/VM console
- from the Host Command interface of the SQLCSU program
- by means of the SQLMFCMD EXEC or using the SQLCOM EXEC (SQLCOM executes SQL commands only).
- by providing a CMS file named **SQLCSI PROFILE**. If such file is found on any of the accessed minidisks, it will be executed during SQLCSI startup. The file can contain CP, CMS and MONITOR commands. DB2 operator commands can not be coded. Like the SQLCS CONFIG, the PROFILE file may also contain **SECTION <databasename>** commands. Commands contained in a section will only be executed during SQL/MF initialization in the named database.

The host command interface can be tailored to installation requirements, using the SQLMUAPC exit, which is described on page 300.

13.1.1   CP Commands

To execute a CP command in the Database server, start the command with the prefix CP.

For example: CP Q RDR ALL

13.1.2   CMS Commands

To execute a CMS command in the Database server, start the command with the prefix CMS.

For example: CMS SENDFILE PROFILE EXEC TO USER1.

Please note the following:

CMS commands that cause a terminal read in the Database server (such as XEDIT), should NOT be issued. If you do so, you will get a CP message on your console. You should then logon the Database server and terminate the terminal read manually. In the mean time, all Database operations are suspended.

When invoking a CMS EXEC, the word EXEC may be omitted. The SQL/MF command processor determines whether the command name is a module or an exec.

13.1.3   DB2/VM Commands

To execute a DB2/VM operator command, no prefix is required.

For example: SHOW ACTIVE

Notes

Executing a DB2/VM command does not require a DB2/VM agent structure. The command is executed by forwarding it to the DB2/VM operator agent.

The DB2/VM command interface has its own transmission path to the Database servers. If the path has been locked by a user, other users will receive the dummy reply message ARI8888. Such lockouts are usually recognized by the command interface program. However, if the lock situation persists, the communications path with the default Database server can be unlocked by issuing the following command at the CMS prompt:

    SQLCOM RESET_PATH

13.2 DB2/VM Operator Command Extensions

13.2.1 FORCE_CHKPW

The FORCE_CHKPW command forces the designated agent only when a DB2/VM checkpoint is currently being delayed. In other cases, the force is ignored.

The command is intended for use by the SQLMUAP7 exit, when it attempts to clear the checkpoint delay condition.

Command format

FORCE_CHKPW agent_number

Example

SQLMFCMD <databasename> FORCE_CHKPW agent_number

13.2.2   QUERY_CHKPNT

The QUERY_CHKPNT command returns the number of DB2/VM checkpoints since the previous QUERY_CHKPNT command.

The command returns **N_CHKP nnn**, where nnn is the number of checkpoints. The first QUERY_CHKPNT in a DB2 session will return N_CHKP -1, meaning that no checkpoint counter can be returned yet.

End of output is signalled by a line that starts with ARI006.

Command format

QUERY_CHKPNT

Example

'PIPE CMS EXEC SQLMFCMD <databasename> QUERY_CHKPNT | stem reply.'
number_of_checkpoints = word(reply,1)

13.2.3 SHOW IDBSPUSE

The SHOW IDBSPUSE command returns following data for all agents that are using DB2 internal DBspaces:

- VM userid
- DB2 userid
- agent number
- packagename
- section number
- number of internal DBspaces in use for the package section
- number of internal DBspaces in use for the LUW
- number of internal Dbspace pages in use for the package section
- number of internal Dbspace pages in use for the LUW

End of output is signalled by a line that starts with ARI006.

Command format

SHOW IDBSPUSE

Example

'PIPE CMS EXEC SQLMFCMD <databasename> SHOW IDBSPUSE | stem reply.'

Notes:

- The result "number of internal Dbspace pages in use" is computed as: "number of internal Dbspaces in use" multiplied by the internal Dbspace size.
- The DB2 server where the SHOW IDBSPUSE command is executed, must have access to an SQLDBGEN or SQLADBSP file that states the current internal Dbspace size.

13.3 MONITOR Commands

A MONITOR command can be issued from the DB2/VM console, from the Host Command interface of the SQLCSU program or by means of the SQLMFCMD EXEC.

To execute an SQL/MF MONITOR command, start the command with the prefix MONITOR.

Following MONITOR commands are provided:

13.3.1   AUTOPREP

MONITOR AUTOPREP { ON | OFF | ? }

-   If the AutoPrep facility has been enabled (i.e. a <database> AUTOPREP file found), the facility can be set (AUTOPREP ON), reset (AUTOPREP OFF) or queried (AUTOPREP ?). When AutoPrep is off, dynamic SQL statements are executed unchanged.

-   The AUTOPREP status is maintained across database sessions.

-   The AUTOPREP ON command will synchronize the internal autoprep list with the current contents of the SQLMF_AUTOPREP table. This provides for restoring a backup of the Autoprep table, using the SQLMFENV EXEC for example.

13.3.2 BENCH

MONITOR BENCH + packagename

Inserts the named package in the benchmark list. The package benchmark request is temporary and effective for the current DB2/VM session only. When the session is restarted or when a MONITOR CONFIG command is executed, only the benchmark requests from the monitor configuration file will be in effect. The command can also be used to re-enable an already benchmarked package for benchmarking again.

You request specific benchmarking by specifying the exact name of the package. You may also use generic package names such as * or N*. These specifications will benchmark all packages or all package names starting with N.

Warning

During specific benchmarking, SQL/MF reduces the overhead due to benchmarking, by benchmarking a given package statement only once per session. This is not done in case of generic benchmarking. Therefore, generic benchmarking should be done in a controlled environment, that is, when it is known that few applications will execute. Otherwise, important overhead will occur due to benchmarking.

When benchmarking is finished, you should remove the generic benchmarking request as soon as possible, by executing a command such as BENCH - *

MONITOR BENCH - packagename

Removes the named package(s) from the benchmark list.

MONITOR BENCH ?

Lists all packages currently in the benchmarking list. If the package has been effectively benchmarked, it is indicated in the command output. The function also displays the current owner of the benchmarking facility.

13.3.3 CONFIGURE

Syntax : MONITOR CONFIG[URE] [{SQLCS CONFIG | configuration_filename}]

Requests the SQLCSI program in the Database server to perform dynamic reconfiguring by re-processing the SQLCS CONFIG (or "configuration_filename") file. This allows to modify the configuration parameters while the Database server is active.

The CONFIG request must be directed to the database server **and** to SQLCMDM, so that this component may refresh its copy of the configuration file. This is needed when changing configuration parameters that affect SQLCMDM operation (such as NOTIFY LOCKTIME etc.). If you have modified an active SQLMUAP3 EXEC, the CONFIGURE command will also drop the old copy and reload the modified one.

When a configuration filename is specified, that file will be processed, instead of the default filename SQLCS CONFIG. With multiple configuration files, SQL/MF can be reconfigured automatically for different periods of the day, for example, by submitting the MONITOR CONFIG command using the SQLMFCMD EXEC.

13.3.4   CQCOUNT

Syntax : MONITOR CQCOUNT

If a dataspace is used for communications between SQLCSI and SQLCMDM, use this command to obtain statistics about the current space usage in the dataspace. The command will show:

- the number of requests stored (by SQLCSI) and processed (by SQLCMDM)
- the number of times the dataspace was full
- the current number of pages in the free and the request queues
- the maximum number of pages ever allocated to the request queue

Please note that "pages" in the command output means logical pages of 32K. Multiply the counter by 8 to obtain the usage in physical pages. For example: 2 pages in the request queue means that 16 physical pages (or 64K) have been allocated.

13.3.5   DISABLE | ENABLE DBSPACE

Syntax: MONITOR DISABLE DBSPACE <dbspacenumber|dbspacename>

The command puts the DBspace designated by <dbspacenumber> or <dbspacename> offline. All packages attempting to access a table in this DBspace will receive SQLCODE - 711.

Syntax: MONITOR ENABLE DBSPACE <dbspacenumber|dbspacename> [FOR userid]

The command puts the DBspace designated by <dbspacenumber> or <dbspacename> online. If the FOR clause is specified, only the designated DB2 or VM userid will have access to the DBspace. All other users continue to receive SQLCODE -711. Multiple ENABLE commands with different FOR userids can be specified. An ENABLE command without the FOR clause enables the DBspace for all users.

Notes

-   Disabling a DBspace is done by monitor logic only and the monitor must be active to implement the support. The offline status is unknown to the DB2/VM server.

-   The DISABLE DBSPACE command disables data access only (DML statements such as SELECT, INSERT ...) DDL statements (DROP, ALTER ...) are **not** affected.

-   The disabled status of a Dbspace is kept across DB2 sessions.

Examples

MONITOR DISABLE DBSPACE HELPTEXT
MONITOR ENABLE DBSPACE 3

13.3.6   DISABLE | ENABLE PACKAGE

Syntax: MONITOR DISABLE PACKAGE <packagename>

> The command puts the designated package(s) offline. Users attempting to execute an offline package will receive SQLCODE -806. A generic packagename may be used, by terminating the name with an * sign.

Syntax: MONITOR ENABLE PACKAGE <packagename> [FOR userid]

> The command puts the designated package(s) online. If the FOR clause is specified, only the designated DB2 userid will have access to the package. All other users continue to receive SQLCODE -806. Multiple ENABLE commands with different FOR userids can be specified. An ENABLE command without the FOR clause enables the package for all users. A generic packagename may be used, by terminating the name with an * sign.

Note:

-   If the package must remain offline across DB2 sessions, the corresponding MONITOR DISABLE command should be inserted in the SQLCSI PROFILE, so that the command is executed at startup of SQL/MF.

-   Disabling a package is done by monitor logic only and the monitor must be active to implement the support.

13.3.7   DISABLE | ENABLE EXTNAME

Syntax: MONITOR DISABLE EXTNAME <extname.exttype>

> The command puts the designated programs identifed by <extname.exttype> offline. DRDA users attempting to execute such a program will receive SQLCODE -806.

> Both extname and exttype are required and a period must precede exttype. A generic extname or exttype may be used, by terminating the name or type with an * sign.

> Example: MONITOR DISABLE EXTNAME MSACCESS.EXE

Syntax: MONITOR ENABLE EXTNAME <extname.exttype> [FOR userid]

> The command puts the designated program(s) online. If the FOR clause is specified, only the designated DB2 userid will have access to the program. All other users continue to receive SQLCODE -806. Multiple ENABLE commands with different FOR userids can be specified. An ENABLE command without the FOR clause enables the program for all users. Generic names and types may be used, by including a final * sign.

> Example: MONITOR ENABLE EXTNAME *.*

13.3.8   DISABLE ?

This command shows all disabled DBspaces and packages on the console.

13.3.9 DISPTRACE

The DISPTRACE command implements the dispatcher trace facility. The facility generates an entry in a trace buffer, whenever a general purpose agent:

- enters the dispatcher, that is, enters the wait state
- returns from the dispatcher, that is, leaves the wait state

The trace buffer is managed in a wraparound manner and contains the last 4096 dispatcher events. Each trace entry contains following data:

- the event timestamp with microsecond precision
- the time elapsed (in microseconds) since the previous trace entry, that is, since the previous entry into the DB2 dispatcher
- the agent number
- the VM userid
- the package and the section number
- the type of the statement in progress
- the new dispatch state of the agent as "running" when the agent is dispatched, or a wait state descriptor (such as IOwait), when the agent enters the wait state
- if the agent returns from wait, the time passed in the wait state, in microseconds
- if the agent enters wait, the time passed in the running state, in microseconds
- the number of buffer lookups performed by the LUW
- the number of pages consecutively retrieved by the agent without entering the wait state

Although detail tracing is not performed for system agents, an entry is provided in the buffer at the start and the end of each DB2 checkpoint.

13.3.9.1 Starting a dispatcher trace

Syntax : **MONITOR DISPTRACE START [trace options]**

Starts a dispatcher trace. A number of trace options can be specified on the START keyword. These options can be specified in any order, as follows:

AUTOSTOP WAIT <time>
Requests that the trace be automatically formatted and stopped, as soon as an agent has spent more than <time> milliseconds in a short wait state (a short wait is an I/O or a dataspace pagefault wait). The purpose of AUTOSTOP is to avoid that abnormally long waits do not appear in the trace, due to buffer wraparound. The last entry in the trace output will be for the long-waiting agent.

AUTOSTOP DELAY <time>
Requests that the trace be automatically formatted and stopped, when a dispatch delay of more than <time> milliseconds is detected. A dispatch delay is the situation where an agent is ready to run but is not dispatched by DB2/VM because other agents have higher priority or because an agent temporarily dominates the Database. The last entry in the trace output will be for the delayed agent.

AUTOSTOP RUN <time>
Requests that the trace be automatically formatted and stopped, as soon as an agent has spent more than <time> milliseconds in the running state. The last entry in the trace output will be for that agent.

BUFFER
Specifies the number of entries in the trace buffer. The default is 4096 which results in a trace buffer of about 200K. With LEVEL 2 tracing, the default is 8192 which results in a trace buffer of about 400K. Specifying a very large buffer value may cause considerable paging overhead.

LEVEL 2
When level 2 tracing is active, the trace entry that shows an agent dispatch (a "running" entry) will also show all agents ready for dispatch at that time.

LOCK
Locks the trace buffer in real storage, so that no pagefaults occur due to tracing. This option requires that you have VM privilege A. If you don't, the lock option is ignored. An UNLOCK is performed automatically when the trace is stopped.

MULTITASK
Stores trace entries only when more than 1 agent is running. However, once tracing has started for an agent because it was not the only agent running, its tracing will not be stopped, even when the number of agents drops to 1, due to commits performed by other agents.

PACKAGE <name>
Performs tracing for the named DB2 package only.

VMID <name>
Performs tracing for the named VM userid only.

13.3.9.2 Querying trace status

Syntax : MONITOR DISPTRACE SHOW

The command shows:

- whether a trace has been started (autostop may deactivate the trace automatically)
- the number of active entries in the buffer
- the number of trace entries for dataspace faults and for I/O and communication waits
- the highest run time traced in microseconds
- the average run time traced in microseconds
- the highest short wait time traced in microseconds
- the average short wait time traced in microseconds

13.3.9.3 Formatting a trace

Syntax : MONITOR DISPTRACE FORMAT

The DISPTRACE FORMAT command formats the trace buffer entries into the SQLMF DISPTRACE file and send the file to the virtual reader of the user that initiated the trace. Tracing is not stopped.

13.3.9.4 Stopping a trace

Syntax : MONITOR DISPTRACE STOP

Performs a DISPTRACE FORMAT and stops tracing.

13.3.9.5 Trace Samples

MONITOR DISPTRACE START LEVEL 2 LOCK BUFFER 12000
MONITOR DISPTRACE STOP

13.3.10   ENDLINKS

Syntax : MONITOR ENDLINKS

Forces all running agents and terminates all pseudo-agents by severing their APPC or TCP/IP links. The ENDLINKS facility is executed during the DB2 shutdown process.

The command has been designed to facilitate the DB2 shutdown process. The command should be issued **prior** to the DB2 SHUTDOWN command. The SHUTDOWN command also causes the SQL/MF command interface to be closed: MONITOR commands can then no longer be issued.

Note
In a TCP/IP environment, ENDLINKS will issue the **NETSTAT DROP** command during link termination. Since NETSTAT DROP is a privileged command, the database server virtual machine must be listed in the **OBEY** list in the **PROFILE TCPIP** file.

13.3.11   FREE

Syntax : MONITOR FREE { Userid | ALL }

Removes a designated or all DB2/VM users from the hold state. For example, the commands

MONITOR HOLD ALL
MONITOR FREE ADM1

will hold all users except ADM1 and SQLDBA (by default).

13.3.12   FORCE ALL

Syntax : MONITOR FORCE ALL

Issues a DB2 FORCE operator command for all executing agents. Running SQL/MF components however are not forced.

13.3.13   FORCE_CHECKPOINT

Syntax : MONITOR FORCE_CHECKPOINT [LOCK_DB2]

Initiates DB2/VM checkpoint processing by posting the checkpoint agent. The command is ignored, if a checkpoint is already in progress.

The LOCK_DB2 option will force the DB2 subsystem into a global wait state at the end of the checkpoint, until an UNLOCK_DB2 command is received. During the wait state, no SQL statements can be executed, nor can DB2 operator commands be entered. Transmitted DB2 requests are queued until the end of the wait state. The purpose of the LOCK_DB2 option is to ensure that the consistency of the database, resulting from a checkpoint, is preserved, so that a database copy may be taken.

Note

A LOCK_DB2 request will be honoured, even if a DB2 checkpoint is already in progress when the FORCE_CHECKPOINT is received.

13.3.14   FORCE_RO

Syntax : MONITOR FORCE_RO {SUSPEND | RESUME | database dbspacename}

The FORCE_RO command should be used only when the FORCE_RO statement has been specified in SQLCS CONFIG.

MONITOR FORCE_RO SUSPEND

This command disables the FORCE_RO facility during the current DB2 session.
This command should be directed to the DB2 server machine.

MONITOR FORCE_RO RESUME

This command re-enables the FORCE_RO facility.
This command should be directed to the DB2 server machine.

MONITOR FORCE_RO database dbspacename

This command re-enables the FORCE_RO facility for the specified dbspace in the specified database.

This command should be directed to the SQLCMDM service machine.

13.3.15   FORCE_RW

Syntax : MONITOR FORCE_RW database dbspacename

This command disables the FORCE_RO facility for the specified dbspace in the specified database.

This command should be directed to the SQLCMDM service machine.

13.3.16   HOLD

Syntax : MONITOR HOLD { Userid | ALL | ? }

HOLD Userid   Puts the designated DB2/VM userid in the hold state.
HOLD ALL   Puts all users, except SQLDBA, in the hold status.
HOLD ?   Displays the names of the users current held.

When a user is being held, SQLCODE -561 is forced by the monitor for all SQL statements executed by that user, thus putting that user in an "offline" state.

The MONITOR HOLD ALL command is particularly useful during database maintenance, by ensuring that no users except the DBA can access the database.

LUW's active when the HOLD is issued are not forced. SQLCODE -561 will be returned at the next statement.

13.3.17   LOCK DBSPACE

Syntax :

MONITOR LOCK <databasename> DBSPACE <Dbspace_ID> IN {SHARE|EXCLUSIVE} MODE

The LOCK command locks the designated PUBLIC Dbspace in the named database according to the specified mode. The lock is held until an UNLOCK DBSPACE is submitted.

Specify the Dbspace_ID as:

- a Dbspace number
- a Dbspace name
- **ALL** to lock all Dbspaces (except those whose name starts with SYS)

The LOCK DBSPACE command must be transmitted using SQLMFCMD to the SQLCMDM service machine.

Example:

SQLMFCMD SQLCMDM MONITOR LOCK DBN1 DBSPACE TEST IN EXCLUSIVE MODE

13.3.18   PKGSHOW

Syntax : MONITOR PKGSHOW <Packagename>

If the named package is resident in the DB2/VM package cache, the command shows all package RT and RB structures on the console. These show all DBspace, table and index id's used by the sections of the package.

Please note that the console output may be quite extensive, if the package has many sections. During the console output, the database server is not able to handle user SQL requests. Therefore, the command should be used with discretion.

13.3.19   RESET_RUNSTATS

Syntax : MONITOR RESET_RUNSTATS

Clears the run statistic counters for all packages executed thus far.

13.3.20   RECORDER SHOW

Syntax : MONITOR RECORDER SHOW

Shows the entities (users, packages or terminals) for which recording is active, as a result of the RECORDER START command.

13.3.21   RECORDER START

Syntax : MONITOR RECORDER START { PACKAGE|TERMINAL|VMUSER|DB2USER} nnn

Starts recording for the named package, terminal, VM or DB2 userid. Up to 1024 recording entities may be active simultaneously. Before using this command, ensure that the recorder facility has been setup, as described on page 232.

13.3.22   RECORDER STOP

Syntax : MONITOR RECORDER STOP { PACKAGE|TERMINAL|VMUSER|DB2USER|ALL} nnn

Stops recording for the named package, terminal, VM or DB2 userid. STOP ALL will terminate recording for all entities previously started.

13.3.23   RELOAD

Syntax : MONITOR RELOAD

Terminates and restarts SQL/MF (more specifically, the SQLCSI component) in the target database, without restarting the DB2/VM server. The reload logic is also called, should an abend occur in the SQLCSI component. Reload can be performed while monitored agents are executing. However, after reload, the monitored data concerning these agents will be lost. Therefore, it is suggested that reload be performed when DB2 activity is low. The RELOAD command must be issued from the DB2/VM console; it cannot be issued from SQLCSU or SQLMFCMD.

13.3.24   RESUME

Syntax : MONITOR RESUME

Resumes monitoring after a SUSPEND command.

13.3.25   SHOW AGENT

Syntax : MONITOR SHOW AGENT n

Displays the addresses of several DB2/VM control blocks for the designated agent. Please note that the first user agent number is 4, which allows to issue the command for system agents.

13.3.26   SHOW DBSPACE

Syntax : MONITOR SHOW DBSPACE { n | RO }

This command shows the R/O or the R/W status for a given DBspace number.
SHOW DBSPACE RO will show all DBspaces that are in the read-only status.

13.3.27   SNAP

Syntax : MONITOR SNAP

This diagnostic command is provided for software support purposes. It creates the following
printer files:

SSPDUMP FILE
a snap dump of the monitor and the monitor shared segment

SQLCSI SNAPFILE
a snap dump of the major DB2/VM control blocks and module call structures for all real agents

13.3.28   STATS

Syntax : MONITOR STATS

The STATS command causes all pending package and package section statistics to be sent
immediately to the SQLCMDM machine, thus forcing a STAT_FREQ interval.

13.3.29   SUSPEND

Syntax : MONITOR SUSPEND

Immediately stops SQL statement monitoring and the related facilities, such as logging and
benchmarking.

13.3.30 TRACE

The MONITOR TRACE command controls the SQL/MF Application Tracing facility. The facility can be invoked only by means of the SQLMFCMD interface command.

The tracing facility records all SQL calls made by a designated user or package into the SQL/MF trace buffer, which is allocated within the shared segment SQLCSMDS.

A 32K buffer is provided per monitored Database for application tracing. The trace buffer size allows to keep the last 511 SQL calls issued from the application. Consecutively repeated calls from the same package section (such as an uninterrupted series of FETCH calls) require only one trace entry, which records the number of calls effectively performed in the "execution count".

The application trace buffer is used in a wrap-around manner. When the end of the trace buffer has been reached, the next trace requests re-use the oldest entries at the top of the trace buffer.

The TRACE command communicates with the target Database server by means of the SQL/MF shared segment only. IUCV communications are not used during the MONITOR TRACE command.

Therefore:

Trace output can be retrieved when the Database server is down.
Trace data stored remain available in the shared segment across Database sessions and trace data are accumulated across Database sessions, until a new trace request is submitted.
Trace parameters established using the MONITOR TRACE command are stored in the shared segment and are controlled by the TRACE command only. A trace request remains in effect for all subsequent Database sessions, until the trace is explicitly stopped by the user or until the shared segment is cleared (by a VM IPL for instance).

Due to the in-core tracing technique used, the system overhead due to tracing will be very small.

13.3.30.1   Starting a trace

An application trace must be initiated from the CMS prompt using the command:

    SQLMFCMD <databasename> MONITOR TRACE <trace-entity>

where:

<databasename> designates the name of the Database to be traced

<trace-entity> is specified as follows:

ALL	records all SQL calls made by all packages and all users. (Useful for global system tracing or problem determination).
VMUSER <name>	enables tracing for all packages executed by the designated VM userid.
SQLUSER <name>	enables tracing for all packages executed by the designated DB2/VM userid.
PACKAGE <name>	enables tracing for the designated DB2/VM package. (A package creator name is not specified or examined during tracing).

For example: SQLMFCMD DBN1 MONITOR TRACE PACKAGE PRG1

Once the TRACE command has been entered, tracing is active.

Per monitored Database, only one user or package can be traced at one point in time. If you request a trace while a trace for another user or package is in progress, an error message is returned.

13.3.30.2   Stopping a trace

The command SQLMFCMD <databasename> MONITOR TRACE STOP stops tracing without formatting. The traced data are lost.

13.3.30.3   Querying the trace status

The command SQLMFCMD <databasename> MONITOR TRACE QUERY shows the current owner of the trace facility and the number of trace buffer entries currently in use.

13.3.30.4　Stopping and formatting a trace

An application trace is stopped and the generated trace data are formatted by using following command from the CMS prompt:

SQLMFCMD databasename MONITOR TRACE FORMAT

The command generates the trace report on the A-disk of the requesting user as a CMS file named <trace-entity> SQLTRACE. If trace-entity was specified as 'ALL', the filename will be set to <databasename> SQLTRACE.

For example, if a trace was requested for package PRG1, the resulting trace report would be named PRG1 SQLTRACE. The SQLTRACE file records are 80 bytes long. The file is XEDITed as part of the TRACE FORMAT command. It can be manipulated subsequently using CMS commands.

13.3.30.5   DB2/VM data traced

Following data items are traced and appear in the SQLTRACE report file:

Userid
Name of the executing VMuser (or DB2/VM user if TRACE SQLUSER requested).

Package
Name of the package traced.

Section
Number of the package section traced.

Statement
Type of the statement traced (OPEN, FETCH and so on).

Started
Starttime of the statement.

Runseconds
Number of milliseconds elapsed during the statement as the sum of (CPUtime, IOtime and locktime). Please note that communications wait is not counted in the elapsed time.

CPU
CPUtime used by the statement.

TotIO
Number of I/O requests issued by the statement.

Buflooks
Number of buffer lookups performed during the statement.

Nrows
Number of rows processed by the statement. For blocked sections, the nrows column may be higher than the number of rows actually processed by the application. For blocked sections, the count shows how many rows meeting the predicate have been passed by DB2/VM to the application in the data block. The application does not necessarily process all these blocked rows.

Occurs
Execution count when multiple identical SQL statements are executed consecutively. For blocked sections, the occurs column may be lower than the number of FETCH calls actually executed by the application. For blocked sections, the count shows how many times DB2/VM filled the block with rows meeting the predicate. These rows are "deblocked" by the DB2/VM Resource Adapter at the client side.

At the end of the trace report, an Index of SQL Statements is printed. It provides the following items for each section in each DB2/VM package traced:

Path index
The index used when accessing the data. None if no index used.

StatementText
The statement text for the static (prepped) package section is retrieved from the SQLCS_SQL_STMNTS table.

13.3.30.6   Sample trace report

```
User        Program   Sect   Statement Started        RunSec    Statistics
_____

TSTAST2     CSTESTB   1      AUXCall   21:10:24.158 0.005       CPU        5
                                                                Tot-IO     0
                                                                Buflook    3
                                                                Nrows      1
TSTAST2     CSTESTB   2      AUXCall   21:10:24.163 0.001       CPU        1
                                                                Tot-IO     0
                                                                Buflook    4
                                                                Nrows      1


_____

                     Index of SQL statements traced
_____

Package CSTESTB Section 1  Path index:  SQLDBA.SQLCS_LOG_I02
                                        +DATABASE +PROGCREA +PROGNAME

SELECT COUNT ( * ) INTO :INT
   FROM SQLCS_LOG
   WHERE DATABASE = :CHAR8


_____

Package CSTESTB Section 2  Path index:   SQLDBA.SQLCS_STMNT_I01
                                        +DATABASE +PROGNAME +PROGCREA

SELECT COUNT ( * ) INTO :INT
   FROM SQLCS_SQL_STMNTS
   WHERE DATABASE = :CHAR8
```

13.3.31   TRACE_SYSAGENT

Syntax : MONITOR TRACE_SYSAGENT { 0 | 1 | 2 }

The TRACE_SYSAGENT command provides a limited dispatcher trace for the DB2 system agents OPERATOR and CHECKPOINT.

To trace the operator agent, enter: TRACE_SYSAGENT 1
To trace the checkpoint agent, enter: TRACE_SYSAGENT 2
To stop tracing, enter: TRACE_SYSAGENT 0

The SYSAGENT trace writes a line to the virtual console of the DB2 server, whenever the system agent enters or leaves the wait state. The output line shows:

A trace timestamp in the format HH:MM:SS.milliseconds.
The new agent state ("Running" if the agent leaves wait, else a wait-state descriptor such as "I/O wait" or "SLD wait").

Using TRACE_SYSAGENT to trace the checkpoint agent

When a checkpoint begins, the agent state is shown as "Running".
The checkpoint agent state will display as "DSPF wait" (in a dataspace system) or "I/O wait" (in a BLOCKIO system) when it is saving the data buffers to DASD, prior to checkpoint initiation.
The checkpoint agent goes to the "ECB wait" state, to wait for the eventual completion of LUW's that delay the checkpoint process.
When the checkpoint effectively begins, the agent state will alternate between "Running", "DSPF wait" and "SLD wait" (if dataspaces are used) or between "Running" and "I/O wait" (in a BLOCKIO system). Use the trace timestamp to monitor the duration of the I/O event (in a dataspace system, DB2 uses the MAPMDISK SAVE function to request CP to save the dataspace pages to the mapdisk).
At the end of the checkpoint, the agent goes into "ECB wait" to wait for the next checkpoint.
If several checkpoints occur while TRACE_SYSAGENT = 2, the trace timestamp will show the time between checkpoints.

Note

While SYSAGENT trace is active, ensure that the DB2 server's console is disconnected, to prevent the screen from going to MORE... status, which would distort the trace timestamp.

13.3.32   UNLOCK DBSPACE

Syntax :
MONITOR UNLOCK <databasename> DBSPACE

The UNLOCK command unlocks all Dbspaces previously locked. The UNLOCK DBSPACE command must be submitted to the SQLCMDM service machine, using SQLMFCMD.

Please note that the automatic SQLCMDM restart at midnight implies a Dbspace unlock.

Example:

SQLMFCMD SQLCMDM MONITOR UNLOCK DBN1 DBSPACE

13.3.33   UNLOCK_DB2

The UNLOCK_DB2 command terminates the wait state forced by the LOCK_DB2 option of the FORCE_CHECKPOINT command.

The command should be entered from SQLMFCMD only and submitted to the database where the FORCE_CHECKPOINT command has been executed.

.

13.4 Using the SQLMFCMD command

SQL/MF provides the SQLMFCMD EXEC to execute a DB2/VM operator command, a CMS, a CP or a MONITOR command from the CMS prompt or a user EXEC. To execute the command, you should have the necessary IUCV privileges to connect to the database server machine.

The syntax of SQLMFCMD is as follows:

    [EXEC] SQLMFCMD [databasename [prefix] host_command]

where:

Databasename:
The VM ID of the database server (not the databasename)

Prefix:
Omitted   :   execute a DB2/VM command
CP   :   execute a CP command
CMS   :   execute a CMS command
MONITOR :   execute a monitor command

Host_command:
The text of the command to execute.

If SQLMFCMD is invoked without arguments, the above fields will be prompted interactively in fullscreen mode. Moreover, SQLMFCMD arguments entered interactively are saved in CMS global variables and will be presented as default values at the next interactive invocation of SQLMFCMD.

Examples

SQLMFCMD DB1 CMS Q DISK executes a CMS command in DB1
SQLMFCMD DB1 SHOW ACTIVE executes a DB2/VM command in DB1
SQLMFCMD DB1 MONITOR CONFIG executes a MONITOR command in DB1
SQLMFCMD requests interactive argument prompt

For a description of the MONITOR command, refer to page 200.

13.5 Issuing a DB2/VM command from CMS

SQL/MF provides the SQLCOM EXEC to execute a DB2/VM operator command from the CMS environment and to receive the command output on the CMS console. The text of the command can be specified as an argument to SQLCOM. If "SQLCOM" is entered alone, the EXEC will prompt for the command to execute. The command operates in the currently connected database, which should be monitored by SQL/MF.

No agent structure is required to execute the operator command. However, the requestor must have IUCV authority to connect to the target database server.

Example   SQLCOM SHOW ACTIVE

14   Statement Recording Facility

14.1 Description

Statement Recording is an intensive tracing facility that notes the execution characteristics of all executing DB2/VM package sections (SQL statements) and checkpoint events into a CMS "recorder" file. Recording can be requested on a **chronological** basis, in which case all SQL during the specified period is recorded. Recording **on demand** can be requested for named users, packages or terminals, using the MONITOR RECORDER START command (see page 213).

In order to achieve acceptable performance, the facility has been implemented as follows.

Recorder Generator

The Generator executes in the DB2/VM server(s) and is part of the SQLCSI program. It stores recorder entries for each SQL statement executed into an unmapped and shared VM dataspace, named "SQLMF#RECORDER#<databasename>". This dataspace is handled as a queue, containing entries for processing by the Recorder Writer.

Recorder Writer

The Writer is an SQL/MF program (SQLCSCRW) executing in a disconnected virtual machine. At a user-defined "polling" interval (expressed in seconds), the writer inspects all recorder dataspaces active in the database servers. Queued recorder entries are extracted from the shared dataspace, transferred to a private dataspace and written to disk from the private dataspace.

The recorder file for a given database is a logical file consisting of up to 16 extents, each extent being a regular varying-length CMS file, named <databasename> #<seqno>. The installation must define in the SQLCS CONFIG dataset, the maximum amount of disk space to be used for writing the recorder file. During writing, recorder file extents are created dynamically and allocated 1/16th of the total space defined. When all allocated space has been used up, the oldest extent is deleted before creating the next one. This technique ensures that the most recent recorder entries are always available. The number of entries kept depends on the allocated space.

To reduce the disk space required by the recorder file(s), you can request data compression, if VM/ESA Version 2 has been installed. The data compression facility is described on page 235.

Recorder File Extract

The Extract function is part of the SQL/MF user interface program SQLCSU. The extract function accesses the Writer's RECORDER file for a given database, applies the user supplied search criteria and stores the selected entries into an "Extract" table, which is created by the function in a private or public DBspace. The Extract table is then processed using the SQL/MF Table Editor. The object names referenced and the statement text of the package sections are inserted during display. All functions of the Table Editor ("Analyze statement", "List statement objects" e.a.) are available during inspection of the recorded data.

Data recorded

The number of data stored in the dataspace and on the RECORDER CMS file depends on the RECORDER LEVEL parameter set in SQLCS CONFIG.

Level 1 recording provides the following data for each executed package section:

- Start date and time (milliseconds)
- LUWID
- DB2/VM Userid
- VM Userid
- User location
- DB2/VM agent number
- Package creator
- Package name
- Package section number
- Database server CPU time in microseconds
- User CPU time in milliseconds
- I/O wait time in milliseconds
- Lock wait time in milliseconds
- Communication wait time in microseconds
- Number of input/output requests
- Number of buffer lookups
- Number of locks (LRB's) currently held by the agent
- Execution count (if a statement is executed multiple times consecutively)
- Statement type from RDIIN
- Number of rows processed
- SQLCODE
- Access path (dbspaceno, table and index id)
- Statement text:

Level-2 recording adds the following counters for each executed package section:

- Number of RDS and DBSS calls
- Number of lock escalations
- Number of pages read
- Number of pages written
- Number of directory buffer lookups
- Number of directory block reads
- Number of directory block writes
- Number of log page reads
- Number of log page writes
- Number of dataspace accesses
- Number of BLOCKIO requests
- Number of DB2 dispatcher calls
- Statement elapsed time in milliseconds
- Number of dataspace pagefaults

Level-3 recording adds the contents of the hostvariables in the WHERE clause of the SELECT statements.

Level-4 recording adds the contents of all hostvariables in the statement, provided that the length of the expanded statement text does not exceed 512 characters. If it does, no substitution occurs.

Notes

- All execution statistics are obtained and stored at the end of the statement, with the exception of user CPU_time and comm_wait time, as explained below.

- "User CPU_time" reflects the CPUtime consumed by the DB2/VM client between the end of the previous SQL statement and the begin of the current one. Statistics are based on the DB2/VM agent structures, which are not related to the VM userid, for which the CPUtime is computed. Therefore, the first statement of an LUW shows user CPUtime zero.

- "Commwait_time" represents the time elapsed between the end of the previous SQL statement and the begin of the current one.

- If the recorder entry represents multiple statements (fetch statements for instance), it shows the sum of the statistics produced during the entire statement sequence.

- When a checkpoint event is recorded, 3 entries are produced: a "schedule" entry, a "start" and an "end" entry. The time between checkpoint scheduling and starting is the checkpoint delay, caused by concurrent agent activity.

- The statement text for prepped statements is obtained at extract time from the SQLCS_SQL_STMNTS table. However, this table does not contain the text for dynamic statements. Therefore, the LOG DYNCOM option will automatically be set while the recorder is active.

14.2 Configuring the Recorder

Operation of the Recording facility is controlled using the RECORDER statement in the SQLCS CONFIG dataset.

Following statements are available:

RECORDER ALLDAYS

By default, recording is not performed on Saturdays and Sundays. If it should, specify the ALLDAYS option.

RECORDER COMPRESS

Specify the statement to enable data compression for the recorder file(s). Do not request compression, unless VM/ESA Version 2 has been installed.

RECORDER CPUTIME=V

Specify the statement to record the virtual user CPUtime. By default, the total user CPUTIME is stored. Total CPU time is the virtual processor time plus the CP overhead to service the virtual machine.

RECORDER DIRECTORY xxxxxxxx

If the RECORDER files of the Writer are kept in an SFS directory, specify the fully qualified directory name, that is, poolid:directoryname. This directory should be accessed by the Recorder Writer in filemode A.

RECORDER DSPSIZE [shared_size] [private_size]

<shared_size> specifies the size of the shared recorder dataspace in 4K pages. If the statement is omitted, a dataspace of 1024 pages (4 megabyte) will be created. The maximum DSPSIZE value allowed is 32767.

<private_size> specifies the size, in 4K pages, of the private recorder dataspace, acquired by the Recorder Writer for this database. If the statement is omitted, <private_size> is set to (<shared_size> * 4). The maximum DSPSIZE value allowed is 32767.

RECORDER EXCLUDE creator.packagename

Excluded packages are recorded only once during a DB2/VM session. This option can be used for packages that are automatically scheduled at regular time intervals and that otherwise would unnecessarily fill up the recorder file. EXCLUDE's are processed by the Recorder Writer.

As many EXCLUDE statements as needed may be specified.

RECORDER FROM hhmm TO hhmm

This optional statement defines the recording period. The period applies to statement recording only. If the period is omitted, no statement recording occurs. Checkpoint and lockwait recording however are always performed as soon as a RECORDER statement has been found.

If statement recording "on demand" is desired, you can:

- Use the MONITOR RECORDER START / STOP command, as described on page 213. This is the preferred method.
- Specify a dummy recorder period, such as RECORDER FROM 0000 TO 0001. When recording is actually needed, alter the recording period by command, for example: MONITOR RECORDER CRTO 2359.

RECORDER INTERVAL n [MILLISEC]

Specifies the writer's polling interval in milliseconds or in seconds (if the MILLISEC keyword is omitted). During polling, the Writer scans the dataspace(s) of the database server(s) subjected to statement recording and writes the extracted entries to disk. The statement is mandatory. To reduce the "working set" of the recorder dataspace, the interval should be low, preferably less than 3 seconds.

RECORDER LEVEL n

Specify LEVEL 1 to record the basic monitor counters.
Specify LEVEL 2 to record all statistical section counters.
Specify LEVEL 3 to include the contents of the hostvariables in the WHERE clause of the SELECT statements.
Specify LEVEL 4 to include the contents of all hostvariables in all SQL statements.

The statement is mandatory.
When using the recorder "on demand" by means of the MONITOR RECORDER command, LEVEL 4 is recommended.

RECORDER MAXSIZE n [%]

This mandatory statement specifies the maximum disk space to be used for the statement recorder file on the A-disk (or SFS directory) of the Recorder Writer. A recorder file consists of 16 extents, each extent being dynamically allocated with a size of (MAXSIZE/16).

The recorder disk space can be specified as an absolute value (as a number of megabytes) or as a percentage (a number followed by a blank and a % sign). In the latter case, the absolute disk space will be computed at startup of the recorder writer, using the number of CMS blocks on its A-disk or SFS directory, leaving a margin of 64 blocks for system files.

Each recorded database has its own recorder file. If multiple databases are subject to recording, multiple recorder files will be maintained, each one requiring (MAXSIZE) space. In these cases, it will be easier to specify MAXSIZE as a percentage. Each database recorder can then be assigned a percentage of space on the Writer's A-disk or SFS directory. However, if the MAXSIZE is specified as a percentage in the global section of SQLCS CONFIG only, and multiple databases are being recorded, the Writer divides the computed size of the A-disk by the number of databases, so that each database receives an equal (maxsize) percent of the A-disk.

RECORDER TIMESLICE shared_timeslice private_timeslice

<shared_timeslice> specifies the maximum time, in milliseconds, that the Recorder Writer is allowed to spend, during each interval, in transferring shared dataspace entries to its private dataspace.

<private_timeslice> specifies the maximum time, in milliseconds, that the Recorder Writer is allowed to spend, during each interval, in transferring private dataspace entries to disk.

Increasing <shared_timeslice> will favor the shared-to-private transfer and tend to delay disk writing. Increasing <private_timeslice> will have the reverse effect. Note however that the shared-to-private transfer is a memory-to-memory process and as such, is faster than the private-to-disk process.

When no RECORDER TIMESLICE statement is supplied, timeslicing is not applied. In this case, the shared-to-private process continues until no more entries are found in the shared dataspace; the private-to-disk process continues until no more entries are found in the private dataspace.

Unless large amounts of data are stored in the recorder dataspace, timeslicing is usually not necessary.

RECORDER WRITER nnn

Specifies the name of the virtual machine running the SQLCSCRW writer program. The statement is mandatory.

Notes

- If the recorder control statements are coded in the global section of SQLCS CONFIG, they enable recording for all databases in the CONFIG file, with identical parameters.

- The RECORDER statements INTERVAL, COMPRESS, WRITER and DIRECTORY must be specified in the global SQLCS CONFIG section:

- Following RECORDER statements may be specified in the global or in a database section of the CONFIG file (or in both): FROM-TO, DSPSIZE, MAXSIZE, TIMESLICE, LEVEL and ALLDAYS

Example

* define the recording parameters for all databases
[global section]
RECORDER FROM 0800 TO 1700
RECORDER WRITER SQLCSCRW
RECORDER INTERVAL 3
RECORDER LEVEL 1
RECORDER DSPSIZE 2048

SECTION DBN1
* define size of recorder for DBN1
RECORDER MAXSIZE 30 %

SECTION DBN2
* define size of recorder for DBN2
* redefine RECORDER LEVEL for DBN2
RECORDER MAXSIZE 60 %
RECORDER LEVEL 3

14.3 Recorder Facility Setup

14.3.1   Database Server Setup

- Since the Statement Recorder creates a dataspace, the database server must execute in XC-mode.
- The recorder dataspace is unmapped. Mapping minidisks are not required.
- If you wish that the user CPU utilization is recorded together with the DB2/VM CPU usage, you must give VM privilege class C or E to the database server. This enables the monitor to extract the CPUtime from the VM/ESA control blocks. If you don't, the user CPU column will be zero.
- XCONFIG statements must be coded in the VM/ESA directory entry. If you are using DB2/VM dataspaces, these requirements have usually been met already. However, check the following:
  the XCONFIG ACCESSLIST statement should provide one additional ALET for the recorder generator

    - the XCONFIG ADDRSPACE statement should provide for one recording dataspace and for its size (which is defined in SQLCS CONFIG) in the MAXNUMBER resp. TOTSIZE statement keywords.

    - the SHARE option must be coded on the XCONFIG ADDRSPACE statement, since the Recorder Generator shares its dataspace with the Recorder Writer (DB2/VM dataspace support does not require the SHARE option)

14.3.2   Recorder Writer Setup

- The Recorder Writer should have the IUCV authority to connect to all database servers for which recording has been activated. Although the Writer does not issue SQL statements, it sends occasional IUCV messages to the servers.

- An IUCV ALLOW statement must be provided in the directory entry of the Writer, to allow the Recorder Generator executing in the database servers, to send IUCV messages to the Writer at database startup or when passing recording-related statements.

- Since the Recorder Writer accesses a dataspace, the virtual machine executing the writer program SQLCSCRW must execute in XC-mode.

- The default directory control statement XCONFIG ACCESSLIST ALSIZE 62 provided by VM/ESA for an XC-machine, is usually appropriate, since the Writer needs two ALET's for each database that is enabled for statement recording.

- The XCONFIG ADDRSPACE directory statement should be supplied so that the recorder writer can create one private dataspace for each recorded database. The statement keywords MAXNUMBER and TOTSIZE should define the number of private dataspaces, resp. their total size (which is defined in SQLCS CONFIG). The SHARE option should not be coded, as the writer creates private dataspaces only.

- Enough disk space must be provided on the A-disk to maintain the desired number of entries in the recorder file. That amount of space is defined using the RECORDER MAXSIZE keyword in SQLCS CONFIG.

14.3.3 Recorder Extract Setup

The user invoking the Recorder Extract function must be able to access the SFS directory (or the 191 minidisk) of the Recorder Writer in order to read the RECORDER file. The necessary LINK (if minidisk) and ACCESS and RELEASE commands to the file are automatically performed during extract.

The user invoking the Recorder Extract function, must have the necessary DB2/VM resources to acquire a private or public DBspace SQLCS_CRX_EXTRACT and to create the Extract table SQLCS_CRX_EXTRACT in the SQL/MF LOG database. The DBspace type (public or private), the number of DBspace pages and the DBspace storage pool is defined by the user during extraction.

Note that recorder extract can also be done in memory, without using a DBspace and without needing the DB2 privileges described in the preceding paragraph.

14.3.4 Dataspace Space estimates

Each section recorded in the dataspace requires 168 bytes. Identical SQL statements executed from the same package section (a series of FETCH statements for instance) require one recorder entry only. The required size of the dataspace mainly depends on how frequently the Writer stores the requests on disk. The queueing algorithm used for the dataspace is such that there is no penalty in over-allocating the dataspace, except for the space taken by the dataspace in the VM/ESA paging area.

14.3.5 File Space estimates

At the begin of the LUW and when package switching occurs during an LUW, a package record is written. The size of this record is 56 bytes.

For each recorded section a section record is written. Its record length is 77 bytes for level-1 recording and 130 bytes for level-2 recording. For level-3 recording, the length, datatype and value of the extracted hostvariables add to the recordlength. Identical SQL statements executed from the same package section require one section record only.

14.4 Starting the Writer

To start the Recorder Writer in a disconnected virtual machine, insert the following statement in the machine's PROFILE EXEC:

SQLCSCRW

14.5 Using the Recorder Compression facility

14.5.1   Description

The Recorder Facility can be requested to compress the data stored on the CMS recorder file. Since recorder compression calls the VM/ESA Data Compression functions, VM/ESA Version 2 must have been installed. Recorder compression is requested by means of the RECORDER COMPRESS statement in SQLCS CONFIG. Compressing the recorder will result in disk savings of 50 percent and more. Recorder data is compressed by the Writer component and expanded during recorder extraction. While compression saves disk space, the additional CPU time used by the compression facility, is significant. If intensive use is made of the recording facility, resulting in a very large recorder file, you may consider recorder archiving as an alternative for compression. The archiving facility is described on page 246.

14.5.2   Building the compression dictionaries

VM/ESA Data Compression uses a variant of the Ziv-Lempel algorithm to replace compressible strings with their index into a compression or expansion dictionary. When calling VM/ESA data compression, an application must provide a compression and an expansion dictionary. SQL/MF comes with standard recorder compression dictionaries called RECORDER ACDICT and RECORDER AEDICT. These dictionaries should normally achieve a compression rate of 50 percent at least. (At the beginning of a new recorder session, the Writer shows the actual compression rate achieved during the previous session in the console spool). By building or re-building your own dictionaries, you can achieve better compression results.

Dictionary (re)building is done as follows:

1) logon the Recorder Writer machine and bypass its profile
2) ensure you have access to the SQL/MF product disk
3) on the CMS prompt, enter the command:

   SQLCSCFB databasename [size]

   where:

   databasename is the CMS filetype of the recorder extent files

   size is the number of records to be stored on the sample scanfile used as input for the dictionary build. It defaults to 6000. The larger n, the more time building will take. More virtual storage may be required too.

   SQLCSCFB performs the following functions:

   - expand the recorder file extents if they were previously compressed (that is, if the RECORDER COMPRESS option is currently stated in SQLCS CONFIG)
   - build the sample scanfile from the first recorder file extent until size is reached
   - build the compression dictionaries using the scanfile
   - erase the previous recorder file extents

4) now, specify the RECORDER COMPRESS statement in SQLCS CONFIG (unless compression was already used before this build run)
5) start the writer

14.5.3   Program checks during compression

The VM/ESA compression facility will generate a data exception program check (CMS abend code C7), when compression services are called using an incorrect dictionary. For example: when expansion is requested using a dictionary different from the one used for compression.

14.5.4   Using the compression facility for other CMS files

The compression facilities used by the recorder component can also be used to compress and expand other CMS files using VM/ESA compression, as follows:

[EXEC] CFCOMP fn ft fm   compresses a CMS file
[EXEC] CFDCOMP fn ft fm   expands a previously compressed CMS file
[EXEC] CFBUILD fn ft fm   builds compression dictionaries for a CMS file

Notes

-    if no compression dictionaries exist for a given file, it will be compressed or expanded using the standard compression dictionaries
-    for program source files and REXX programs, the standard dictionary should give acceptable compression rates by reducing the file to about 30 percent of its non-compressed size
-    issuing CFBUILD will result in still better compression rates but should be done for larger files only, as the compression dictionaries for the file will take about 50 CMS 4K-blocks
-    CFCOMP, CFDCOMP and CFBUILD can also be entered on a CMS filelist
-    a file's compression dictionary is named <fn> ACDICT
-    a file's expansion dictionary is named <fn> AEDICT
-    the standard compression dictionary is named STANDARD ACDICT
-    the standard expansion dictionary is named STANDARD AEDICT

14.6 Using the Recorder Extract function

The extract function is invoked from the LogTable (PF10) interface of the SQLCSU program. When the interface detects RECORDER statements in the SQLCS CONFIG file, it provides an additional menu item Statement Recorder Reports following Benchmark Reports.

Select the Recorder Reports by cursor positioning.

The interface then allows you to enter the recorder file search arguments.

```
 SQL/MF Statement Recorder Extract
---------------------------------------------------

   Enter RECORDER extraction arguments

                 Database =
                     Date =
                Time FROM =
                  Time TO =
             Program name =
                  Section =
                     User =
                 Location =
              CICS taskid =
                     Cost =
                   LUW-id =
           SUM by section =
            DBspace class =
                   Append =
                   Npages =
                  Storpool =
            Insert ComText =
```

The first part of the screen is used to specify the extraction arguments, the second part to define the parameters for acquiring the extract DBspace, if one is used.

Extraction parameters

Database
Required parameter indicating the database whose RECORDER file must be accessed.

Date
Required parameter indicating the date to be extracted in the format yyyymmdd. By default, the parameter is set to the current date.

Time_from
Optional parameter delimiting the file search chronologically. If not specified, searching is from begin of file. Time_from format is hhmmss or -n. The -n format requests the last n minutes on the recorder, that is, the period (current_time minus time_from minutes) to (current_time).

Time_to
Optional parameter delimiting the file search chronologically. If not specified, searching is to end of file. Time_from format is hhmmss. Omit the argument if time_from was specified as a period (-n). If you specify time_from and time_to, the function uses binary search to position the recorder close to the requested time. This reduces the file search time significantly.

Program
Optional parameter to select the recorder entries produced by the named package. If omitted, all packages are selected. A generic package name can be specified by entering a * or % sign at the end of the name.

Section
Optional parameter to select the recorder entries produced by the named package section. If omitted, all sections are selected.

User
Optional parameter to select the recorder entries produced by the named DB2/VM or VM userid. If omitted, all users are selected. A generic username can be specified by entering a * or % sign at the end of the name.

Location
Optional parameter to select the recorder entries produced on the named VTAM or CICS terminal-ID.

CICS_taskid
Optional parameter to select the recorder entries produced by the CICS task taskid.

Cost
Optional parameter selecting the recorder entries whose cost equals or exceeds the specified value. If omitted, no selection on cost is performed.

LUW-id
Optional parameter to select the recorder entries for the specified LUW id.

SUM by section

Optional parameter. Specify Y to summarize the recorder entries per package section number. In this case, the recorder output will be ordered by packagename and section number. If not specified, the recorder entries are shown in detail for each execution of the section.

If multiple selection arguments are supplied, recorder entries are selected only if they satisfy all your search criteria.

Dbspace parameters

DBspace_class

Specifies whether an extract DBspace and table will be used and how it will be acquired:
- NONE   No extract table will be used. The selected recorder data are transferred from the recorder Writer's A-disk to the editor's internal list and processed in storage. This is the default class.
- PRIVATE   A private DBspace will be acquired in the monitor database to create the extract table.
- PUBLIC   A public DBspace will be acquired in the monitor database to create the extract table.

Append

By default, the private extract DBspace is dropped before creating the extract table. Entering Append Y allows you to collect the output of multiple recorder searches into a single extract table. Not applicable when DBspace class = NONE.

Npages

Specify the number of pages to be acquired for the Extract DBspace. Not applicable when DBspace class = NONE.

Storpool

Specify the storage pool number for the ACQUIRE DBSPACE statement. Not applicable when DBspace class = NONE.

Insert_ComText

If Y is specified, the statement text for static package sections recorded will be retrieved from the SQLCS_SQL_STMNTS table while the extract table is being built. This may increase the time needed for the extraction process to complete. If N is specified (which is default), the static statement text is obtained only when a given recorder entry is examined in detail (in "Pagemode") during browsing of the recorder extract. Dynamic statement text however is always retrieved during extraction.

PF keys

PF1
Shows the function help file.

PF3
Quits the selection panel.

PF4
Shows the chronological period available on the recorder dataset.

Note

All parameters entered on the above screen are saved as global CMS variables and represented as defaults at the next invocation of the extract function.

Direct extract

If no extract DBspace / table is requested (DBspace class = NONE), the recorder file is accessed and the selected data are transferred to the editor lists. When selection is completion, the editor list is displayed.

Extracting without a table is considerably faster and the size of the selected data is not limited by the DBspace size, but by virtual storage size only.

However, following restrictions apply during direct extract:

- the selected data can be shown in chronological order only
- the editor's report switching facility cannot be used
- a virtual storage size of at least 6 megabytes is needed
- the maximum number of recorder entries that can be stored in the list is 32000
- the current maximum number of recorder entries depends on the virtual storage size, as follows: (virtual_storage_in_megabytes - 4) * 1024

Using an extract table

After accepting the search arguments, the Extract table is built in the monitor database. Then the recorder file is accessed and the selected recorder entries are transferred to the Extract table. Depending on your recorder selection criteria, this may take some time.

When the table has been built, the Table Editor is invoked and the following menu of Recorder Reports displayed:

RxCHRONO
recorder entries in chronological order

RxCOST
recorder entries ordered by descending cost (BUFLOOK)

RxPROG
recorder entries ordered by packagename

RxUSER
recorder entries ordered by username

RxYOURS
recorder entries ordered by your clause

(the letter x in the report names is replaced with the RECORDER LEVEL you defined for the database)

Select a report using the Execute key PF4 and proceed as with another SQL/MF report.

When you leave report browsing, you will be able to execute another report from the menu, without rebuilding the extract table.

14.7 Sample Recorder Extract Report

```
 SQL/MF Report R2CHRONO                                        Page 8 of 100
 ------------------------------------------------------------------------------
  SQLUSER  TSTAST2           VMUSER  TSTAST2            AGENT   1
    LUWID  425533          LOCATION  T25TT261         CICSTID   0
 PROGCREA  SQLDBA          PROGNAME  SQLCAXC2          SECTION  10
    CTYPE  AUX - DELETE    STARTIME  17:28:47.372       SQLCPU  2
  USERCPU  5                %IOWAIT  13                 %COMMW  86
   %LOCKW  0                BUFLOOK  68               N_LOCKS  44
  X_COUNT  1                  NROWS  22               SQLCODE   0
  RDSCALL  1               DBSSCALL  47              ESCALATE   0
  PAGREAD  3               PAGWRITE  0                DIRBUFL  26
  DIRREAD  0               DIRWRITE  0                LOGREAD   0
 LOGWRITE  1                  TOTIO  4                 DSP_IO   3
  BLOCKIO  0                 %DSPIO  100               %BLKIO   0
 DATABASE  SPRDB2
   ACCESS  Index SQLDBA.SQLCA_RX01 (+QUERYNO +QBLOCKNO
 ------------------------------------------------------------------------------
 DELETE FROM SQLCA_REFERENCE WHERE QUERYNO=:INT




 ------------------------------------------------------------------------------
 PF  1 Help      2 Objects   3 Quit      4 Analyze   5 Format    6 Top
     7 Page <<<  8 Page >>>  9 Bottom   10 Search   11 Hardcopy 12 ListMode
```

14.8 Recorder Commands

The following commands can be used to control the Recording facility during execution. These commands can be issued from the DB2/VM console, using the SQLMFCMD command on the CMS prompt, or via the Host Command interface of the SQLCSU program.

MONITOR CRSTOP

Stops the Recorder Generator in the designated database. The Writer is stopped indirectly as no more recorder entries are stored in the dataspace.

MONITOR CRSTART

Restarts the Recorder Generator in the designated database.

MONITOR CRFROM hhmm
MONITOR CRTO hhmm

Modifies the recording period in the designated database. This command may be used for statement recording "on demand" by specifying a null period in the SQLCS CONFIG file (RECORDING FROM 0000 TO 0001) and by modifying the TO value (e.g. CRTO 2359) when recording is needed.

MONITOR CRCOUNT

Displays following statistical counters for the shared recorder dataspace:

-   The total number of recorder entries stored during this database session. The counter is roughly equivalent to the number of DB2/VM package sections executed.
-   The total number of recorder entries currently processed by the writer. The difference with the preceding counter is the number of entries currently in queue.
-   The total number of recording requests flushed because the recorder dataspace was full.
-   The number of dataspace pages in the "free" queue.
-   The number of dataspace pages in the "request" queue.
-   The maximum number of dataspace pages ever in the "request" queue.

Displays following statistical counters for the private recorder dataspace:

-   The total number of entries stored during this database session.
-   The total number of entries currently processed by the writer.
-   The total number of entries flushed because the dataspace was full.
-   The number of dataspace pages in the "request" queue.
-   The maximum number of dataspace pages ever in the "request" queue.

Note

("free pages " + "request pages" + 1) should equal your RECORDER DSPSIZE.
Even when no requests are queued (entries stored = entries processed), the request queue will not be empty, because processed pages are reclaimed at the next recording request.

MONITOR CREXEC command-text

The command requests the Recorder Writer to perform a CP, CMS command or a user EXEC at the next write interval. The facility can be used for a variety of purposes, for example, to off-load a RECORDER file, to erase the recorder file and so on. The command should contain the prefix CP for executing CP commands, the prefix CMS for executing CMS commands and the prefix EXEC for executing an EXEC. The command-text can be sent to any database server subject to statement recording. The server will then transmit the message to the Writer.

14.9 Batch recorder extract

The **SQLCSCR** EXEC can be invoked from a user EXEC or from the CMS prompt, to extract specified events from the SQL/MF recorder. Output of this function always goes to the DB2 table SQLCS_CRX_EXTRACT (or SQLCS_CRL_EXTRACT when extracting from the lock recorder).

After the statement recorder extract table has been built, the user has the option of executing additional SQL clauses on the SQLCS_CRX_EXTRACT table, by using the OPT_CLAUSE argument.

Syntax:

```
SQLCSCR
   TYPE { CR | LR }
   DATABASE n
   DATE yyyymmdd
   TIMEFROM hhmm
   TIMETO hhmm
   PACKAGE n
   USER n
   LOCATION n
   CICSTID n
   COST n
   LUW n
   SUM { Y | N }
   APPEND { Y | N }
   NPAGES n
   STORPOOL n
   DBSPTYPE { PUBLIC | PRIVATE }
   OPT_CLAUSE <CMS-filename>
```

TYPE
Specify CR to extract from the Statement Recorder file. Specify LR to extract from the Lockwait recorder.
Default is CR.

DATABASE
Specify the name of the database for which the recorder must be inspected.

DATE
Specify the date of the entries to be extracted. Default is the current data

TIMEFROM
TIMETO
Specify the time period to be extracted. If omitted, all entries for **DATE** will be inspected.

PACKAGE
If specified, only the recorder entries for the named package are extracted.

USER
If specified, only the recorder entries for the named DB2 or VM userid are extracted.

LOCATION
If specified, only the recorder entries for the named terminal are extracted.

CICSTID
If specified, only the recorder entries for the named CICS taskid are extracted.

COST
If specified, only the recorder entries for the statement equal to or exceeding the specified cost are extracted.

LUW
If specified, only the recorder entries for the named LUWID are extracted.

APPEND
APPEND Y will append the extracted recorder entries to a previous selection on the extract table. Default is N.

SUM
SUM Y will summarize the recorder entries per package section, thus providing one row per section in the output extract table. Default is N.

NPAGES
Specify the number of pages to acquire for the output extract DBspace. Default is 512.

STORPOOL
Specify the storage pool for the output extract DBspace. Default is 1.

DBSPTYPE
Specify the type of the output extract DBspace. Default is PRIVATE.

OPT_CLAUSE

Specify the filename for the file <filename> **SQLCSCR** *. This file contains WHERE, ORDER or GROUP clauses to be applied to the SQLCS_CRX_EXTRACT table after build. CMS pipes are used to incorporate the user file. As a results the clauses are free-format. No semicolon should be supplied at the end of the clauses.

Notes

- DATABASE is the only required SQLCSCR argument, all other arguments may default.
- When using the OPT_CLAUSE, sufficient space is required on the A-disk to create a temporary file that contains the table rows satisfying the user clauses.

14.10   Archiving the Recorder

Instead of allocating a considerable amount of disk space to the recorder file, an installation may decide to archive the recorder to tape or cartridge. The archiving facility may also be used to maintain backup recorder files, so that recorded data can be kept for a longer period of time.

14.10.1   Archive Processor

The Recorder archiving function is performed by a disconnected virtual machine that executes the SQLCSARC EXEC. The EXEC is invoked from the CMS profile. Its only invocation argument is the number of archive extents to process in one archive pass. The default value is 8, that is, the maximum number of extents on the recorder file divided by 2.

SQLCSARC must be able to link to the 191 minidisk of the Recorder Writer SQLCSCRW.

SQLCSARC inspects the recorder file extents every 30 seconds, to determine whether enough new recorder extents (15 by default) have been created to initiate an archive. If so, these extents are written to a labelled tape or cartridge, using QSAM.

Archiving does not delete the recorder extents from disk, where they may be needed for the extract function. The oldest extents will be deleted by the recorder writer, when the RECORDER MAXSIZE is reached, or when the A-disk is full.

14.10.2   ArchiveTape Exit

Before writing to tape, SQLCSARC calls the user exit SQLCSARX. This exit is responsible for tape/cartridge mounting and dismounting, possibly by calling the tape management system in use at the customer's installation.

SQLCSARX is called with the 2 arguments function and tape label

-   "tapelabel"   has the format "yyyymmdd_databasename" and is used to identify the tape
-   "function"   has one of the following values on entry:

    -   "CREATE" Archiving occurs for the first time of the day; hence, tape yyyymmdd_database does not exist and must be created by requesting a scratch tape from the tape manager. The HDR1 tape label will be created during QSAM write.
    -   "APPEND" Archiving has already been performed and the tape yyyymmdd_database must be extended. It should be recalled from the tape manager for output. The generated FILEDEF will specify the DISP MOD option.
    -   "OPEN"   The function is called during recorder extract to mount tape yyyymmdd_database for input.
    -   "CLOSE"   Called when reading or writing to the tape/cartridge has finished. The tape volume and drive should be released.

A model SQLCSARX with the filetype EXECS is delivered with SQL/MF. The customer should copy the file to filetype EXEC and insert his tape managing commands. The existence of the SQLCSARX EXEC also determines whether recorder extract will attempt to open an archive tape.

14.10.3   Extracting recorder data from tape or cartridge

The recorder extract function will attempt to read data from an archive tape under the following conditions:

When it is found that archiving has been enabled because the customer has copied or renamed the SQLCSARX EXECS to SQLCSARX EXEC.

When a TIME_FROM search criterion is specified and the requested date or time period is not available on disk.

An archive tape will never be accessed when the "Time_from" search argument has been omitted.

Data can extracted from disk only, from tape only or from tape and disk, depending on the date/time period specified.

14.11   Recording: Operational notes

When the recorder dataspace is full, the recording generator issues a message on its console (once per database session only) and disables recording. Recording resumes automatically when space becomes available, because the writer has processed queued requests. You can issue the CRCOUNT command, described above, to check for dataspace full conditions. If "dataspace full" occurs regularly, you should either enlarge the dataspace (RECORDER DSPSIZE statement) or decrease the RECORDER WRITER INTERVAL (so that queued data get written out faster).

When DB2/VM shutdown has been requested and recording data are still present in the dataspace, the Recorder Generator will issue a message and wait until all pending recording requests have been processed. However, when the recorder notices that the Writer is not functional (because entries do not get processed), the wait sequence is given up after 30 seconds.

15   Lockwait Recording Facility

15.1 Description

The facility registers the event where an agent is put in lockwait by DB2/VM. Following data are stored for each lockwait event:

- the name of the database
- the locked DB2/VM and VM userid
- the locked agent number
- the terminal name of the locked user
- the CICS taskid of the locked user, if appropriate
- the package name and section number of the locked command
- the type of the locked command (OPEN, FETCH etc.)
- the LUWID of the locked user
- the time of the lockwait as HH.MM.SS.MMM
- the duration of the lock in milliseconds
- the number of locks and long locks already held by the locked user
- the mode of the wanted lock as:
  ```
  -  IS    intention share lock
  -  IX    intention exclusive lock
  -  S     share lock
  -  SIX   share and intention exclusive lock
  -  X     exclusive lock
  -  U     update lock
  ```
- the duration of the wanted lock as:
  ```
  -  INST  state of a lock is being tested
  -  SHORT lock held until end of DBSS call
  -  LONG  lock held until end of LUW
  -  USER  lock held until end of LUW or explicit release
  ```
- the type of the wanted lock as:
  ```
  -  SYSTEM      system lock
  -  DATABASE    database lock
  -  DBSPACE     DBspace lock
  -  TABLE       table lock
  -  INDEX-KEY   index key lock
  -  INDEX-PAGE  index page lock
  -  TABLE-PAGE  table page lock
  -  TABLE-ROW   table row lock
  ```
- the name of the wanted DBspace
- the VMID and terminal name of the user holding the lock
- the package name and section of the command holding the lock
- the text of the locked command

15.2 Lockwait Recorder Setup

The lockwait recorder is part of the Command Recording facility, which must be setup as described on page 232. The recorded lockwaits are stored in the dataspace of the Command Recorder and stored by the Recorder Writer in a separate lockwait recorder file.

15.3 Configuring the Lockwait Recorder

Lockwait recording is enabled by coding, in the SQLCS CONFIG dataset, the command

RECORDER TRACELOCKS n [%]

Specify n as the disk space to be allocated to the lockwait recorder file on the A-disk of the Recorder Writer.

The recorder disk space can be specified as an absolute value (as a number of megabytes) or as a percentage (the space value is followed by a blank and a % sign). In the latter case, the absolute disk space will be computed at startup of the recorder writer, using the number of CMS blocks on its A-disk and leaving a margin of 64 blocks for system files.

For example, when TRACELOCKS is specified as 10 %, the space in bytes will computed as: ((CMS_blocks-64)*CMS_blocksize)*0.10

The disk space required depends on the expected number of lockwaits during the period you want to keep, multiplied by 153 (the length of the lockwait disk record).

The RECORDER FROM - TO statement, although required, applies to statement recording only: lockwait events are always recorded. If lockwait recording only is required, the statement recorder must be configured, but can be disabled by specifying a dummy recording period such as FROM 0000 TO 0001.

15.4 Inspecting Recorded Lockwaits

The lockwait extract function is invoked from the LogTable (PF10) interface of the SQLCSU program. When the interface detects a RECORDER TRACELOCKS statement in the SQLCS CONFIG file, it provides an additional menu item Lockwait Recorder Report following the Statement Recorder Reports item. Select the Lockwait Recorder Report by cursor positioning.

A search criteria panel is then displayed. Specify the criteria and continue processing the report, as described for the Statement Recorder Extract function on page 237.

While the statement recording extract function creates the table SQLCS_CRX_EXTRACT, the name of the lockwait extract table is SQLCS_CRL_EXTRACT.

15.5 Recorded Lockwait space estimates

Each recorded lockwait requires 157 bytes in the recorder dataspace and on the lockwait recorder file.

16   AutoPrep Facility

16.1 Functional description

The AutoPrep facility automatically generates DB2/VM packages for frequently executed dynamic statements and replaces the dynamic statement sequence with a static sequence that invokes the generated package.

AutoPrep avoids the resource consumption (CPU usage mainly) that results from repeated access path selection during the SQL PREPARE statement. It also decreases contention on the system catalogs accessed during prepare.

AutoPrep is available for packages that use parameter markers (e.g. REXX/SQL) and for packages that do not (e.g. ISQL). [20] Because execution frequency cannot be determined adequately for non-parameterized statements, such statements are converted to the parameterized format, by replacing the constant values with the parameter marker ?.

---

[20] A parameterized statement executes as follows:
- EXEC SQL PREPARE 'SELECT ... FROM <table> WHERE <column> = ?'
- EXEC SQL OPEN ... USING <value>

A non-parameterized statement uses the following sequence:
- EXEC SQL PREPARE 'SELECT ... FROM <table> WHERE <column> = value'
- EXEC SQL OPEN ...

16.2 Operation

When a dynamic statement is executed by a package that is eligible for AutoPrep (as stated in the AutoPrep control file), following operations are performed:

-   If the statement has not been autoprepped previously, the AutoPrep facility proceeds as follows:

    -   a non-parameterized statement is converted to the parameterized format
    -   the statement text is reduced to a 32-byte hash value
    -   the execution frequency for the hash is incremented
    -   if the frequency equals the frequency defined in the AutoPrep control file, an AutoPrep request is forwarded to the AutoPrep server program SQLCSAPS, which executes in a dedicated virtual machine
    -   the AutoPrep server assigns a package name, builds a package source and preps the source, using SQLPREP; the generated package name has the format $xxxxxxx, where xxxxxxx is a sequence number assigned by the server
    -   after successful prep, the server updates the SQLMF_AUTOPREP table and stores the relationship between the statement hash and the newly generated package

-   If the statement hash occurs in the AUTOPREP table, the statement has been autoprepped previously. The SQL statement flow is then modified in such a way, that the autoprepped package is executed and not the dynamic statement. The SQL PREPARE statement is no longer executed for such statements.

-   The AUTOPREP table maintains the last access date for all AutoPrep packages. If a package is not used for 60 days (because the statement text has been modified for instance), it is automatically dropped.

16.3 Installation

The AutoPrep facility must be installed in all databases eligible for AutoPrep and in the database containing the SQL/MF tables (the "monitor database"). Installation in the monitor database is a technical requirement. If AutoPrep will not be used effectively in this database, you can acquire a minimal 128-pages DBspace.

16.3.1 Create the AutoPrep table

Prerequisites

The default create command stream requires a DBspace of 2048 pages. This provides for about 8000 AutoPrep package rows. The actual space requirements in the AUTOPREP DBspace depend on the length of the dynamic statement text, the number of columns fetched and the number of columns in the SELECT predicate. The SQLMF_AUTOPREP table must be created in each database eligible for AutoPrep.

Interactive table creation

Enter the command **SQLCSAPT** on the CMS prompt. The command will prompt for:

- the name of the databaseserver for which autoprep should occur
- the password of user SQLBDA in the above database
- the name of the virtual machine that will execute the AutoPrep server program SQLCSAPS

Subsequently, the DBSU create command stream will be shown in an XEDIT session. In the XEDIT session, fill in the STORPOOL parameter of the ACQUIRE DBSPACE statement. If needed, modify the PAGES parameter. Issue an XEDIT file. This will acquire the AutoPrep DBspace and create the tables with their indexes.

Non-interactive table creation

Enter the command:

> **[EXEC] SQLCSAPT DBNAME a DBAPASS b APSERVER c STORPOOL d PAGES e**

where:

a = name of databaseserver eligible for autoprep
b = the password of user SQLDBA in that database
c = the name of the virtual machine running the SQLCSAPS program
d = the storage pool for the AutoPrep DBspace
e = the number of pages for the AutoPrep DBspace

16.3.2   Update the database server directory

The AutoPrep facility uses a 16-Mb dataspace for internal operations. Therefore:

- The database server machine should execute in XC mode.
- The XCONFIG ACCESSLIST statement should provide for one additional ALET (the VM/ESA default is 62).
- The XCONFIG ADDRSPACE statement arguments MAXNUMBER and TOTSIZE should provide for the AutoPrep dataspace.

Autoprep storage requirements

The Autoprep facility keeps control information in the AutoPrep list for each autoprepped SQL statement and for each grant issued on the autoprepped package to a user other than the user that initiated autoprep. The Autoprep list resides in main storage. The length of the list entries depends on:

- the length of the dynamic statement text
- the length of the input and the output SQLDA, that is, the number of columns selected and the number of predicate values (or parameter markers) in the WHERE clause, each column descriptor taking 44 bytes

Multiply the average length of the autoprep list entries (which will probably be between 2 and 3 KB) with the number of entries in the list. Add the result to the server's virtual storage definition in the VM directory.

The SHOW DBSPACE <DBspno_of_SQLMF_AUTOPREP> command may also be used to estimate the space required for the autoprep list, as (4K * number of occupied data pages).

16.3.3   Setup the AutoPrep server

- Setup a virtual machine for the server and provide 16 Mb of virtual storage and a small 191 (say 5 3390 cylinders). The server does not need to be an XC machine.
- Provide an IUCV ALLOW statement, to enable the database servers to forward AutoPrep messages.
- Ensure that the server has access to the DB2/VM and SQL/MF product disks.
- In the PROFILE EXEC of the machine, insert the command **EXEC SQLCSAPS**. This starts the AutoPrep server program.

16.4 Configuring the AutoPrep facility

The AutoPrep facility for a given database is controlled by means of a CMS file named <databaseservername> AUTOPREP. Create such a file for each database eligible for AutoPrep. The file normally resides on the SQL/MF product disk.

Following statements are provided in the AUTOPREP file:

SERVER statement

- Syntax : **SERVER VM_id**
- For <VM_id> substitute the name of the virtual machine that runs the **SQLCSAPS** program.

PREP statement

- Syntax : **PREP packagename AFTER frequency**

- For <packagename>, specify the name of the package that generates dynamic statements eligible for autoprepping. A generic packagename is indicated by terminating the name with a % sign.

- For <frequency>, specify a number between 1 and 255. Automatic statement prep will occur after a given statement text has been executed <frequency> times a day. Once AutoPrep has occurred, frequency checking for that statement is no longer performed.

- Multiple PREP statements may be provided, if needed.

AUTOPREP Example

The AutoPrep server SQLCSAPS should prepare all dynamic statements generated by QMF and by packagenames starting with SQLL, when these statements have been executed 10 times on the same day in the database server SQLPR1. This is achieved by coding the control file **SQLPR1 AUTOPREP** as follows:

    SERVER SQLCSAPS
    PREP SQLL% AFTER 10
    PREP DSQ% AFTER 10

16.5 Security Considerations

- DBA authority is needed for the AutoPrep server. It is granted automatically when installing the AutoPrep facility. However, the GRANT DBA statement issued does not specify a password. As a result, connecting this DB2 userid is possible from the AutoPrep server machine only.

- When a statement is autoprepped for the first time, the AutoPrep server grants the RUN privilege on the package to PUBLIC. Since AutoPrep occurs only after a successful SQL PREPARE, it has been established that the user initiating AutoPrep has all necessary table privileges.

- When another DB2 user successfully issues an SQL PREPARE for the same statement text, the user has the required table privileges and will be allowed to use the public autoprep package on subsequent executions of the statement. In its AUTOPREP table, the facility records all users that performed a successful PREPARE of the SQL statement.

- The AutoPrep packages, which have been granted to PUBLIC, can only be invoked under control of the AutoPrep facility.

16.6 Restrictions

- AutoPrep support is limited to SELECT statements. Update, insert, delete statements are **not** handled.

- SELECT statements that do not specify a table creator will not be autoprepped.

- Wen a statement has been autoprepped and is executed using the AutoPrep package, a DB2 prepare is no longer executed. Hence, the SQL statement cost estimate (returned in the prepare SQLCA) is no longer available. SQL/MF returns a dummy cost of 1.

- Autoprepped statements always execute with "cursor stability" isolation (you can use the **SET ISOLATION UR FOR $*** configuration statement if uncommitted read isolation is desired).

17   Monitor Tables

17.1 SQL Statements

SQLCS_SQL_STMNTS Table

```
DATABASE    CHAR(8)     Database name
PROGCREA    CHAR(8)     Package creator
PROGNAME    CHAR(8)     Package name
SECTION     SMALLINT    Section nr of the statement within the package
PLAN        SMALLINT    Each table access by the package section receives a
                        plan number, starting from 1. If multiple tables are
                        accessed,  or  if  the  a  table  participates  in  a
                        referential  constraint,  multiple  rows  will  be
                        provided, each with its own plan number.
MODLEVEL    SMALLINT    Modification level of the statement
                        0=last modification level
                        -1=previous modification level
                        -2 etc. (up to CONFIG MODLEVEL parameter)
SAMPLED     INTEGER     Total number of times the statement has been executed,
                        since the package was prepped.
CPUTIME     INTEGER     CPU time used by the statement in milliseconds
LOCKTIME    INTEGER     Lock wait time in the statement in milliseconds
IOWAIT      INTEGER     I/O wait time the statement in milliseconds
COMMWAIT    INTEGER     Communications  wait  time  in  the  statement  in
                        milliseconds
RDSCALL     INTEGER     RDS calls by the statement
DBSSCALL    INTEGER     DBSS calls by the statement
DISPCALL    INTEGER     DB2/VM dispatcher calls by the statement
ESCALATE    INTEGER     Number of lock escalations in the statement
WAITLOCK    INTEGER     Number of lock waits in the statement
DEADLOCK    INTEGER     Number of deadlocks the statement
NROWS       INTEGER     Number of table rows processed by the statement
BUFLOOK     INTEGER     Number of buffer lookups by the statement
PAGREAD     INTEGER     Number of page reads by the statement (including
                        dataspace reads)
PAGWRITE    INTEGER     Number of page writes by the statement
DIRBUFL     INTEGER     Number of directory buffer lookups by the statement
DIRREAD     INTEGER     Number of directory page reads by the statement
                        (including dataspace reads)
DIRWRITE    INTEGER     Number of directory page writes by the statement
LOGREAD     INTEGER     Number of log reads by the statement
LOGWRITE    INTEGER     Number of log writes by the statement
TOTIO       INTEGER     Number of I/O's by the statement
INT_DBSP    INTEGER     Number of internal Dbspace accesses by the statement
DSFAULT     INTEGER     Number of dataspace pagefaults by the statement
BLOCKIO     INTEGER     Number of *BLOCKIO requests by the statement
DOWNER      CHAR(8)     Dbspace owner
DNAME       CHAR(18)    Dbspace name accessed
TCREATOR    CHAR(8)     Table creator
TNAME       CHAR(18)    Table name accessed
ICREATOR    CHAR(8)     Index creator
INAME       CHAR(18)    Index name accessed
BLOCKED     CHAR(1)     Y if the package statement executes with blocking; N
                        otherwise.
ISOL        CHAR(2)     The statement's isolation level as RR for repeatable
                        read isolation, CS for cursor stability and UR for
                        uncommitted read.
SEL         CHAR(1)     Y for selective index access, N for non-selective
                        access and blank for non-indexed access.
```

```
INDEXCOL    CHAR(40)       Index column definition for INAME
LAST_ACC    TIMESTAMP      Last time sampled
COMM_TEXT   VARCHAR(8192)  Text of the statement
```

17.2 Statement Monitor Log Table

SQLCS_LOG Table

```
DATABASE     CHAR(8)      Database name
USERNAME     CHAR(8)      VM-name of user serviced
AGENT        SMALLINT     DB2/VM agent number
PROGCREA     CHAR(8)      Creator of the package
PROGNAME     CHAR(8)      Name of the package
SECTION      SMALLINT     Package section number
PREPFLAG     SMALLINT     Set to 1 when package prepped
COMM_TYPE    CHAR(8)      Type of statement (open, fetch etc.)
ISOLATION    CHAR(2)      The statement's isolation level as:
                          RR for repeatable read isolation
                          CS for cursor stability
                          UR for uncommitted read
BLOCKED      CHAR(1)      Blocking attribute as Y or N
SEL          CHAR(1)      Y for selective index access
                          N for non-selective index access
                          blank for non-indexed access
INDCREA      CHAR(8)      Creator of the index used in the
                          statement or blank
INDNAME      CHAR(18)     Name of the index or blank
                          (if access without index)
BEGDATE      DATE         Start date of statement execution
BEGTIME      TIME         Start time of statement execution
ENDDATE      DATE         Stop date of statement execution
ENDTIME      TIME         Stop time of statement execution
ELAPSED      TIME         Elapsed statement time
CPUTIME      INTEGER      CPU time used by the statement
                          in microseconds
LOCKTIME     INTEGER      Total lockwait time in milliseconds
IOWAIT       INTEGER      Total IOwait time in milliseconds
COMMWAIT     INTEGER      Total Communication wait time
                          in microseconds
RDSCALL      INTEGER      Number of RDS calls
DBSSCALL     INTEGER      Number of DBSS calls
DISPCALL     INTEGER      Number of DB2/VM dispatcher calls
ESCALATE     INTEGER      Number of lock escalations
WAITLOCK     INTEGER      Number of lock requests that
                          resulted in wait
DEADLOCK     INTEGER      Number of deadlocks
NROWS        INTEGER      Number of table rows processed
BUFLOOK      INTEGER      Number of page buffer lookups
PAGREAD      INTEGER      Number of page reads
PAGWRITE     INTEGER      Number of page writes
DIRBUFL      INTEGER      Number of directory buffer lookups
DIRREAD      INTEGER      Number of directory page reads
DIRWRITE     INTEGER      Number of directory page writes
LOGREAD      INTEGER      Number of log page reads
LOGWRITE     INTEGER      Number of log page writes
TOTIO        INTEGER      PAGREAD+PAGWRITE+DIRREAD+DIRWRITE
                          +LOGREAD+LOGWRITE
```

```
INT_DBSP      INTEGER      the number of buffer lookups
                           in internal DBspaces
DSFAULT       INTEGER      Number of dataspace pagefaults
                           (when DB2/VM dataspaces installed)
BLOCKIO       INTEGER      Number of *BLOCKIO requests
                           (when DB2/VM dataspaces installed)
LOCATION      CHAR(8)      Terminal_ID where statement executed
CICSTID       INTEGER      CICS taskid of statement executed
LUWID         INTEGER      LUWID of statement executed
P_USERCPU     INTEGER      Total user CPU time in the package
                           (if LOGTYPE P)
P_CHKPTIME    INTEGER      Total checkpoint wait in package
                           (if LOGTYPE P)
P_TOT_ELAPS   INTEGER      Total elapsed time in package
                           (if LOGTYPE P)
P_MAX_ELAPS   INTEGER      Maximum elapsed time in package
                           (if LOGTYPE P)
P_TOT_ROWS    INTEGER      Total nr of rows processed by package
                           (if LOGTYPE P)
LOGTYPE       CHAR(1)      B benchmark log
                           P package statistics log
                           X exception log
LOGREASON     CHAR(8)      For logtype X the reason for logging
                           (Dbspace scan, TOTIO etc.)
LOGSTAMP      CHAR(13)     Value of hardware clock at start
                           of logged statement
COMM_TEXT     VCHAR(8K)    Text of the statement.
                           For a static statement, the text is
                           retrieved from the SQL_STMNTS table.
                           There is no text for section=0
                           statements (such as CONNECT).
```

17.3 System Monitor Tables

TABLE SQLMF_SESSION_CTR
SQL Session Counters (one row per day and per database)

```
DATABASE    CHAR(8)
MONDATE     DATE
RDSC        INTEGER     # RDS calls
DBSSC       INTEGER     # DBSS calls
LUWS        INTEGER     # LUW's
ROLLB       INTEGER     # Rollback's
CHKP        INTEGER     # Checkpoints
LOCKL       INTEGER     # Long locks
ESCAL       INTEGER     # Lock escalations
WAITL       INTEGER     # Locks resulting in wait
DEADL       INTEGER     # Deadlocks
PAGBUF      INTEGER     # Page buffer lookups
PAGRD       INTEGER     # Page reads
PAGWR       INTEGER     # Page writes
DIRBUF      INTEGER     # Directory buffer lookups
DIRRD       INTEGER     # Directory page reads
DIRWR       INTEGER     # Directory page writes
LOGR        INTEGER     # Log page reads
LOGW        INTEGER     # Log page writes
DASDR       INTEGER     # Reads
DASDW       INTEGER     # Writes
TOTIO       INTEGER     DASDR + DASDW
DSPF        INTEGER     # Dataspace pagefaults
```

TABLE SQLMF_COUNTS
SQL Counters

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
RDSC            INTEGER     # RDS calls
DBSSC           INTEGER     # DBSS calls
LUWS            INTEGER     # LUW's
ROLLB           INTEGER     # Rollback's
CHKP            INTEGER     # Checkpoints
LOCKL           INTEGER     # Long locks
ESCAL           INTEGER     # Lock escalations
WAITL           INTEGER     # Locks resulting in wait
DEADL           INTEGER     # Deadlocks
PAGBUF          INTEGER     # Page buffer lookups
PAGRD           INTEGER     # Page or dataspace reads
PAGWR           INTEGER     # Page or dataspace writes
DIRBUF          INTEGER     # Directory buffer lookups
DIRRD           INTEGER     # Directory page reads
DIRWR           INTEGER     # Directory page writes
LOGR            INTEGER     # Log page reads
LOGW            INTEGER     # Log page writes
DASDR           INTEGER     # Reads
DASDW           INTEGER     # Writes
TOTIO           INTEGER     DASDR + DASDW
PHIT            DEC(5,2)    Page buffer hit ratio (see page 376)
DHIT            DEC(5,2)    Directory buffer hit ratio (see page 376)
LRBS            SMALLINT    # LRB's defined
LRBU            SMALLINT    # LRB's used
MAXLUW          SMALLINT    Max # of LRB's used in one LUW
CP_CPU          INTEGER     CPU time (in seconds) used by the database server
                            during the interval
CP_PAGRD        INTEGER     Number of VM page reads for the database server
                            during the interval
CP_PAGWRT       INTEGER     Number of VM page writes for the database server
                            during the interval
CP_PAGWS        INTEGER     The storage working set (computed by VM/ESA) of the
                            database server at the interval
N_CPSAVE        INTEGER     Number of times dataspace pages were saved during
                            interval as a result of the SAVEINTV parameter
P_CPSAVE        SMALLINT    Average number of pages saved per request
WAITSLD         SMALLINT    Number of times database had to wait for a save
                            block
N_PREFETCH      INTEGER     Number of dataspace page prefetches during the
                            interval
P_PREFETCH      SMALLINT    Average pages on the prefetch list
N_RELPAGE       INTEGER     Number of times dataspace pages were released
                            during interval as a result of the TARGETWS
                            parameter
P_RELPAGE       INTEGER     Average number of pages released per request
VMCPU           SMALLINT    Total VM CPU usage at interval
VMPAG           SMALLINT    VM paging rate at interval
VMEXP           SMALLINT    VM expansion factor
VMSTOR          SMALLINT    VM storage utilization
BLKIO_DUR       INTEGER     Duration of a Block I/O request in microseconds
DSPF_DUR        INTEGER     Duration of a dataspace pagefault in microseconds;
                            if DB2 has performed prefetching, multiple pages
                            are retrieved by a single pagefault and duration is
```

		for all pages in the prefetch block; therefore, DSPF_DUR should be divided by P_PREFETCH to get the average time required to page-in one dataspace page
DISPDELAY_I	INTEGER	Dispatch delay for interactive agents in microseconds
N_DELAY_I	INTEGER	Nr of times dispatching delay for interactive agents was computed
DISPDELAY_NI	INTEGER	Dispatch delay for non-interactive agents in microseconds
N_DELAY_NI	INTEGER	Nr of times dispatching delay for non-interactive agents was computed
CHKP_DUR	INTEGER	Average duration of a checkpoint in milliseconds
CHK_DELAY	INTEGER	Number of checkpoint delays
DELAY_DUR	INTEGER	Average duration of a checkpoint delay in milliseconds
MAX_PROGS	SMALLINT	# Slots in the DB2 package cache
PROGS_LOADED	SMALLINT	Packages loaded
PROGS_IN_USE	SMALLINT	Packages loaded for active agents
TOT_PROGS	INTEGER	Packages monitored in the DB2 session
TOT_AGENTS	SMALLINT	Maximum number of agents active
DSHIT	DEC(5,2)	Dataspace hit ratio for all pools (see page 376)
DSAHIT	DEC(5,2)	Dataspace access hit ratio for all pools (see page 376)
R_RATIO	SMALLINT	Percentage of PAGRD within (PAGRD+PAGWR)

TABLE SQLMF_DSP_COUNTS
Dataspace Counters

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
POOL            CHAR(5)     poolnumber:
                            DIR if directory dataspace
                            IDBSP if internal DBspace
TYPE            CHAR(3)     DSP if dataspaces, BLK if BlockIO
BUFLOOK         INTEGER     buffer lookups during interval
DSREAD          INTEGER     dataspace reads during interval
DSWRITE         INTEGER     dataspace writes during interval
DSFAULT         INTEGER     dataspace faults during interval
IUCVBIO         INTEGER     BlockIO requests during interval
DSHIT           DEC(5,2)    dataspace hit ratio for pool
DSAHIT          DEC(5,2)    dataspace access hit ratio for pool
TARGWS          SMALLINT    Target working set specification
CURWS           SMALLINT    Current working set
SAVEINT         SMALLINT    Save_interval specification
```

TABLE SQLMF_DBSP_COUNTS
DBspace Buffer Usage Counts

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
DBSPACENAME     CHAR(18)
PBUSE           SMALLINT    # Page buffers used by this DBspace
DBUSE           SMALLINT    # Directory buffers used by this DBspace
TPBUSE          SMALLINT    # Page buffers used by all DBspaces
TDBUSE          SMALLINT    # Directory buffers used by all DBspaces
PPBUSE          SMALLINT    PBUSE / TPBUSE ratio
PDBUSE          SMALLINT    DBUSE / TDBUSE ratio
PAGBUF          SMALLINT    # Page buffers in pool
DIRBUF          SMALLINT    # Directory buffers in pool
```

TABLE SQLMF_DBXT_COUNTS
Storage Pool Buffer Usage Counts

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
POOL            SMALLINT
PBUSE           SMALLINT     # Page buffers used for this storage pool
DBUSE           SMALLINT     # Directory buffers used for this storage pool
TPBUSE          SMALLINT     # Page buffers used for all storage pools
TDBUSE          SMALLINT     # Directory buffers for all storage pools
PPBUSE          SMALLINT     PBUSE / TPBUSE ratio
PDBUSE          SMALLINT     DBUSE / TDBUSE ratio
PAGBUF          SMALLINT     # Page buffers in pool
DIRBUF          SMALLINT     # Directory buffers in pool
```

TABLE SQLMF_DBSP_USE
DBspace Usage Counts

```
DATABASE      CHAR(8)
MONDATE       DATE
MONTIME       TIME
MMDD          CHAR(4)
HHMM          CHAR(4)
DBSPACENAME   CHAR(18)
HDRP          INTEGER      # Header pages
HPU           INTEGER      # Header pages used
HFS           SMALLINT     % Header pages free
HPUP          SMALLINT     % Header pages used
DATAP         INTEGER      # Data pages
DPU           INTEGER      # Data pages used
DFS           SMALLINT     % Data pages free
DPUP          SMALLINT     % Data pages used
INDEXP        INTEGER      # Index pages
IPU           INTEGER      # Index pages used
IFS           SMALLINT     % Index pages free
IPUP          SMALLINT     % Index pages used
```

TABLE SQLMF_DBXT_USE
Storage Pool Usage Counts

```
DATABASE      CHAR(8)
MONDATE       DATE
MONTIME       TIME
MMDD          CHAR(4)
HHMM          CHAR(4)
POOL          SMALLINT
NPAGES        INTEGER         # Pages in pool
USED          INTEGER         # Pages used
FREE          INTEGER         # Pages free
PUSED         INTEGER         % Pages used
SOS           CHAR(1)         * If short on storage condition did arise
```

TABLE SQLMF_USER_STATE
User Status Table

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
USERID          CHAR(8)         Username
STATE           CHAR(4)         User waitstate
LOCKS           SMALLINT        # Locks held by user
LONGL           SMALLINT        # Long locks held by user
WANTL           CHAR(4)         Type of lock wanted
WANTDB          SMALLINT        DBspaceno wanted
WANTDBNAME      CHAR(18)        DBspacename wanted
HOLDUSER        CHAR(8)         User holding lock
REQM            CHAR(3)         Lock request mode
DURAT           CHAR(5)         Duration of the lock
```

TABLE SQLMF_LOG_USE
SQL Logfile Usage Counts

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
LOGSIZE         INTEGER         Size of logfile in bytes
USED            INTEGER         Logfile bytes used
PUSED           SMALLINT        % Logfile used
POFLOW          SMALLINT        % remaining before overflow
PCHCK           SMALLINT        # Pages remaining before checkpoint
```

TABLE SQLMF_CHKP_DELAY
Checkpoint Delay Table

```
DATABASE        CHAR(8)
MONDATE         DATE
MONTIME         TIME
MMDD            CHAR(4)
HHMM            CHAR(4)
USERID          CHAR(8)      Username active
STATE           CHAR(4)      Wait state of user
RWAPPL          CHAR(1)      R/W application indicator
AGENT           SMALLINT     User's agent number
PROGNAME        CHAR(8)      Name of package executed
SECTION         SMALLINT     Package section number
```

TABLE SQLMF_CONNECTS
Connections Status Table

```
DATABASE       CHAR(8)
MONDATE        DATE
MONTIME        TIME
MMDD           CHAR(4)
HHMM           CHAR(4)
CONNECTED      SMALLINT    # Users connected
ACTIVE         SMALLINT    # Users in LUW
WAITING        SMALLINT    # Users waiting for agent structure
INACTIVE       SMALLINT    # Users connected and not in LUW
AGENTS_AV      SMALLINT    # Agent structures available
CONNECT_AV     SMALLINT    # Connections available
```

TABLE SQLMF_DBS_AVG
DBspace Buffer Usage Average Counts

```
DATABASE        CHAR(8)
MONDATE         DATE
MMDD            CHAR(4)
DBSP            VARCHAR(18)
TPBU            INTEGER         Average # of page buffers used
TDBU            INTEGER         Average # of directory buffers used
PPBU            SMALLINT        Average % of page buffers used
PDBU            SMALLINT        Average % of directory buffers used
```

TABLE SQLMF_STP_AVG
Storage Pool Buffer Usage Average Counts

```
DATABASE        CHAR(8)
MONDATE         DATE
MMDD            CHAR(4)
POOL            SMALLINT
TPBU            INTEGER        Average # of page buffers used
TDBU            INTEGER        Average # of directory buffers used
PPBU            SMALLINT       Average % of page buffers used
PDBU            SMALLINT       Average % of directory buffers used
```

18 Customizing SQL/MF

18.1 Writing log report files

A log report file is a CMS file with a filetype of SQLCSMAC. The file can be stored on any minidisk or directory accessed when the SQLCSU table editor is invoked. (The editor searches for files with filetype SQLCSMAC using the standard CMS search order.) The CMS filename of the report file stands for the name of the report. This name is displayed on the log report menu, along with the report descriptor, which is taken from the first record of the file.

The filename of the log reports delivered with SQL/MF starts with the letter B for the benchmark log reports, with the letter L for the exception log reports, with the letter P for the package statistics reports, with the letter R for the statement recording reports and with the letter S for the statement statistics reports. It is recommended that you adhere to these conventions when creating your report file.

To create the report, you can use a standard report as a model, such as the BSTAMP or LSTAMP SQLCSMAC reports.

```
Benchmark Log in chronological order
<DEFAULT> &SEL_DATE_FROM CURRENT DATE
<DEFAULT> &SEL_DATE_TO   CURRENT DATE
<DEFAULT> &ORDER_BY      BEGDATE, BEGTIME
T
SQLDBA
SQLCS_LOG
<ENTER>
<PF5>
SELECT  DATABASE, USERNAME, AGENT, --
        PROGCREA, PROGNAME, SECTION, COMM_TYPE, --
        BEGDATE, BEGTIME, ENDDATE, ENDTIME, ELAPSED,  --
        ISOLATION, INDCREA, INDNAME, --
        CPUTIME, LOCKTIME, IOWAIT, COMMWAIT, --
        DBSSCALL, DISPCALL, ESCALATE, WAITLOCK, --
        BUFLOOK, PAGREAD, PAGWRITE, --
        DIRBUFL, DIRREAD, DIRWRITE, --
        LOGREAD, LOGWRITE, TOTIO, --
        COMM_TEXT --
        FROM SQLDBA.SQLCS_LOG --
        WHERE BEGDATE BETWEEN &SEL_DATE_FROM --
        AND &SEL_DATE_TO  --
        AND LOGTYPE = 'B' --
        AND &OPT_CLAUSE --
        ORDER BY &ORDER_BY
9999
<ENTER>
```

Coding rules

On the first line of the report file, code a description of the report. The length of the descriptor should not exceed 62 characters.

Do not modify or omit the lines printed in bold in the above example, unless you want to limit the number of rows fetched. If so, replace 9999 with the maximum number of lines in the report. Specify the maximum as a 4-digit value, padded to the left with zeroes.

Code your SELECT statement between the lines containing <PF5> and 9999. A variable number of blanks may be inserted at any point in the SELECT. These redundant blanks are removed before execution. The length of the statement - after compression of blanks - should not exceed 960 characters. The length of the uncompressed statement should be inferior to 1600 characters. To continue lines, insert the characters -- after the last word of the line. Leave at least one blank between that last word and the continuation mark.

Statement variables

(1) The SELECT statement may contain variable symbol names, identified by an & at the begin of the name. Variable symbols are automatically extracted by the editor and the user is automatically prompted for a replacement value. The replacement cannot be longer than 56 characters.

(2) You can pre-assign variable symbols using the <DEFAULT> statement. This statement names the variable symbol and assigns a default value. The default value is shown on the screen during variable symbol assignment. The user may eventually modify it.

(3) A line that contains an unassigned symbol is not generated. (In the example, the line AND &OPT_CLAUSE). These optional clauses must be coded in such a way, that valid SQL syntax is maintained when line generation is bypassed.

(4) The system variable named &DATABASE receives the name of the default database, when the report is invoked.

The FORMAT statement may be used to modify the name of the column or SELECT expression in the header of the report list. The syntax of the FORMAT statement is as follows:

FORMAT COL x NAME nnnn

where

X
Stands for the number of the column or expression in the SELECT column list. X may be specified as * to denote the column number of the previous FORMAT, incremented by 1.

NNN
Stands for the name of the column or expression in the list header.

If specified, the FORMAT statements should appear at the end of the report file, following the last <ENTER> statement. They should be coded by ascending column (COL) number.

Example:

FORMAT COL 10 NAME Name_10
FORMAT COL * NAME Name_11

18.2 Writing print report files

The facilities for customizing the online reports described in the previous section, may also be used for controlling the printing function of SQL/MF, that is, the SQLCSPRT program. SQLCSPRT uses control files with a filetype of SQLCSPRT to generate the reports. All the syntax rules defined above for coding SQLCSMAC control files also apply to print control files.

By default, SQLCSPRT prints 4 columns on a printline. It expects that the length of the column name is no longer than 10 characters and that the length of the column data is inferior to 20 characters.

The program however provides a facility to control the number of table columns printed on a given line. Coding the constant '//' in the SELECT list of the SQLCSPRT control file will perform a 'new line' function.

For example, to obtain the following print page layout

```
COL1   COL2   COL3   COL4
COL5   COL6   COL7
COL8   COL9   COL10  COL11
```

insert the expression '//' between COL7 and COL8 in the SELECT column list.

Also note that you can use the FORMAT statement, described above, to give a name to report columns or expressions.

18.3 Writing a CONNECT exit for the Statement Monitor

During explicit user connect, the Statement Monitor invokes the EXEC SQLCSUXT, when found on a minidisk or directory accessed by the database server.

SQLCSUXT entry arguments:

- databasename
- DB2_userid performing the CONNECT   (primary authorization ID)
- DB2_userid specified on the CONNECT (secondary authorization ID)
- "duplicate userid" flag (0 or 1). The value 1 is passed when the newly connected DB2 userid has already been connected by another agent. This flag may be used to disallow connection of duplicate userid's.
- name of package issuing the connect
- pointer to the connect password

SQLCSUXT returncode:

0   Proceed with CONNECT
8   Cancel CONNECT : SQL/MF changes the connection parameters to all question marks, which will result in SQLCODE -561.

SQLCSUXT processing

The exit may perform any processing needed to verify the connect. However, since the exit runs in the database server, no SQL statements can be executed.

The password argument is passed as a pointer, to allow for eventual modification.

The password value is obtained using the REXX STORAGE function, as follows:
password=storage(password_pointer,8)

The password can be modified using the REXX STORAGE function, as follows:
x=storage(password_pointer,8,"new_password_value")

Sample SQLCSUXT

A sample connect exit (called SQLCSUXT SAMPLE) is provided with SQL/MF. It performs the following functions:

- Display the date, time, userid and terminal ID of the connect request on the database server's console.
- Deny connect if the user is not found in the CMS file called <databasename> CONNECTS.
- Report such illegal connects on the console.

 The <database> CONNECTS file has the following record layout:

```
col1    primary_userid - CHAR(8)
col10   secondary_userid - CHAR(8)
```

The CONNECTS file should specifically state all the secondary userid's that the primary userid is allowed to connect to. If the combination (primary userid + secondary userid) is not found, connect is denied.

Note:
- To activate a modified version of the user exit, a DB2/VM restart is required.
- You may modify and test the user exit while the server is using it, since the server takes a copy of the SQLCSUXT EXEC in storage and invokes the storage copy.

18.4 Writing a "begin_of_session" exit for the System Monitor

At the beginning of a new session (which occurs at midnight), the System Monitor program SQLMFM calls the user exit SQLMFBOS. A dummy REXX exit is distributed with the product. The installation may insert its own processing for the initialization of the new monitoring session.

18.5 Configuring the [S]MSGROUTE target machine

When the MSGROUTE or SMSGROUTE statement is present in the SQLCS CONFIG file, severe SQL/MF error messages are sent to the virtual machine defined in the [S]MSGROUTE statement.

The MSGROUTE statement causes these error messages to be transmitted to the target via a CP MSG command. The SMSGROUTE statement uses the CP SMSG command for message transmission.

The disconnected MSGROUTE target displays the messages received on its console. If message collection software is running in this machine, additional actions may be triggered on reception.

The disconnected SMSGROUTE target should execute, via its PROFILE, the SQLCSMSG EXEC, which is delivered with SQL/MF. SQLCSMSG obtains the SMSG message sent and calls the SQLMFMSG EXEC, with the following arguments:

-    the VMid of the component originating the message
-    the message text (which starts with a message number in most cases)

The SQLMFMSG EXEC delivered with SQL/MF, simply displays the call arguments on the console. If more extensive message handling is required, it can be inserted by the customer in this EXEC. (You should take a local copy of this exec, as installation of new SQL/MF releases will re-install the default exec).

19   User Attached Process Facility

19.1 Purpose

The facility allows installation written EXECs to retrieve and process the data items available within the SQL Monitoring Facility. Such user written extension is called an UAP (User Attached Process) and is called by SQL/MF components at various stages of monitoring. An UAP is a REXX EXEC named SQLMUAPn with n designating the SQL/MF exit point calling the process. At initialization of the SQL/MF System and Statement Monitors, the existence of the individual SQLMUAPn EXECs is checked. If an EXEC is found on the CMS search chain, it is brought in storage (by means of an EXECLOAD command) and invoked during subsequent processing. This procedure allows you to modify (and test) the UAPs while an active copy is running. The Statement Monitor can be requested to reload a modified UAP3 EXEC by issuing the MONITOR CONFIGURE command. (See Monitor Commands on page 200).

The monitor passes data related to the particular process to the UAP as REXX variables, assigned by SQL/MF before calling the UAP. Depending on the contents of these variables, the UAP may initiate the necessary processing, including SQL access. The SQLMUAP3 and SQLMUAP4 EXEC's may set a returncode at exit. This returncode is kept by the Monitor in the agents monitor block until end of LUW. Its value is represented in the variable USERFLAG when these EXECs are subsequently called. The SQLMFCMD EXEC may be called from an UAP, in order to execute a DB2/VM, a CP, a CMS or a MONITOR command.

Skeleton SQLMUAP EXECs are delivered with a filetype of "EXECS". To activate a given attachment, insert the functional REXX coding after the UAP prologue in the corresponding sample and change the filetype to "EXEC".

In the above cases, the UAP facility is invoked from SQL/MF. An application may also directly call the UAP facility to obtain monitor data during execution. The SQLMUAPI interface program is provided for this purpose, as described below.

19.2 UAPs called by the System Monitor

SQLMUAP1
Global DB2/VM System Performance Counters

SQLMUAP2
Storage Pool Usage Counters

SQLMUAP4
Locked Agent Status

SQLMUAP5
DB2/VM Logfile Usage

SQLMUAP6
DB2/VM Connections Status

SQLMUAP7
Checkpoint or Log_archive Delay

The above UAP's are called by the SQL/MF system monitor program SQLMFM, at the interval defined by the installation in the SQLMFIN CTL configuration dataset. The call is performed only when the corresponding SQLMUAP EXECs exist. At a given interval, the SQLMUAP1 is invoked once for each monitored database. All other UAP's may be called multiple times per monitored database. SQLMUAP4 for instance, will be called for each locked DB2/VM agent in each monitored database.

19.3 UAPs called by the Statement Monitor

When found, SQLMUAP3 (Active Agent Status) is called by the SQL/MF statement monitor program SQLCS for all agents in LUW, at the SQLCS scan interval. This interval can be defined at SQLCS startup. If omitted, the interval is set to 30 seconds. SQLMUAP3 is also called by the statement monitor when a statement has been stored in the SQLCS_LOG table as a result of a LOG parameter defined in the SQLCS CONFIG dataset. The variable LOGREASON is provided in this case and indicates the reason for the logging operation.

When found, SQLMUAP8 and SQLMUAP9 are called by the SQL/MF statement monitor when a DB2 checkpoint begins resp. ends.

When found, SQLMUAPP is called by the SQL/MF statement monitor each time package statistics are received from SQLCSI running in the database server. These statistics are sent at the STAT_FREQ period (every hour by default). They represent the total resource consumption by the package during the previous STAT_FREQ period.

When found, SQLMUAPR is called by the SQL/MF statement monitor when a dynamic short-on-storage condition is detected in a storage pool, that is, when the number of pages written to the pool since the last checkpoint is lower than (2 * current_number_of_free_pages in the pool). The EXEC may issue a message to inform the DBA that the pool should possibly be extended, or it can force a checkpoint (using the MONITOR FORCE_CHECKPOINT command).

When found, SQLMUAPX is called by the SQL/MF governor function before applying a restricting condition. Before forcing the agent, the EXEC is invoked with the entry arguments, described later in this chapter on page 304.

Note

If an auxiliary statement monitor has been declared (using the AUX_MONITOR statement in the SQLCS CONFIG dataset), the EXECs SQLMUAP8, SQLMUAP9, SQLMUAPR and SQLMUAPX are called by the auxiliary monitor.

19.4 Description of UAP input variables

19.4.1   SQLMUAP1: DB2/VM performance counters

The following DB2/VM performance counters are available to the SQLMUAP1 process as interval related values, that is, as indicators of resource consumption during the SQLMFM sampling interval.

```
DATABASE    Databasename sampled
DATE        Monitoring date in SQL format
TIME        Monitoring time in SQL format
RDSC        RDS calls during the interval
DBSSC       DBSS calls during the interval
LUWS        LUW's initiated during the interval
ROLLB       Rollback's during the interval
CHKP        Checkpoints during the interval
LOCKL       Long locks during the interval
ESCAL       Lock escalations during the interval
WAITL       Locks resulting in wait during the interval
DEADL       Deadlocks during the interval
PAGBUF      Page buffer lookups during the interval
PAGRD       Page reads during the interval
PAGWR       Page writes during the interval
DIRBUF      Directory buffer lookups during the interval
DIRRD       Directory page reads during the interval
DIRWR       Directory page writes during the interval
LOGR        Log page reads during the interval
LOGW        Log page writes during the interval
DASDR       DASD Reads during the interval
DASDW       DASD Writes during the interval
TOTIO       Total I/O operations during the interval
PHIT        Page buffer hit ratio (see page 376)
DHIT        Directory buffer hit ratio (see page 376)
LRBS        Number of lock request blocks defined
            (DB2/VM initialization parameter NLRBS)
LRBSU       Number of lock request blocks defined per user
            (DB2/VM initialization parameter NLRBU)
LRBU        Number of LRB's currently  in use
MAXLUW      Maximum number of LRB's used in one LUW
VMCPU       VM CPU usage obtained using the IND LOAD CP
            command
VMPAG       VM paging rate
VMEXP       VM expansion factor
VMSTOR      VM storage utilization
DSHIT       Dataspace hit ratio for all pools
            (see page 376)
DSAHIT      Dataspace access hit ratio for all pools
            (see page 376)
RRATIO      Read ratio as (PAGRD*100)/(PAGRD+PAGWR)
NCPSAVE     number of times SAVEINTV became effective
PCPSAVE     average number of pages saved at SAVEINTV time
WAITSLD     number of times database had to wait on an
            SLD block during save
```

```
NREFPAGE    number of prefetch requests executed
            during interval
PREFPAGE    average number for pages prefetched per request
NDIAG10     number of times dataspace pages were
            released due to TARGETWS
PDIAG10     average number of pages released per request
CPCPU       CPU time (in seconds) used by database
            server during interval
CPPAGRD     VM page reads performed for server
            during interval
CPPAGWRT    VM page writes performed for server
            during interval
CPPAGWS     VM working set of database at interval
```

Note

Although the hit ratio's PHIT, DIRHIT, DSHIT and DSAHIT have DECIMAL(5,2) format, no decimal point will be present in the corresponding REXX variables, presented to SQLMUAP1.

19.4.2   SQLMUAP2: Storage pool counters

The following DB2/VM storage pool counters are available to the SQLMUAP2 process. They are presented at each monitoring interval for each storage pool defined.

```
DATABASE        Databasename sampled
DATE            Monitoring date in SQL format
TIME            Monitoring time in SQL format
POOL            Pool number
NPAGES          Number of pages in pool
USED            Number of pages used
FREE            Number of pages free
PUSED           Percentage of pages used
SOS             Set to * (asterisk) when the short on storage condition
                did arise for the storage pool
```

19.4.3   SQLMUAP3: Active agent status

The following variables are available to the SQLMUAP3 process. They are presented by the Statement Monitor:

- at the SQLCS scan interval for each agent in logical unit of work in each database monitored
- when an SQL statement has been written to the exception log table

DATABASE	Databasename sampled
VM_NAME	The VM name of the user executing the statement.
SQL_NAME	The SQL_id of the user executing the statement.
AGENT	The number of the DB2/VM agent (1 being the first user agent)
AGNTSTAT	The status of the DB2/VM agent. See table on page 97.
CREATOR	The creator of the package containing the executing statement or "N/A" if no package used (during a DRDA CONNECT or COMMIT for example)
PACKAGE	The name of the package containing the statement or "N/A" if no package used
SECTION	The section number of the package containing the statement
LOCATION	The terminal where the agent executes or "DSC" for disconnected machines.
COMMAND	The type of the statement. See table on page 98.
BLOCKED	The statement's blocking attribute as "Y" or "N".
ISOLAT	The isolation level of the statement as: "C" for cursor stability "R" for repeatable read "W" for uncommitted read
RW_STATE	The agent's read/write status: Y for an R/W agent, N for an R/O agent.
STRTDATE	The start date of the statement in SQL date format
STRTTIME	The start time of the statement in SQL time format
STOPTIME	The stop time of the previous statement (if the agent is in communication wait) in SQL time format
ELAPSED	The time elapsed since STARTTIME as hh:mm:ss.
LUWID	The LUW identification assigned by DB2/VM. The LUWID has the INTEGER format.
LUW_STRT	The start time of the LUW in SQL time format.
LUW_TIME	The time elapsed since LUW_START as hh:mm:ss.
LOCKTIME	If the statement is in the lockwait state, the number of seconds passed since the begin of the lock wait.
CPUTIME	The total CPU time used by the database server for executing the statement, in microseconds.
LOCKWAIT	The total number of milliseconds the user has been in the lockwait state during the current statement.
IOWAIT	The total time spent in input/output wait during the current statement, in milliseconds.
COMMWAIT	The total time spent in communication wait during the current statement, in microseconds.
IDLEWAIT	The time elapsed since the end of the previous SQL statement, in milliseconds. Idlewait is non-zero only when the agent is in LUW with no SQL statement in progress.
NROWS	The number of rows processed by the statement.

RDSCALL	The number of calls to the Relational Subsystem during the statement.
DBSSCALL	The number of calls to the Database Subsystem.
DISPCALL	The number of DB2/VM dispatcher calls during the statement.
BUFLOOKS	The number of buffer lookups performed by the statement.
LOCKS	Number of locks currently held by the agent.
WAITLOCK	The number of times a lock request resulted in wait.
ESCALATE	The number of times a lock request resulted in lock escalation.
PAGREAD	The number of page reads during the statement, that is, the number of times a request for data could not be satisfied from the buffer pool. The statistic includes reads from the dataspaces if applicable.
PAGWRITE	The number of page writes during the statement. The statistic includes writes to the dataspaces if applicable.
LOGREAD	The number of reads to the DB2/VM log during the statement.
LOGWRITE	The number of writes to the DB2/VM log during the statement.
DIRBUFL	The number of directory buffer lookups performed.
DIRREAD	The number of directory page reads during the statement. The statistic includes reads from the directory dataspace if applicable.
DIRWRITE	The number of directory page writes during the statement. The statistic includes writes to the directory dataspace if applicable.
TOTIO	The sum of PAGREAD, PAGWRITE, DIRREAD, DIRWRITE, LOGREAD, LOGWRITE and INTDBSP.
INTDBSP	The number of accesses to internal Dbspaces during the statement.
DSFAULTS	The number of dataspace pagefaults during the statement, if DB2/VM dataspaces have been installed.
BLOCKIO	The number of VM IUCV BlockIO requests submitted by the database manager to CP during the statement.
I_ACCESS	"1" if an index is being used, "0" if no index is being used.
COMMTYPE	"STATIC" or "DYNAMIC"
LREASON	Describes the reason for logging, as:

	Blank	if not called by logging component
	"BUFLOOK"	log due to BUFLOOK exceeded
	"CHKPWAIT"	log due to checkpoint wait (ELAPSED column contains the duration of the wait)
	"DBSPSCAN"	log due to Dbspace scan
	"DEADLOCK"	logging a deadlock
	"DYNCOM"	logging a dynamic statement
	"ESCALATE"	log due to a lock escalate event
	"ISOLATN"	log due to isolation = 'repeatable read'
	"LUW_TIME"	log due to LUW_RESPONSE exceeded
	"NOBLOCK"	log due to prepped without blocking
	"RESPTIME"	log due to statement RESPONSE exceeded
	"SQL -nnn"	log due to SQLCODE nnn
	"TOTIO"	log due to TOTIO exceeded

USERFLAG	Flag reserved for the SQLMUAP3 EXEC. (See note 5 below.)

Following variables are setup for agents in lockwait only (AGNTSTAT='LOCKWAIT'):

LONGL	Number of long locks held by the user	
WANTL	Type of lock wanted:	
	DB	lock on database
	DBSP	lock on entire dbspace
	IKEY	lock on indexkey
	IPAG	lock on index page
	PAGE	lock on data page
	ROW	lock on data row
	SYS	system lock
	TABL	lock on an entire table
WANTDB	DBspacenumber wanted	
WANTDBN	Dbspacename wanted	
HOLDVMID	VM-id of the user holding the lock	
HOLDUSER	DB2-id of the user holding the lock	
HOLDAGNT	Number of agent holding the lock	
HOLDPACK	Name of package holding the lock	
HOLDSECT	Package section number holding the lock	
REQM	Mode of lock held	
	SIX	share and intention exclusive lock
	IS	intention share lock
	IX	intention exclusive lock
	S	share lock
	X	exclusive lock
DURAT	Duration of lock held	
	INST	lock being tested
	LONG	lock held until end of LUW
	SHORT	lock held until end of DBSS call
	MED	lock held until end of LUW or explicit release

Notes:

- The above REXX variables contain the resource consumption for the current SQL statement or section (if a cursor-based statement).

- If the process needs the text of the SQL statement being executed or the name of the first plan index used, it should call the interface routine SQLMFCTX which will setup the following REXX variables:

  - CMND_TXT : the running statement text, with hostvar references substituted with their contents; the maximum length of this variable is 8192
  - ICREATOR : the creator of the plan index
  - INAME : the name of the plan index
  - ICOLS : the first 40 characters of the index column definition of INAME

- Retrieving the text of a static statement will cause an SQL SELECT on the table SQLCS_SQL_STMNTS. In addition, if the statement text in that table is not existing or not current, a package unload will be performed automatically.

- Prior to calling the SQLMUAP3 EXEC, the Statement Monitor sets up SQLMFCTX as a CMS nucleus extension, which should called by the SQLMUAPn EXEC as a CMS command.

- The USERFLAG is a binary fullword variable in the Monitor block for the agent. The variable is reserved for the UAP. It is set from the exit returncode of the SQLMUAP3 EXEC and is represented in "USERFLAG" when the EXEC is called again. The flag is shared with the SQLMUAP4 EXEC. It is reset to zero by the Statement Monitor at end of LUW.

19.4.4   SQLMUAP4: Locked agent status

The following variables are available to the SQLMUAP4 process. They are presented by the system monitor at each monitoring interval for each user in the lockwait state.

```
DATABASE        Databasename sampled
DATE            Monitoring date in SQL format
TIME            Monitoring time in SQL format
AGENT           Agent number in lockwait state
USERID          Username in lockwait state
LOCKS           Number of locks held by the user
LONGL           Number of long locks held by the user
WANTL           Type of lock wanted:
                DB      lock on database
                DBSP    lock on entire dbspace
                IKEY    lock on indexkey
                IPAG    lock on index page
                PAGE    lock on data page
                ROW     lock on data row
                SYS     system lock
                TABL    lock on an entire table
WANTDB          DBspacenumber wanted
WANTDBN         Dbspacename wanted
HOLDUSER        Name of user holding the lock
HOLDAGNT        Number of agent holding the lock
REQM            Mode of lock held
                SIX     share and intention exclusive lock
                IS      intention share lock
                IX      intention exclusive lock
                S       share lock
                X       exclusive lock
DURAT           Duration of lock held
                INST    lock being tested
                LONG    lock held until end of LUW
                SHORT   lock held until end of DBSS call
                MED     lock held until end of LUW or explicit release
USERFLAG        Flag reserved for the SQLMUAP4 EXEC. (See note 1 below.)
```

Notes:

1) The USERFLAG is a binary fullword variable in the Monitor block for the agent. The variable is reserved for the UAP. It is set from the exit returncode of the SQLMUAP4 EXEC and is represented in "USERFLAG" when the EXEC is called again. The flag is shared with the SQLMUAP3 EXEC. It is reset to zero by the Statement Monitor at end of LUW.

2) SQLMUAP4 may obtain the complete monitor data for the locked (or locking) agent by issuing the CMS command

   SQLMFUAG <database> <agent_nr>

   This call will setup all the REXX variables described above for the SQLMUAP3 EXEC. To obtain the monitor data for the locking agent for example, the following command should be issued: 'SQLMFUAG' database holdagnt.

   After the call, the SQLMUAP3 variable PACKAGE contains the locking package name.

   After calling SQLMFUAG for an agent, SQLMFCTX may be called to obtain the executing statement text. The plan indexname however cannot be obtained at this point.

3) A locked agent is also presented to SQLMUAP3, if that EXEC is enabled.

19.4.5   SQLMUAP5: DB2/VM Log usage

The following variables are presented by the System Monitor at each monitor interval for each monitored database:

```
DATABASE         Databasename sampled
DATE             Monitoring date in SQL format
TIME             Monitoring time in SQL format
LOGSIZE          Size of the logfile in bytes
USED             Number of logfile bytes used
PUSED            Percentage of logfile bytes used
POFLOW           Percentage of logfile space remaining before overflow
PCHCK            Number of pages remaining before checkpoint
```

19.4.6   SQLMUAP6: DB2/VM Connections

The following DB2/VM counters are available to the SQLMUAP6 process. They show the state of the DB2/VM connections at the monitor sample time.

```
DATABASE        Databasename sampled
DATE            Monitoring date in SQL format
TIME            Monitoring time in SQL format
CONNECTD        Number of users connected
ACTIVE          Number of users in LUW
WAITING         Number of users waiting for an agent structure
INACTIVE        Number of users connected and not in LUW
A_AGENT         Number of agent structures available
A_CONNCT        Number of connections available
```

19.4.7   SQLMUAP7: Checkpoint delays

The following variables are available to the SQLMUAP7 process when a checkpoint or log archive delay occurs in one of the monitored databases.

DATABASE	Databasename sampled
DATE	Monitoring date in SQL format
TIME	Monitoring time in SQL format
USERID	Username active and possibly causing the checkpoint delay
VMID	VM userid active at checkpoint delay
STATE	Wait state of that user
	"RUN" executing
	"COMM" in communication wait:
	"I/O" waiting for input/output completion
	"LOCK" waiting for a database resource
	"DBUF" waiting for a directory buffer
	"PBUF" waiting for a page buffer
	"DSPF" waiting for pagefault completion
	"SLD" waiting for save block
	"CHKP" waiting on checkpoint completion
RWAPPL	"Y if application is read-write; N otherwise
AGENT	User's agent number
PACKAGE	Name of package executed
SECTION	Package section number

Note

At checkpoint delay, all active agents are passed to the UAP. Agents that are not delaying the checkpoint are in state CHKP. Agents with another state are delaying the checkpoint.

Example

Forcing read-only agents that delay a checkpoint:

```
if (STATE <> 'CHKP') & (RWAPPL = 'N') then do
   'EXEC SQLMFCMD' database 'FORCE' agent
   say database 'agent' agent 'package' package ,
        'section' section 'has been forced by SQLMUAP7'
end
```

19.4.8   SQLMUAP8: Start of Checkpoint

The EXEC is called when a DB2 checkpoint starts. The name of the database taking the checkpoint is passed as the EXEC argument. No other variables are presented to the exit.

19.4.9   SQLMUAP9: End of Checkpoint

The EXEC is called when a DB2 checkpoint ends. The name of the database is passed as an argument to the EXEC. No other variables are presented to the exit.

19.4.10   SQLMUAPC: Host Command processing

The SQLMUAPC EXEC is called when a CP, CMS, DB2 or MONITOR command is entered from the SQLCSU Host Command interface or from the SQLMFCMD EXEC.

Following arguments are passed to SQLMUAPC at invocation:

- VM-id of the user forwarding the command
- VM-id of the target DB2 server
- text of the command

On exit, SQLMUAPC should set one of the following return codes:

0 process the input command
4 discard the input command and process the CP, CMS, DB2 or MONITOR command(s) stored by SQLMUAPC in the CMS stack
8 discard the input command

Sample exit

```
/* Sample SQLMUAPC EXEC */

address command
arg userid db2_id command

w1=word(command,1)
w2=word(command,2)

if w1="SHOW" & w2="ALL" then do

   /* Provide the SHOW ALL command */

   queue "SHOW ACTIVE"
   queue "SHOW CONNECT"
   queue "SHOW LOG"
   queue "COUNTER POOL *"
   exit 4

end

if w1="SHOW" & w2="DBSPACE" then

   /* Disallow SHOW DBSPACE commands */

   Exit 8

exit 0
```

19.4.11 SQLMUAPP: Package Statistics

The EXEC is called when the Statement Monitor receives the monitor statistics for a given package.

The following variables are available to the SQLMUAPP process:

```
DATABASE    Databasename
VM_NAME     The VM name of the user executing the package.
CREATOR     The creator of the package.
PACKAGE     The name of the package.
SAMPLED     The number of times packages statistics were stored during
            the statistics interval. This number corresponds to the
            number of LUW's performed by the package.
LASTDATE    The last execution date of the package in SQL date format.
LASTTIME    The last execution time of the package in SQL time format.
CPUTIME     The total CPU time used by the database server for executing
            the package, in milliseconds.
USERCPU     The CPU time, in milliseconds, used by the package in the
            DB2 client machine. Requires VM privilege class C or E in
            the database server; otherwise, zero is stored.
TOTELAPS    The total elapsed time, in milliseconds, for the package
            during the statistics interval. (Elapsed time is the wall-
            clock time between the start and the end of each LUW).
AVGELAPS    The average elapsed time during the interval, that is,
            TOTELAPS / SAMPLED.
MAXELAPS    The maximum elapsed time for the package during the
            interval.
LOCKWAIT    The total number of milliseconds the package has been in the
            lockwait state.
PCTLOCK     Percentage of time spent in lockwait, that is, (LOCKWAIT *
            100) / TOTWAIT.
IOWAIT      The total time spent in I/O or dataspace pagefault wait
            during the package, in milliseconds.
PCTIO       Percentage of time spent in I/O wait, that is, (IOWAIT *
            100) / TOTWAIT.
COMMWAIT    The total time spent in communications wait during the
            package, in milliseconds.
PCTCOMM     Percentage of time spent in communications wait, that is,
            (COMMWAIT * 100) / TOTWAIT.
CHKPWAIT    The total time spent in checkpoint wait during the package,
            in milliseconds.
PCTCHKP     Percentage of time spent in checkpoint wait, that is,
            (CHKPWAIT * 100) / TOTWAIT.
TOTWAIT     LOCKWAIT + IOWAIT + COMMWAIT + TOTWAIT in milliseconds.
PCTCPU      Percentage of CPU time used during the package, that is,
            (CPUTIME * 100) / TOTELAPS.
TOTROWS     The total number of table rows processed by the package.
AVGROWS     The average number of table rows processed by the package,
            that is, TOTROWS / SAMPLED.
RDSCALL     The number of calls to the Relational Subsystem during the
            package.
DBSSCALL    The number of calls to the Database Subsystem.
DISPCALL    The number of DB2/VM dispatcher calls.
BUFLOOKS    The number of buffer lookups performed by the package.
WAITLOCK    The number of times a lock request resulted in wait.
ESCALATE    The number of times a lock request resulted in lock
            escalation.
PAGREAD     The number of page reads during the package, that is, the
            number of times a request for data could not be satisfied
            from the buffer pool. The statistic includes reads from the
```

```
                    dataspaces if applicable.
    PAGWRITE        The number of page writes during the package. The statistic
                    includes writes to the dataspaces if applicable.
    LOGREAD         The number of reads to the DB2/VM log during the package.
    LOGWRITE        The number of writes to the DB2/VM log during the package.
    DIRBUFL         The number of directory buffer lookups performed.
    DIRREAD         The number of directory page reads during the package. The
                    statistic includes reads from the directory dataspace if
                    applicable.
    DIRWRITE        The number of directory page writes during the package. The
                    statistic includes writes to the directory dataspace if
                    applicable.
    TOTIO           The sum of PAGREAD, PAGWRITE, DIRREAD, DIRWRITE, LOGREAD and
                    LOGWRITE.
    DSFAULTS        The number of dataspace pagefaults during the package, if
                    DB2/VM dataspaces have been installed.
    BLOCKIO         The  number  of  VM  IUCV  BlockIO  requests  issued  by  the
                    database manager to CP during the package.
```

Notes:

If a package has been executed by multiple users, statistics are passed per userid. To obtain the
total consumption of the package, all package-user statistics must be totalled.
When all data accumulated during the last statistics period have been passed, an "end" call is
made to SQLMUAPP, by transmitting the fictitious packagename *END.

19.4.12   SQLMUAPR: Dynamic short-on-storage detection

The EXEC is called by the Statement Monitor (or by the auxiliary statement monitor), when the number of pages written to a given pool since the last checkpoint drops below (2 * current number of free pages in the pool).

The following arguments are passed to the EXEC:

- database name
- number of the pool where a possible short on storage condition exists
- an invocation count indicating the number of times the EXEC has been called in the current database session

19.4.13   SQLMUAPX: Agent restricting

The EXEC is called from the SQLCMDM machine (or the auxiliary SQLCMDM) when the Governor function has decided to force an agent because it has reached a restricting condition.

The following entry arguments are passed to the EXEC:

- name of the restricting condition reached with the same value as the keyword that enabled the condition in the RESTRICT file (e.g. MAX_IDLETIME, MAX_LOCKHOLD etc.)
- database name
- VM userid
- DB2 userid
- user location (terminal name)
- package name
- package section

The EXEC should return with one of the following values:

0   Restricting should be bypassed and the user allowed to continue execution.
1   Restricting should proceed to force the user.
2   When a MAX_LOCKTIME restriction has been presented, returncode 2 requests to force the lock holder and to allow the waiter(s) to continue. When a MAX_LOCKHOLD restriction has been presented, returncode 2 requests to force the lock waiter(s) and to allow the holder to continue.

If SQLMUAPX is called for a **MAX_LOCKTIME** restriction, additional data are stored in the CMS stack for all agents involved in the lock. The first entry in the stack is for the locked agent, the second for the locking agent. If more than 2 agents are involved in the lock, the entire lockchain is passed. In such cases, more than 2 entries will be present in the stack. The agent state in all these entries will be LOCKWAIT, except for the lock holder.

Each entry in the stack contains following data (words):

- agent number
- agent state
- agent R/W state
- agent idletime in seconds
- VM-userid
- DB2-userid
- packagename
- locked DBspace number

The "agent state" field takes one of the following values:

Label	Meaning
RUNNING	agent is processing
READYRUN	agent is ready for processing
IOWAIT	agent is waiting for completion of an I/O operation or a dataspace pagefault
LOCKWAIT	agent is waiting on a locked resource
COMMWAIT	agent is waiting for next SQL statement from client
CHKPWAIT	agent is waiting for completion of checkpoint
PBUFWAIT	agent is waiting for a page buffer in the buffer pool
DBUFWAIT	agent is waiting for a directory buffer
DSPFWAIT	agent is waiting for completion of a dataspace page fault (VMDSS only)
SLDWAIT	agent is waiting for a Save List Definition block (VMDSS only)

The "agent R/W state" field is set to **Y** for an R/W agent. Otherwise the value **N** is passed. The field "agent idletime" is available only when the agent is in COMMWAIT state. Otherwise, idletime is zero.

Note: If SQLMUAPX does not process the CMS stack entries, the stack will be cleared by the Governor on return from SQLMUAPX.

19.4.14   Sample SQLMUAP3 Process

```
/*   User attached process for active agents */

'SQLMFUAP 3'

/*
The procedure exits with RC 0 or RC 1, RC 1 indicating that
the LUW has been processed by the exit. The Statement
Monitor keeps this exit RC as the USERFLAG for the active
LUW and represents it to SQLMUAP3 at the next call.
*/

Exit_RC = 0

/*   (1)   Send message to DBADMIN for a long running Dbspace
           scan */

If (I_ACCESS=0)&(TOTIO>200)&(USERFLAG=0) then do
   'SQLMFCTX'
   Opcode = word(CMND_TXT,1)
   If (opcode='SELECT')|(opcode='DELETE')|(opcode='UPDATE')
     then do
       'CP MSG DBADMIN Dbspace scan. Database' DATABASE,
         'User' SQL_NAME,
         'Package' PACKAGE 'Section' SECTION,
         'Rows processed' NROWS
       'CP MSG DBADMIN SQL statement text',left(CMND_TXT,72)
       Exit_RC = 1
   End
End

/* (3)  Show users locked during 30 seconds or more */

If (LOCKTIME >= 30) & (USERFLAG = 0) then do
   'CP MSG DBADMIN Lockwait status. Database' DATABASE,
      'User' SQL_NAME
   'CP SET SECUSER DBADMIN'
   'EXEC SQLMFCMD' DATABASE 'SHOW LOCK GRAPH AGENT' AGENT
   'CP SET SECUSER RESET'
   Exit_RC = 1
End
Exit exit_RC
```

19.4.15 Sample SQLMUAPX Process

```
/*   User attached process for restricting agents */

arg restrict database VMid DB2id loc package section

/* assume agent should be forced */
exit_RC = 1

/* package ABC executed at terminal XYZ is allowed
   to violate the MAX_RESPONSE restriction */

if (restrict='MAX_RESPONSE')&(package='ABC')&(loc='XYZ')
   then exit_RC = 0

Exit exit_RC
```

20Application Program Interfaces

20.1Issuing an operator command from an application

Application programs may call the SQLMOPC module to execute a DB2/VM operator command in a database monitored by SQL/MF. A DB2/VM agent structure is not required for command execution. However, the executing VMid must be able to establish an IUCV connection with the target database server.

The SQLMOPC module is called with two parameters

- the 8-character name of the database server that must execute the command
- the 80-character DB2/VM command to execute

On return from SQLMOPC, a pointer is returned to the command reply, which is maintained within the SQLMOPC module as an array of 512 80-character output lines. The array is set to all blanks before issuing the command. After command completion, the array contains the command output, as it would have appeared on the console of the database server. In most cases, the last line of the reply starts with the string "ARI".

The SQLMOPC module has been designed mainly for use by assembler language programs. However, other languages may be able to satisfy the entry requirements described. The module is part of the SQL/MF distribution material and is stored on the SQL/MF product disk with the name SQLMOPC TEXT. This component should be linked with the application by declaring SQLMOPC as an external name.

When calling SQLMOPC, following general registers should be setup:

register 1 :
address of the parameterlist, which contains 2 pointers, namely (a) the address of an 8-byte area containing the databasename (b) the address of an 80-byte area containing the command to execute

register 13 :
address of a 72-byte register savearea

register 14 :
return address to the application

register 15 :
entry address of SQLMOPC

On return from SQLMOPC, general register 15 points to the command reply area.

Example:
```
        EXTRN   SQLMOPC
        CALL SQLMOPC,(DATABASE,Command)
*  REGISTER 15 NOW POINTS TO OUTPUT OF SHOW ACTIVE

DATABASE      DC      CL8'TESTDBN'
COMMAND       DC      CL80'SHOW ACTIVE'
```

20.2 Obtaining active agent information from an application

By calling the module SQLMFSA, an application program can get information about all user agents running in a designated DB2/VM database server.

The API is designed as a client-server application. The client program SQLMFSA performs an IUCV transaction with the SQLMFSAS server program, executing in the database. The server obtains the agent related information from the DB2/VM control blocks and passes them in the reply to the client's IUCV message.

20.2.1   Starting SQLMFSAS

In the DB2/VM startup stream, code the command EXEC SQLMFSAS before the EXEC SQLCSI command. This starts the Show Active server. The DB2/VM production disk must be accessed before starting SQLMFSAS, because the SQLLEVEL exec is called during SQLMFSAS initialization.

Please note that the server executes independently from the SQLCSI program. It responds to SQLMFSA requests, even when SQLCSI is inactive (because the MONPERIOD has expired for example).

By providing the authorization file SQLMFSAS USERS, access to the server can be restricted to a list of named users. If no authorization file is found on the CMS search chain by SQLMFSAS, all requests from users with an IUCV connection to the database will be accepted. Otherwise, the authorization file should specify the VM usernames allowed to access the service. Users not appearing in the file will be denied access and receive the "not authorized" returncode on their SQLMFSA call. Each user must be named in a separate file record and the username must be coded, starting in column 1 of the record.

20.2.2   Invoking SQLMFSA

SQLMFSA is linked with user applications to obtain show active services. The component returns DB2 information for all agents currently in LUW into a reply array provided by the application. The client needs IUCV authority to connect to the virtual machine running the SQLMFSAS program.

An application calls the SQLMFSA module with 2 parameters:

- the address of the 8-byte VMID of the DB2 server to be accessed
- the address of the show active reply array in the user program

The applications reply array should provide for 256 agents. Each array entry is 40 bytes long and contains the following information on return from SQLMFSA:

Item returned	Format
Agent Number	Smallint
Agent Status	Smallint
LUW id	Integer
Agent's VM-id	Char(8)
Agent's DB2-id	Char(8)
Creator of last package loaded by agent	Char(8)
Name of last package loaded by agent	Char(8)

Notes:

- System agents are not shown.
- The agent number is shown in internal DB2 format, that is, the first general purpose agent has agent number 4 (or 5 if TCP/IP support has been enabled).
- The reply array is delimited by agent number = 0.
- The agent status field takes the following values:

Value	Meaning
0	Agent is running or ready to run
2	Agent is in communication wait
3	Agent is in lock wait
4	Agent is in checkpoint wait
5	Agent is waiting on a page buffer
6	Agent is waiting on a directory buffer
7	Agent is waiting on dataspace pagefault completion
8	Agent is SLD-bound
9	Agent is waiting for I/O completion

SQLMFSA provides the following return codes (in register 15 if the call is from an assembler program)

Value	Meaning
0	Access successful and reply area filled in
4	SQLMFSA access not allowed in SQLMFSAS USERS file
8	ARICMOD not loaded
12	DS2MODE control block not found
16	DS2CVT control block not found
20	Timeout connecting to SQLMFSAS
24	HNDIUCV error
1xx	IUCV SEND errorcode
1xxx	IUCV CONNECT errorcode

Returncodes 8, 12 and 16 indicate that DB2/VM is no longer active in the designated VMID.
Returncode 24 should not occur and indicates an SQLMFSA error.
Returncodes 20 and 1xx should not occur and indicate an IUCV error in the SQLMFSA support modules.
Returncodes 1xxx may occur as follows:

- 1011 the requested DB2 server is not logged on
- 1012 the requested DB2 server is logged on but does not run SQLMFSAS
- 1013 MAXCONN exceeded for the client machine
- 1014 MAXCONN exceeded for the SQLMFSAS server
- 1015 client has no IUCV authority to connect to the database server

When calling SQLMFSA from an assembler application, following general registers should be setup:

register 1
address of the parameterlist, which contains pointers for the databasename and the reply area

register 13
address of a 72-byte register savearea

register 14
return address to the application

register 15
entry address of SQLMFSA (and returncode after call)

If SQLMFSA is called from a PL/1 program, SQLMFSA should be declared with OPTIONS(ASSEMBLER RETCODE).

Example

```
DCL    SQLMFSA ENTRY OPTIONS(ASSEMBLER RETCODE);
DCL    DATABASE CHAR(8) INIT('DBNAME');
DCL    1  REPLY(256),
          2  AGENT_NR BIN FIXED(15),
          2  ETC ...

CALL SQLMFSA,(DATABASE,REPLY);
IF PLIRETV() = 0 THEN DO;
   DO I = 1 TO 256 WHILE AGENT_NR(I) > 0
      /* PROCESS REPLY(I) */
   END;
END;
```

20.3 Formatting SQL statement text from an EXEC

When writing an UAP, the SQLCSFMT module may be used to format an SQL statement text. The function splits the statement text into multiple lines so that each major SQL clause causes a "line break".

The function must be called from a REXX exec, with the unformatted statement text as the argument. SQLCSFMT returns the formatted statement in the console stack.

Invocation:

```
call SQLCSFMT <statement_text>
do while queued() > 0;
    pull line;
    process line;
    end;
```

20.4 Obtaining active agent information from an EXEC

The SQLMFSAM module can be executed from the CMS prompt or from an EXEC. It uses the SQLMFSA link module to provide active agent information on the console or in the program stack.

SQLMFSAM is invoked as follows:

SQLMFSAM databaseserver [STACK]

where:

-    databaseserver is the VM userid of the DB2 server to inspect
-    the STACK option requests that output should be returned in the CMS stack
-    if the STACK option is omitted, output is shown on the console

The STACK option puts a line on the stack for each active agent. Each line contains the following data in character format, separated by a blank:

-    Agent number
-    Agent status
-    LUWid
-    Agent's VM-id
-    Agent's DB2-id
-    Package creator
-    Package NAME

If STACK is omitted, the same data are written to the console, preceded with a field label.

20.5 Calling SQLMUAP3 from an application

A VM EXEC may invoke SQLMUAP3 (Active Agent Status) to obtain monitor data for its own active LUW. To do so, the EXEC should proceed as follows:

Invoke the function SQLMUAPI. This prepares the application for the UAP call. The function exits with a non-zero returncode if no LUW is in progress for the requesting VM userid in the currently connected database.

Issue the command SQLMFUAP 3. This function stores all monitor data for the running agent into REXX variables, which can be processed by the EXEC and returned to the application program invoking the EXEC. For a list of the REXX variables created by SQLMFUAP 3, refer to SQLMUAP3 Active Agent Status below on page 291.

Example

Using a CMS CSL call, a PL/1 program invokes the following EXEC in order to obtain its current LUW identification.

```
LU=0                    /* Assume no LUW in progress */
'SQLMUAPI'              /* Initialize UAP */
IF RC = 0 THEN DO  /* If successful */
   'SQLMFUAP 3'        /*  build  REXX  variables  for running
                       agent */
   LU=LUWID             /* Return LUW-id */
END
RETURN  LU
```

21   Operational Notes

21.1   SQLCSI

SQLCSI is invoked in the DB2/VM startup procedure and terminates itself when a DB2/VM shutdown request is detected. Restarting SQLCSI is done by restarting the database server. The size of the monitor program is about 156K. SQLCSI is loaded below 16 Mb. Dynamic storage required during monitoring, is always allocated above the 16 Mb line.

To avoid interference with dynamic CMS storage requests issued by DB2/VM during operation, SQLCSI obtains a 4 Megabyte storage block from CMS during initialization and allocates its fixed-length storage requests from that block. The internal storage pool and all other dynamic storage is always acquired above the 16 Mb line. If your DB2/VM system is already storage constrained, you should allocate an additional 4 megabytes of virtual storage to the DB2/VM server.

The additional CPU consumption in the database server associated with monitoring depends on the application statement mix: in the average, it will range from 5 to 15 percent. The monitoring overhead will be higher for packages executing using the DRDA protocol. The monitor runs more efficiently when the database server has the VM privilege class C or E.

SQLCSI uses DB2/VM RDS tracing (EXEC 1) for the purpose of monitoring. Several actions are taken to limit the performance impact of tracing.

The SQLCSI abend exit is active while the monitor is running, to prevent an abend in the DB2/VM server. Should an abend condition occur within the monitor code, the abend exit will display and dump abend information and perform a monitor reload.

Since SQLCSI runs as an extension of the DB2/VM database server, VM favouring options granted to the database server also apply to the monitor.

21.2 SQLCS

SQLCS executes in the SQLCMDM machine. Although SQLCS operations are less critical than those in SQLCSI, the program is more vulnerable, since it processes DB2/VM tables and is open to external service requests. When an abend occurs in SQLCS, an abend dump is taken and the program performs an IPL CMS to restart itself.

SQLCS receives messages (via IUCV or the communications dataspace):

-   to process a statement benchmark or log request from SQLCSI
-   to process a statement or program statistics request from SQLCSI at the end of each program (SQLCSI queues the statement and program statistics and transmits them to SQLCS at regular time intervals and at shutdown of DB2/VM)
-   to process an object identification request from SQLCSU

If intensive logging or benchmarking occurs, it may be necessary to provide some VM favorings (such as QUICKDSP) to SQLCS.

Since SQLCS processes requests from all monitored databases, its virtual machine should remain active at all times. SQLCS issues COMMIT WORK RELEASE after processing, in order to release its connection to the serviced database.

SQLCS performs a daily program restart at midnight (or at the time specified on the EXEC SQLCS command). Restart implies a CMS IPL, and the PROFILE exec is executed at this time. The installation may include additional functions in the profile, to be executed during daily restart, such as printing the monitor tables, reorganizing them and so on.

21.3 SQLCSMDS full condition

SQLCSI stores its monitoring results in the SQLCSMDS shared segment. Shared storage management routines are used to allocate and deallocate monitor blocks within the segment. The first time the shared storage manager cannot satisfy a "get block" request, it issues a message on the console. Monitoring for new agents is disabled until the short on storage condition is cleared or until a monitored database server is restarted.

The MONITOR SHOW command can be used to display the number of times the full condition has occurred.

21.4 Shutdown considerations

The SQLCSI component that runs in the DB2 server initiates its own shutdown at DB2 shutdown.

To orderly terminate and logoff the system monitor, issue the command:
SQLMFX <Vmid> END

To orderly terminate and logoff the statement monitor, issue the command:
SSPIUCV <VMid> SQLCSS STOP

To orderly terminate and logoff the recorder writer, issue the command:
SSPIUCV <VMid> SQLCSCRW STOP

To orderly terminate and logoff the Autoprep server, issue the command:
SSPIUCV <VMid> SQLCSAPS STOP

Where <VMid> is the name of the virtual machine that runs the component.

Orderly termination means that the designated component will process all entries remaining in its request queues, before issuing a CP LOGOFF.

21.5 DRDA considerations

SQL/MF provides all its monitoring facilities for packages executing within the DB2/VM server using the DRDA protocol.

Due to the differences between the DB2/VM-only mail protocol and the DRRM data streams used by DRDA, the following should be noted:

- since a DRRM CONNECT does not provide a package name, the connect statement will not appear in the monitor output

- since a DRRM commit or rollback does not provide a package section number, the fictitious package number 9999 will be assigned by the monitor

21.6 Using secondary consoles

It is suggested that the VM secondary console facility be activated for the monitor components running in disconnected mode, that is, for the machines executing the SQLCS and the SQLMFM programs.

Note that the MSGROUTE statement in the SQLCS CONFIG dataset allows routing of all severe error messages, issued by an SQL/MF component, to a single VM userid. The MSGROUTE facility is explained on page 67.

To avoid flooding the secondary console, most components of the monitor facility contain logic to avoid repetitive display of the same message during the same monitoring session. If the log Dbspace is full for example, the message SQLCS005 SQLCODE -701 INSERTING INTO SQLCS_LOG would be displayed at every access to the log table. The SQLCS component ensures that a message related to a given SQLCODE is displayed only once per session.

21.7 Starting a DB2/VM trace

SQL/MF uses the DB2/VM tracing facility for its own purposes. Therefore, when you want to start a DB2/VM trace while the monitor is active, you must observe the following procedure.

Ensure that no users are in LUW.

To define the trace output device or filename, issue the following command at the CMS prompt:

SQLMFCMD <databasename> CMS FILEDEF .....

On the console of the database server, issue a DB2/VM **TRACE OFF** command. This closes the dummy trace started by SQL/MF. It also suspends monitoring while the DB2/VM trace is in progress. While monitoring is suspended, you can no longer use SQL/MF services (such as the SQLMFCMD interface).

Issue your own **TRACE ON cuu** command and proceed with tracing.

When tracing is finished, issue the **TRACE OFF NOCLOSE** command. This resumes SQL/MF Monitoring automatically.

21.8 SQL/MF support at a DB2/VM System Error

Should a DB2/VM system error occur, you may find the information recorded by SQL/MF in the SQLCSMDS shared segment useful to locate the origin of the system error.

SQL/MF permanently records the status of the active agents in the shared segment. When a DB2/VM system error occurs, the shared segment contains the status of the agents at that time. To display the shared segment, invoke the SQLCSU program before restarting the database. This will show which agents were active at system breakdown. The user shown by SQLCSU with status Run is probably the agent active at (and possibly responsible for) the error. For this agent, the shared segment shows the username, the packagename, the section number, the type of the statement and the statement text (provided the SQL_STMNTS table is not in the failing database).

Alternatively, you can use the SQLCSFMS utility to copy the status of all active agents to a CMS file, before restarting the database. The SQLCSFMS utility is described on page 333 of this manual.

21.9 DB2/VM Release Migrations

Installing a new DB2/VM release, supported by SQL/MF, does not require any special actions, as far as SQL/MF is concerned. The monitor code contains support for all DB2/VM releases from 3.3 to 7.2. Using the SQLLEVEL information, the monitor automatically activates the appropriate release code at startup of the database server.

22 Utility Programs

22.1 SQLMFRTM: APPC Response time monitor

Purpose

Show various statistics about the number and duration of APPC requests. The main purpose is to monitor the APPC requests that are directed to a DB2 server. However, other APPC requests will be monitored using separate counters, provided these requests generate an external interrupt x'4000' at request completion. (This is not the case with SFS-related APPC.)

22.1.1 Initiating SQLMFRTM

On the CMS prompt or from a user exec issue:

[EXEC] SQLMFRTM ON [SCALE n m]

When SCALE is specified, SQLMFRTM will distribute the intercepted waittimes on the scale n*1, n*2, n*3, ..., m

Waittimes above m will appear as last entry on the scale.

The values n and m should be expressed in milliseconds. An invocation error will result if n is zero or if m is not a multiple of n.

22.1.2 Terminating SQLMFRTM

On the CMS prompt or from a user exec issue:

[EXEC] SQLMFRTM OFF

Following report will be displayed on the console:

```
    Number of non-DB2 APPC requests: x
        Total non-DB2 APPC waittime: hh:mm:ss.xxxxxx
      Largest non-DB2 APPC waittime: hh:mm:ss.xxxxxx
     Smallest non-DB2 APPC waittime: hh:mm:ss.xxxxxx

        Number of DB2 APPC requests: x
            Total DB2 APPC waittime: hh:mm:ss.xxxxxx
          Largest DB2 APPC waittime: hh:mm:ss.xxxxxx
         Smallest DB2 APPC waittime: hh:mm:ss.xxxxxx

    (where xxxxxx is the microsecond part of the waittimes)
```

When SQLMFRTM has been initiated with the SCALE option, following report will be added:

```
Scale      N_Waits      Total_waittime

 < n       x            hh:mm:ss.xxxxxx
 < n*2     x            hh:mm:ss.xxxxxx
 < n*3     x            hh:mm:ss.xxxxxx
  .
  .
  .
>= m       x            hh:mm:ss.xxxxxx
```

22.2 SQLCSXPL: Analyzing recorded SQL statements

If the SQL/Command Analysis program product has been installed on your system, you can use the SQLCSXPL program to analyze prepped DB2/VM packages and statements, based on their actual execution characteristics, as recorded by SQL/MF in the SQLCS_SQL_STMNTS table.

For example, you can use the SQLCSXPL utility to automatically analyze:

all statements doing more than a defined number of I/O requests
all statements fetching a defined number of table rows
all statements executing without an index etc...

Invocation Syntax

SQLCSXPL <databasename> [WHERE] <search-clause>

where:

<databasename> Designates the database containing the DB2/VM packages to be inspected.

WHERE An optional keyword provided for readability.

<search-clause> Defines one or more SQL WHERE predicate clauses to be applied by SQLCSXPL when searching the SQLCS_SQL_STMNTS table for statements to be analyzed. The search clause may contain references to one or more of the SQLCS_SQL_STMNTS table columns. For a description of these columns, please refer to page 257.

SQLCSXPL stores the selected statements into a CMS file named SQLCSXPL ACCMOD. This file is then submitted to SQL/CA for analysis. The ACCMOD file may contain statements from different packages. In this case the name of the originating package can be found in the header lines of the SQL/Command Analysis report.

Examples

SQLCSXPL DBN1 WHERE BUFLOOK > 10000
Analyses all statements doing more than 10000 buffer lookups.

SQLCSXPL DBN1 WHERE INAME = ' ' AND NROWS > 100
Analyses all statements processing more than 100 table rows without using an index.

SQLCSXPL DBN1 WHERE (DBSSCALL / RDSCALL) > 10
Analyses "complex" statements, that is, statements (RDSCALLs) needing more than 10 calls to the database subsystem to execute.

22.3 SQLCSXSA: Analyzing logged SQL statements

If the SQL/Command Analysis program product has been installed on your system, you can use the SQLCSXSA program to analyze SQL statements recorded by SQL/MF in the exception table SQLCS_LOG.

Invocation Syntax

SQLCSXSA <search-clause>

where:

<search-clause>   Defines one or more optional predicate clauses to be applied when searching the SQLCS_LOG table for statements to be analyzed. If no search clause is specified, SQLCSXSA will select all exception and benchmark rows in the table. The search clause may contain references to one or more of the SQLCS_LOG table columns. For a description of these columns, please refer to page 259.

SQLCSXSA stores the extracted statements into a CMS file named SQLCSXSA FILE. This file is then XEDITed for verification. When quitting XEDIT, the user is prompted   submitted to SQL/CA for analysis.

Examples

SQLCSXSA BUFLOOK > 10000
Analyses all statements doing more than 10000 buffer lookups.

SQLCSXPL LOGTYPE = 'B' AND TOTIO > 1000
Analyses all benchmarked statements performing more than 1000 I/O requests.

22.4 SQLCSUNL: Unloading Session RunStats

The session run statistics facility records, for each user and program executed:

- the total number of samplings during the current DB2/VM session
- the total CPUtime usage in milliseconds
- the total number of buffer lookups
- the total number of I/O requests and dataspace-to-bufferpool transfers

These statistics can be inspected interactively, using SQLCSU. The SQLCSUNL utility allows to off-load the section statistics into a CMS file. The record layout has been designed in such a way, that the file can be easily loaded into a DB2/VM table, using SQLDBSU DATALOAD.

22.4.1   Invocation

The program is invoked using the following command:

SQLCSUNL database-servername

The program creates a file on the requester's A-disk named <servername> RS<yymmdd>.

The output file is ordered by username, program name and section number.

If a runstats file already exists for the named databaseserver and date, it will be replaced.

22.4.2   Unload file record layout

Contents	Format	Position	Length
executing_SQL_userid	CHAR	1	8
filler	CHAR	9	1
package_creator	CHAR	10	8
filler	CHAR	18	1
package_name	CHAR	19	8
filler	CHAR	27	1
package_section_number	NUMCHAR	28	4
filler	CHAR	32	1
section_sample_count	NUMCHAR	33	11
filler	CHAR	44	1
program_sample_count	NUMCHAR	45	11
filler	CHAR	56	1
CPUtime_milliseconds	NUMCHAR	57	11
filler	CHAR	68	1
number_of_buffer_lookups	NUMCHAR	69	11
filler	CHAR	80	1
number_of_I/O_requests	NUMCHAR	81	11

Note

NUMCHAR fields are stored left-aligned, without leading zeroes or blanks.

22.5 SQLCSRST: Rebuilding the SQL_STMNTS table

The SQLCS_SQL_STMNTS table contains the text and the resource consumption for all SQL statements in a database. SQL/MF automatically maintains the text of the statement, by sensing program prep events or by unloading the program from the database when needed.

In some cases however, it may be necessary to explicitly rebuild the SQLCS_SQL_STMNTS table, for example to remove the statistics for programs that are no longer in use.

The SQLCSRST utility allows to rebuild the SQL_STMNTS rows for a designated database. The utility will do this by

-    deleting all the SQL_STMNTS rows for the database
-    reload the statement text for all programs from the database

After refreshment, all statistical counters are reset to zero.
After the table refresh, the SQLCS_SQL_STMNTS Dbspace should be reorganized to reclaim the empty pages resulting from DELETE.

To invoke the utility, type SQLCSRST at the CMS prompt.
On the next panel, specify:

-    the name of the database to be refreshed
-    the password of SQLDBA in this database
-    the password of SQLDBA in the monitor database

Note   If the target database to be refreshed has many packages, SQLCSRST can take a long time to execute. Execution should then be scheduled during off-peak periods.

22.6 SQLCSCLR: Dropping the Statement Monitor tables

The SQLCSCLR EXEC is used:

to drop the SQLCS_LOG and SQLCS_SQL_STMNTS DBspaces in the SQL/MF log database
to drop the SYSCAT_TABID_I1 index created by SQL/MF in each monitored database

Type SQLCSCLR at the CMS prompt.

- When prompted, enter the name of the SQL/MF log database. (The log database is the database where the SQLCS_LOG and SQLCS_SQL_STMNTS DBspaces have been created).
- When prompted, enter the password of SQLDBA in the log database.
- When prompted, enter the name of the first or next monitored database.
- When prompted, enter the password of SQLDBA the monitored database.
- When all databases have been processed, enter a blank databasename. This will terminate SQLCSCLR.

22.7 SQLCSFMS: Formatting the Monitor Shared Segment

Purpose

Take a copy of the shared segment entries for a designated database to a CMS file, for future reference.

The utility can be used to save the status of all active agents in the database, for example after a DB2/VM system error.

The utility creates a CMS file on the requester's A-disk with a filename equal to the databasename and the first 4 bytes of the hardware clock as the filetype. (These bytes contain the current date and time, in 1-second precision).

Invocation

SQLCSFMS databasename

22.8 SQLCSDR0: Migration Utility

Purpose

From SQL/MF version 5.1 on, the SQLCS_SQL_STMNTS table no longer has descriptor rows for the statement's hostvariables. This results in a significant saving of page space.

Users migrating from a previous SQL/MF version, should execute the SQLCSDR0 utility to remove the hostvariable rows for a named database.

The utility commits its work, each time 100 hostvariable rows have been deleted.

To reclaim the space freed by the delete process, you should reorganize the SQLCS_SQL_STMNTS table.

Invocation

SQLCSDR0 databasename

22.9 SQLCSBRL : Extract from a backup Lockwait Recorder

Purpose

The utility processes a Lockwait Recorder file or fileset that has been restored from a backup device to a virtual disk of the invoking virtual machine.

The virtual device address is passed as the only argument when SQLCSBRL is invoked. The utility then displays the normal lockwait recorder extract panel.

Invocation

SQLCSBRL disk_device_address

Note:

The recorder file backup must be done by user procedures.

22.10   SQLCSBRX : Extract from a backup Statement Recorder

Purpose

The utility processes a Statement Recorder file or fileset that has been restored from a backup device to a virtual disk of the invoking virtual machine.

The virtual device address is passed as the only argument when SQLCSBRX is invoked. The utility then displays the normal statement recorder extract panel.

Invocation

```
SQLCSBRX disk_device_address
```

Note

The recorder file backup must be done by user procedures.

22.11   SQLMFQRY: Query Monitor Status

Purpose

The utility returns the current status of the monitor program SQLCSI in a designated DB2 database server. The status is passed as the module returncode, as follows:

- 0   monitor is not active or has abended
- 1   monitor is running
- 2   monitor has been suspended by the MONITOR SUSPEND command
- 3   monitor is in bypass, due to the PERIOD statement in SQLCS CONFIG

Invocation

SQLMFQRY database_server_name

Example

```
/* REXX program to test SQLCSI status */

arg DB2_server
'SQLMFQRY' DB2_server
if rc = 1 then   /* monitor is running normally */
else             /* monitoring is inactive */
```

22.12   SQLMFGPR: Generic Package Rebind

Purpose

Perform a rebind for multiple packages in one SQLDBSU run.

Invocation

[EXEC] SQLMFGPR database packagename

where **database** is the target database and **packagename** generically designates the package(s) to be rebound

Example

SQLMFGRP SQLPRD $%

Will rebind all autoprep packages in database SQLPRD.

Note
SQLMFGRP will connect the database under the user's default SQL-id.

22.13   SQLMFTCP: Terminate outstanding TCP/IP connections

Purpose

Terminates outstanding TCP/IP connections on a named TCP/IP client and portnumber.

Invocation

[EXEC] SQLMFTCP KILL clientname portnumber

Example

SQLMFTCP KILL SQLDBA 8000

Will terminate all outstanding connections on port 8000 of client SQLDBA.

Note

The SQLMFTCP exec is called by the MONITOR ENDLINKS command.

22.14   SQLMFENV: Unload/Reload SQL/MF environment

Purpose

- Save or restore the autoprep packages and the autoprep table.
- Restore the SQL/MF packages using the current SQL/MF product disk.

When a database is copied over another (using DDR for instance), the autoprep data can be saved in the copy target prior to the copy and restored after the copy.

Invocation

[EXEC] SQLMFENV type database SQLDBA_password filemode [Autoprep_servername]

- **TYPE** should be specified as:
    - UNLOAD to unload all autoprep packages and the autoprep table for "database" on the minidisk designated by "filemode"
    - RELOAD to reload the autoprep packages, the autoprep table and the SQL/MF packages for "database" from the minidisk designated by "filemode"
    - RELOADAP to reload the autoprep packages and the autoprep table only
- **DATABASE** designates the databasename for which the requested operation should be performed
- **SQLDBA_PASSWORD** specifies the password of user SQLDBA in that database; the password is required only when reloading the SQL/MF packages (type=RELOAD). The RELOADAP function does not require the password. However, since the EXEC arguments are positional, a password value must be specified. That value can be a dummy, if desired.
- **FILEMODE** designates the CMS filemode for the unload output or the reload input files
- **AUTOPREP_SERVERNAME** designates the VM-id of the SQL/MF Autoprep server, which is also the creator of the autoprep packages; the SQL/MF default value **SQLCSAPS** is assumed when the argument is omitted

Examples

EXEC SQLMFENV unload DBTEST **password** W will unload all autoprep packages and the autoprep table for the database DBTEST on the minidisk accessed as W

EXEC SQLMFENV reloadap DBTEST **password** W will reload all autoprep packages and the autoprep table for the database DBTEST from the minidisk accessed as W

Notes

- When executing SQLMFENV, the autoprep facility **MUST** have been disabled previously via a **MONITOR AUTOPREP OFF** command. Since the autoprep status is kept across database sessions, a MONITOR AUTOPREP ON command is required after SQLMFENV to re-activate the facility.
- The CMS filename of the saved autoprep table is **AUTOPREP <databasename>**
- The CMS filename of a saved autoprep package is **$xxxxxxx <databasename>**
- The command **EXEC SQLMFSI SQL_INSTALL** will be used during the SQL/MF package restore function; this command reloads the SQL/MF packages from the $ACCMOD distribution files on the SQL/MF product disk. The product disk can be accessed read-only in any CMS filemode.
- DBA authority is required to execute the SQLMFENV exec.
- If a non-dummy SQLDBA password has been specified, the autoprep Dbspace will be dropped and re-acquired during SQLMFENV reload.

23 SQL/MF Component Description

23.1 Statement Monitor Components

23.1.1 SQLCSI

The SQLCSI program contains the SQL/MF statement monitoring logic. The program executes in the virtual machine of the DB2/VM database server and dynamically installs itself as an extension of the DB2/VM software. SQLCSI screens all executing SQL statements and maintains a description of their characteristics in the monitor shared segment "SQLCSMDS". SQLCSI maintains execution statistics for all packages and package statements executed. If requested in the SQLCS CONFIG configuration file, following services are provided:

- statement logging
- statement benchmarking
- statement notification
- statement recording

The logging, benchmarking and statistic functions are not executed within SQLCSI itself (because they imply an SQL INSERT to the Log tables). Instead, the logging request is passed to SQLCS (described in the next paragraph), by means of an IUCV transaction or a communications dataspace.

23.1.2 SQLCS

The SQLCS program runs in a dedicated virtual machine as a functional extension of SQLCSI. It provides the following functions:

- Write to the Monitor tables on behalf of an SQLCSI request to log or benchmark a given SQL statement, or to store the statement and package statistics.

- Perform the notification and user restricting functions.

- Call the User Attached Process, if defined.

- Perform the package unload function. For existing packages, package unload retrieves the statement text from the SQL_STMNTS table. For new packages or for packages reprepped while SQLCSI was inactive (due to the PERIOD statement), SQL Database Services will be invoked to obtain the statement text by means of a Package UNLOAD request. The unloaded package is then stored in the SQL_STMNTS table. Packages reprepped or reloaded while SQLCSI is active are intercepted by SQLCSI and directly stored into the SQL_STMNTS table. In such cases, no subsequent DBSU unload will be needed.

- Provide object names to the SQLCSU program.

23.1.3   SQLCSU

SQLCSU is a user interface program invoked from the CMS prompt. It displays the list of executing SQL statements and acts as a starting point for invoking following components of the user interface.

SQLCSLST
the SQL/MF table editor. It accesses the SQL/MF Statement Monitor Log and SQL_STMNTS tables, using a log report menu. It provides for consultation of the benchmark log, the exception log and the statement or program statistics.

SQLGRAPH
provides a graphical representation of DB2/VM activity within a designated database server by showing a bar graph for the DB2/VM counters: RDSCALL, BUFLOOK, LUW, TOTIO.

SQLPULSE
provides a graphical representation of DB2/VM activity within all monitored databases. It shows a bar graph for the DB2/VM counters: BUFLOOK, CPU, TOTIO. The program's detail function shows a bar graph of all counters for the designated statement.

SQLCSCRX
extracts selected data from the CMS recorder file when the recorder facility has been enabled.

23.1.4   SQLCSCRW

The Recorder Writer SQLCSCRW is an optional component and runs in its own virtual machine. It is active when the SQL/MF statement recording facility has been enabled. While SQLCSI writes the recorded data to the recorder dataspace, the recorder writer transfers these data from the dataspace to the CMS recorder file.

23.1.5   SQLCSAPS

The AutoPrep server SQLCSAPS is an optional component and runs in its own virtual machine. It should be activated when the AutoPrep facility has been enabled. The component performs DB2 prep for the dynamic statements sent by SQLCSI.

23.2 System Monitor Components

23.2.1   SQLMFM

The SQLMFM program executes in a dedicated virtual machine and periodically samples the global DB2/VM system status, by inspecting the global system data, collected by SQLCSI in the monitor shared segment. SQLMFM saves these data into CMS datasets. Every hour, and when an interactive inspect request is received, the CMS files are loaded into the system monitor tables.

23.2.2   SQLMF

SQLMF is a user interface program invoked from the SQLCSU program. It requests SQLMFM to insert its session datasets into the SQL system monitoring tables. These tables may then be inspected, using a menu oriented interface.

24 SQL/MF Messages

24.1 SQLCSI Messages

SQLCSI000 **Monitoring initiated for DB2/VM Version N Release M**

Informatory message showing the DB2/VM version and release number obtained from the SQLLEVEL EXEC.

SQLCSI001 **HNDIUCV RC is xx**

The server-mode HNDIUCV macro returns errorcode xx

Call for software support.

SQLCSI002 **IUCV SEND to <name> returns RC xx**

During a log request, the client-mode SEND to virtual machine <name> returns errorcode xx

Check that the correct LOG monitor name has been specified in the SQLCS CONFIG file and that the monitor is logged on. Verify that SQLCSI has the required IUCV privileges to establish connections with the virtual machine running SQLCS.

For a description of the IUCV returncodes, refer to page 368.

SQLCSI003 **IUCV SEND RC is xxx**

During a log request, the client-mode SEND CONNECT to virtual machine <name> returns errorcode xx

Check that the correct LOG monitor name has been specified in the SQLCS CONFIG file and that the monitor is logged on. Verify that SQLCSI has the required IUCV privileges to establish connections with the virtual machine running SQLCS.

For a description of the IUCV returncodes, refer to page 368.

SQLCSI004 **HNDIUCV RC is xx**

The client-mode HNDIUCV macro returns errorcode xx

Call for software support.

SQLCSI005 **"xxxx" Notification. User N, package N, section N**

The condition "xxxx" specified in the configuration NOTIFY statement has occurred for the named user and program.

SQLCSI010   **SQLCSI at xxxxxxxx**

Informatory message showing the load address of SQLCSI.

SQLCSI011   **SQLCSMDS at xxxxxxxx**

Informatory message showing the address of the shared monitor segment.

SQLCSI012   **Timeout during IUCV CONNECT to <log_monitor_name>**
SQLCSI013   **Timeout during IUCV SEND to <log_monitor_name>**

When forwarding a request to the log monitor using IUCV, a timeout has occurred, as the monitor is temporarily not responding. If more than 5 consecutive timeouts occur, message SQLCSI014 is issued and the IUCV path to the monitor machine is reset.

SQLCSI014   **IUCV path to <log_monitor_name> has been reset**

5 consecutive timeouts have occurred. The path with the log_monitor is reset.

Determine the error occurring in the log monitor. Restarting the SQLCS program running in this machine will resume IUCV communications.

SQLCSI015   **IUCV path to <log_monitor_name> has been initialized**

Informatory message indicating that the log monitor machine has established a communications path with the SQLCSI program.

SQLCSI016   **Force for agent n LUW nnn ignored. Current LUWID is nnn.**

When the Governor Facility sends a FORCE command to SQLCSI, it includes the LUWID of the current agent. SQLCSI has compared this LUWID with the current LUWID of the agent and found a mismatch. Therefore, the force request was ignored.

SQLCSI017   **Processing SQLCSI PROFILE**

This informatory message is issued during SQL/MF startup when the file SQLCSI PROFILE was found on any of the accessed minidisks. All statements contained in the PROFILE are now executed.

SQLCSI080   **Following statements read from SQLCS CONFIG file:**

The statements read from the SQLCS CONFIG file are printed on the console.

SQLCSI081   **MONITOR name is required.**

A LOG statement has been specified in the SQLCS CONFIG file, but no MONITOR statement did precede.

SQLCSI082   **ANALYZE statement invalid.**

The ANALYZE statement specifications in the SQLCS CONFIG file are in error.

SQLCSI083 **Invalid CONFIG keyword xxxxx**

An unrecognized statement keyword has been found in the SQLCS CONFIG file.

SQLCSI084 **Monitor name missing.**

A MONITOR statement without monitor name has been found in the SQLCS CONFIG file.

SQLCSI085 **Monitoring period incorrect.**

The PERIOD statement in the SQLCS CONFIG file specifies an incorrect time (hhmm) argument.

SQLCSI086 **Invalid LOG argument xxxx.**

The LOG statement in the SQLCS CONFIG file specifies the unrecognized argument xxxx.

SQLCSI087 **xxx statement invalid.**

The BENCHMARK statement in the SQLCS CONFIG file specifies the unrecognized argument xxxx.

SQLCSI088 **Invalid NOTIFY parameter xxx**

The NOTIFY statement in the SQLCS CONFIG file specifies the unrecognized parameter xxxx.

SQLCSI089 **NOTIFY NAME argument missing.**

The NOTIFY statement in the SQLCS CONFIG file omits the required operand NAME.

SQLCSI090 **Unable to get a free MDS block.**

The shared storage manager was unable to get a free MDS block, because the shared segment SQLCSMDS is full. The message may indicate that you should increase the size of the SQLCSMDS shared segment and redefine it.

This message is issued only once. To determine the number of times the MDS full condition occurred, issue a MONITOR SHOW command and look at the corresponding statistic.

SQLCSI091 **RC nn reading SQLCS CONFIG.**

Reading the SQLCS CONFIG file results in the CMS file system error nn. Verify that the record length of this file does not exceed 256 characters.

SQLCSI092 **RC nn loading "SQLCSMDS" shared segment.**
SQLCSI093 **Monitor facility has NOT been installed.**

When loading the shared monitor segment, the CMS SEGMENT macro returns errorcode nn. Verify that the segment has been installed correctly.

SQLCSI094   **INITIAL statement in error.**

The INITIAL statement in the SQLCS CONFIG file has been coded incorrectly.

SQLCSI095   **MONITOR command in error.**

The MONITOR command you supplied in the HostCommand section of the SQLCSU program is incorrect.

SQLCSI096   **Monitoring has been suspended.**

The MONITOR SUSPEND command you supplied in the HostCommand section of the SQLCSU has taken effect.

SQLCSI097   **Monitoring has been resumed.**

The MONITOR RESUME command you supplied in the HostCommand section of the SQLCSU has taken effect.

SQLCSI098   **RC <n> issuing [UN]LOCK <type> <address-1> <address-2>**

Processing the LOCK statements in the SQLCS CONFIG file resulted in the returncode displayed. The returncode is the suffix of the CP message that would have been issued, if the [UN]LOCK command were issued from the console of the server.

Ensure that the database server machine has the CP privilege class A.

SQLCSI099   **Syntax errors processing SQLCS CONFIG**

Syntax errors were found in the configuration file, as indicated in a preceding error message.

SQLCSI100   **SQLLEVEL EXEC could not be executed.**

At startup of SQL/MF, the SQLLEVEL EXEC was not on the CMS search chain. Monitoring has not been initiated.

Ensure that the DB2/VM production minidisk can be accessed by the SQLCSI program.

SQLCSI101   **Verification of system counters fails.**
SQLCSI102   **Verification 2 of ARICDSP fails.**
SQLCSI103   **Verification 1 of ARICDSP fails.**

The 3 above messages indicate a discordance between the DB2/VM level expected by the monitor and the actual DB2/VM code loaded.

Verify that the correct SQLLEVEL EXEC is being used. Otherwise, call for software support.

SQLCSI104   **Short DB2 trace call path enabled.**

Informatory message indicating that the DB2/VM short trace path is active.

SQLCSI105   **<command_text> submitted by <VM_userid>**

Informatory message indicating that the designated VM user submitted the named DB2 operator command, using the SQL/MF interface.

SQLCSI106   **MONITOR command processing complete**

Informatory message indicating normal completion of a MONITOR command.

SQLCSI200   **RC n {creating | querying} recorder dataspace**

Where n is the returncode from the CP ADRSPACE CREATE macro, as follows:

8   Creating the new address space would cause the maximum number of address spaces allowed for your virtual machine to be exceeded. No new address space has been created.
12 Creating the new address space would cause the total size of all address spaces owned by your virtual machine to exceed the maximum permitted. No new address space has been created.
16 The specified address-space name contains invalid characters. No new address space has been created.
20 The specified size for the new address space is out of range. No new address space has been created.

This message is issued only when the Statement Recording facility has been requested in the SQLCS CONFIG dataset. The Recording facility is now disabled.

SQLCSI201   **RC n after ALSERV ADD**

Where n is the returncode from the CP ALSERV ADD macro, as follows:

4   There are no unused entries in the host access list that can be used to establish the new ALE.
8   The specified ASIT does not identify a currently-existing address space for which your virtual machine is authorized for the requested type of access.

This message is issued only when the Statement Recording facility has been requested in the SQLCS CONFIG dataset. The Recording facility is now disabled.

SQLCSI202   **RC n permitting recorder dataspace access**

Where n is the returncode from the CP ADRSPACE PERMIT macro, as follows:

4   The specified ASIT does not identify a currently existing address space that your virtual machine owns. No authorization has been granted.

28   The specified user ID or VCIT does not designate a currently logged-on or disconnected virtual machine. No authorization has been granted.

32   Your virtual machine is not authorized to use the PERMIT function of the ADRSPACE macro.

This message is issued only when the Statement Recording facility has been requested in the SQLCS CONFIG dataset.The Recording facility is now disabled, except for returncode 28, which indicates that the Recorder Writer is currently not logged-on. The access permit request will be re-issued automatically when recorder data are being stored.

SQLCSI203   **RC n after ALSERV REMOVE**

Where n is the returncode from the CP ALSERV ADD macro, as follows:

4   The ALET specified by the ALET operand does not designate an ALE in either the valid or revoked states. No ALE states have been changed.

12   The ALET specified by the ALET operand contains invalid bit settings. No ALE states have been changed.

This message is issued at DB2/VM shutdown only when the Statement Recording facility has been requested in the SQLCS CONFIG dataset.

SQLCSI204   **Recorder dataspace full. Recording temporarily suspended.**

This message is issued only when the Statement Recording facility has been requested in the SQLCS CONFIG dataset.

The message probably indicates that the Recorder Writer component is not active. When the Writer has been activated, processed recorder entries will create free space and recording will continue.

However, if the message is issued because the recorder in-transit queue is too large, you should increment the RECORDER DSPSIZE parameter in the SQLCS CONFIG file.

SQLCSI205   **Waiting for recorder dataspace depletion..**

DB2/VM shutdown has been requested and non-processed statement recording entries are found in the recorder dataspace. Shutdown waits until all entries have been processed, or until it is found that the writer is not processing, in which case message SQLCSI206 is given, after which shutdown proceeds.

SQLCSI206   **Recorder writer is not processing.**

Message SQLCSI205 has been given and it is found after 30 seconds that the recorder writer is active but not processing the recorded data. Shutdown proceeds.

SQLCSI207 **Recorder writer inactive. Recorder data have been lost**

During monitor shutdown, it is observed that recorder data are queued in the dataspace but that the statement writer is not active. Shutdown proceeds. The Recorder data have been lost.

SQLCSI211 **RECORDER FROM - TO argument missing.**
SQLCSI212 **RECORDER WRITER argument missing.**
SQLCSI213 **RECORDER INTERVAL argument missing.**
SQLCSI214 **RECORDER LEVEL argument missing.**
SQLCSI215 **RECORDER KEEP argument missing or invalid.**
SQLCSI216 **RECORDER DSPSIZE argument invalid.**
SQLCSI217 **Invalid RECORDER parameter nnn**

Messages SQLCSI201 - SQLCSI207 are issued only when the Statement Recording facility has been requested in the SQLCS CONFIG dataset. The messages indicate that a RECORDER statement argument is specified incorrectly or that a required argument has been omitted.

The statement recording facility is disabled.
Correct the SQLCS CONFIG statement in error.

SQLCSI218 **Statement Recording has been suspended.**

Message confirming the MONITOR CRSTOP statement.

SQLCSI219 **Statement Recording has been resumed.**

Message confirming the MONITOR CRSTART statement.

SQLCSI220 **Statement Recording now from hhmm to hhmm**

Message showing the actual recording period after the MONITOR CRFROM or CRTO commands.

SQLCSI221 **Recorder dataspace addressability error.**
SQLCSI222 **Snap dump has been taken and recording has been disabled.**

A software error has occurred during statement recording. A snap dump has been sent to the virtual printer.

Save the console output and the snap dump and forward them to software support.

SQLCSI300   **RC n {creating | querying} communications dataspace**

Where n is the returncode from the CP ADRSPACE CREATE macro, as follows:

8   Creating the new address space would cause the maximum number of address spaces allowed for your virtual machine to be exceeded. No new address space has been created.
12 Creating the new address space would cause the total size of all address spaces owned by your virtual machine to exceed the maximum permitted. No new address space has been created.
16 The specified address-space name contains invalid characters. No new address space has been created.
20 The specified size for the new address space is out of range. No new address space has been created.

This message is issued only when the COMQ_SIZE statement has been coded in the SQLCS CONFIG dataset, to request dataspace-based communications. After the above error, IUCV is used as the communications protocol.

SQLCSI301   **RC n after ALSERV ADD**

Where n is the returncode from the CP ALSERV ADD macro, as follows:

4   There are no unused entries in the host access list that can be used to establish the new ALE.
8   The specified ASIT does not identify a currently-existing address space for which your virtual machine is authorized for the requested type of access.

This message is issued only when the COMQ_SIZE statement has been coded in the SQLCS CONFIG dataset, to request dataspace-based communications. After the above error, IUCV is used as the communications protocol.

SQLCSI302   **RC n permitting communications dataspace access**

Where n is the returncode from the CP ADRSPACE PERMIT macro, as follows:

4   The specified ASIT does not identify a currently existing address space that your virtual machine owns. No authorization has been granted.
28 The specified user ID or VCIT does not designate a currently logged-on or disconnected virtual machine. No authorization has been granted.
32 Your virtual machine is not authorized to use the PERMIT function of the ADRSPACE macro.

This message is issued only when the COMQ_SIZE statement has been coded in the SQLCS CONFIG dataset, to request dataspace-based communications. After the above error, IUCV is used as the communications protocol, except when the returncode equals 28, which indicates that the Statement Monitor machine is not logged-on currently. In this case, the access permit request will be retried automatically.

SQLCSI303 **RC n after ALSERV REMOVE**

Where n is the returncode from the CP ALSERV ADD macro, as follows:

4   The ALET specified by the ALET operand does not designate an ALE in either the valid or revoked states. No ALE states have been changed.

12 The ALET specified by the ALET operand contains invalid bit settings. No ALE states have been changed.

This message is issued at DB2/VM shutdown only when dataspace-based communication has been requested in the SQLCS CONFIG dataset.

SQLCSI304 **Communications dataspace full. Communication temporarily suspended.**

This message is issued only when dataspace-based communication has been requested in the SQLCS CONFIG dataset.

The message may indicate that the SQLCMDM machine is not logged-on or that it is no longer polling the dataspace (because it is in a DB2/VM wait state for instance).

However, the message may also indicate that you should request a larger dataspace (the COMQ_SIZE parameter in the SQLCS CONFIG file).

SQLCSI305 **Waiting for communications dataspace depletion..**

DB2/VM shutdown has been requested and non-processed requests were found in the communications dataspace. Shutdown waits until all entries have been processed, or until it is found that SQLCMDM is not processing, in which case message SQLCSI306 is given, after which shutdown proceeds.

SQLCSI306 **Statement Monitor is not processing. Communications dataspace dropped.**

Message SQLCSI305 has been issued and it is found that SQLCMDM did not perform dataspace polling during the last 30 seconds. DB2/VM shutdown is resumed and the log requests or statistical data in the communications dataspace are lost.

SQLCSI990   **Program check during monitoring.**
SQLCSI992   **DUMP is in progress ...**

The above messages are issued when a program check occurs in the Statement Monitor. Save the messages and the dump files and call for support.

SQLCSI993   **DB2/VM shutdown requested due to auditing restriction**

A monitor abend has occurred and the SQL/Auditing Facility is active without the UNRESTRICTED option. To prevent non-audited access, DB2/VM shutdown has been requested.

SQLCSI994   **Monitor RELOAD complete**

The requested MONITOR RELOAD command has completed.

SQLCSI0999   **SQL/Statement Monitor is terminating.**

Monitor shutdown in in progress.

24.2 SQLCS Messages

SQLCS001   **SQLCODE xxx CONNECTING TO nn**

SQLCODE xxx occurred when SQLCS attempts to connect the database containing the log table. (The database name in the LOG DATABASE configuration statement).

Start the database if needed and ensure that the necessary privileges have been granted to the virtual machine running the SQLCS program. The connect is retried when the next log request is received. The current log request however is not processed.

SQLCS002   **SQLCODE xxx AFTER nn**

When building the catalog object lists for a monitored database, SQLCODE xxx has occurred after the operation specified in nn.

Ensure that the default DB2/VM userid running the SQLCS program has the SELECT privilege for the catalog tables SYSDBSPACES, SYSCATALOG and SYSINDEXES.

SQLCS003   **IUCV SEND RC IS xx**

When forwarding a "host statement" request to a database server, IUCV rc xx was returned.

Verify the IUCV privileges of the virtual machine running SQLCS.

SQLCS004   **RECEIVE RC IS xx**

When receiving a client request, IUCV rc xx was returned.

Call for support.

SQLCS005   **SQLCODE xxx INSERTING INTO SQLCS_LOG**

SQLCODE xxx was returned during insert to the log table.

Examine the SQLCODE. Call for software support if you cannot recover from the error.

SQLCS006   **IUCV REPLY RC IS xx**

When replying to a client request, IUCV rc xx was returned.

Call for software support.

SQLCS007   **SQLCODE xxx INSERTING PROGSTATS**

SQLCODE xxx was returned when inserting program statistics into the SQLCS_LOG table.

Examine the SQLCODE. Call for software support if you cannot recover from the error.

SQLCS008   **SQLCODE xxx UPDATING PROGSTATS**

SQLCODE xxx was returned when updating program statistics on the SQLCS_LOG table.

Examine the SQLCODE. Call for software support if you cannot recover from the error.

SQLCS009   **{Client | Server} mode HNDIUCV RC is xx**

CMS returns RC xx when establishing the IUCV environment.

Call for software support.

SQLCS010   **SQLCODE xxx BUILDING SQL_STMNTS**

SQLCODE xxx was returned when inserting statement statistics into the SQLCS_SQL_STMNTS table during an end_of_statement log.

Examine the SQLCODE. Call for software support if you cannot recover from the error.

SQLCS011   **SQLCODE xxx SELECTING SQL_STMNTS**

SQLCODE xxx was returned when retrieving the statement text from the SQLCS_SQL_STMNTS table during logging or benchmarking.

SQLCS012   **SQLCODE xxx BUILDING SQL_STMNTS**

SQLCODE xxx was returned when inserting a package into the table SQLCS_SQL_STMNTS during logging or benchmarking.

SQLCS013   **SQLCODE xxx SELECTING SQL_STMNTS**

SQLCODE xxx was returned when retrieving the statement text from the SQLCS_SQL_STMNTS table when building the source file for the SQL/Command Analysis call.

SQLCS014 **SQLCODE xxx SELECTING SYSDBSPACES**

SQLCODE xxx was returned when retrieving the Dbspace name during object identfication.

SQLCS015 **SQLCODE xxx SELECTING SYSCATALOG**

SQLCODE xxx was returned when retrieving the tablename during object identfication.

SQLCS016 **SQLCODE xxx SELECTING SYSINDEXES**

SQLCODE xxx was returned when retrieving the indexname during object identfication.

SQLCS017 **SQLCODE xxx CONNECTING yyy**

SQLCODE xxx was returned when connecting to database yyy during object identification.

SQLCS018 **SQLCODE xxx CREATING SYSCAT_TABID INDEX**

SQLCODE xxx was returned when creating the index SYSCAT_TABID. See under message SQLCS019 for further explanation.

SQLCS019 **CREATING INDEX SYSCAT_TABID**

The index SYSCAT_TABID on SYSCATALOG(DBSPACENO,TABID) is created during product installation. However, SQLCS will create it automatically during execution, when it finds that the index does no longer exist in a given database. The index is used for performance reasons only, namely to allow fast retrieval of a tablename using the internal table identification.

SQLCS020 **Virtual storage less than 16 Mb**

16 megabytes of virtual storage is required to run the SQLCS program. The program terminates following this message.

SQLCS021 **Abend limit exceeded**

SQLCS restarts itself after an abend. However, continuous (more than 4) abend restarts are detected. In this case the above message is issued and SQLCS goes into a dormant state. Full program functionality is re-attempted at the begin of a new session (which occurs at midnight by default). If the error could be corrected earlier, erase the file ABEND COUNTER on the A-disk of SQLCMDM. This will resume normal operation.

SQLCS022 **Database nnnnnnnn forced read_only.**

The message is issued as a result from the FORCE_RO configuration statement.

SQLCS091   **Deleting log entries older than n days**

Message issued during SQLCS startup. Indicates that the LOG RETAIN parameter is being processed.

SQLCS092   **Deleting ProgStats older than n days**

Message issued during SQLCS startup. Indicates that the LOG RETAIN parameter is being processed.

SQLCS093   **Auto-restart in progress**

Informatory message indicating that SQLCS automatic restart is in progress. By default, auto-restart occurs at midnight.

SQLCS096   **Performing session initialization for database N**

The first SQL statement executed in a database session causes the catalog object lists to be built.

SQLCS097   **Session initialization for database N complete.**

SQLCS098   **CMS file system rc n writing reload file**

A write error occurred when SQLCS processes reprepped programs. There is probably insufficient space on the A-disk to contain the unloaded program text.

SQLCS100 **Syntax error in file <databasename RESTRICT> at keyword <keyword>**

A statement in the restrict file has invalid syntax and is ignored.

SQLCS101 EXCLUDE syntax error in file <databasename RESTRICT> at keyword <keyword>

An EXCLUDE statement in the restrict file has invalid syntax and is ignored.

SQLCS102 INCLUDE syntax error in file <databasename RESTRICT> at keyword <keyword>

An INCLUDE statement in the restrict file has invalid syntax and is ignored.

Date is mm/dd/yy, time is hh:mm:ss
SQLCS103 **<database><SQLuser> agent <number> forced due to <restriction_condition>**
SQLCS103 **Program name is <name>. Statement started at <starttime>**

The named restriction has been applied to the named user. The named DB2/VM userid has been forced.

SQLCS104 **No restrictions in effect for database <databasename>**

No restrict file was found for the named database.
Informatory message displayed during SQLCS startup.

SQLCS105 **Restriction file databasename RESTRICT in effect for database <databasename>**

The named restrict file is in effect for the named database.
Informatory message displayed during SQLCS startup.

SQLCS997 **Maximum access time exceeded for SQLCS_LOG table**

The access time to the above table was more than 5 seconds during 20 consecutive accesses.

Check that the indexes on this table have been created. See message SQLCS999 for more details. To rebind the SQL/MF packages, reload the SQL/MF packages in the log database, using SQLCSPI.

SQLCS998   **Maximum access time exceeded for SQLCS_SQL_STMNTS table**

The access time to the above table was more than 5 seconds during 20 consecutive accesses.

Check that the index on this table has been created. See message SQLCS999 for more details. To rebind the SQL/MF packages, reload the SQL/MF packages in the log database, using SQLCSPI.

SQLCS999   **SQL/MF index <nn> not in SYSINDEXES.**

The required index nn on one of the SQL/MF monitor tables does not exist at the start of the statement monitor. The statement monitor stops processing and exits to CMS.

Determine why the index was not created. If SQL/MF installation did successfully complete, re-create the named index. The SQL/MF members SQLCSLOG DBS (for the log table) and SQLCSTMT DBS (for the STMNTS table) can be used to retrieve the necessary CREATE INDEX statement.

24.3 SQLCSU Messages

SQLCSU01 **CMSIUCV rc x connecting to y**

When forwarding a service request to virtual machine y, IUCV returncode x was presented. The target virtual machine may be a DB2/VM database server or the machine running the SQLCS program.

Verify that the virtual machine running the SQLCSU program has the necessary IUCV privileges to connect to the target machine named in the above message.

For a description of the IUCV returncodes, refer to page 368.

SQLCSU02 **HNDIUCV RC IS x**

CMS returns RC x when establishing the IUCV environment.

Call for software support.

SQLCSU03 **CMSIUCV rc x connecting to y**

CMS presented returncode x when initiating the IUCV communication with virtual machine y.

For a description of the IUCV returncodes, refer to page 368.

SQLCSU04 **SQLCODE x after SELECT SQLCS_SQL_STMNTS**

SQLCODE x has been returned when the statement detail function attempts to retrieve the statement text from the SQL Statements table.

SQLCSU05 **Timeout on IUCV path to target**

The target IUCV communicator did not respond to a request within the timeout limit.

SQLCSU10 **The above list is not complete.**

Message issued by the SQLPULSE function of SQLCSU. More agents are executing than can be displayed on the SQLPULSE screen.

Return to SQLCSU to view the remaining list entries: SQLPULSE is not able to provide more than one screen of data. SQLCSU on the contrary allows browsing in the running statement list.

24.4 SQLCSLST Messages

SQLCSL01   **PF key unassigned or function not available**

You pressed an unsupported PFkey.

SQLCSL03   **SQL executor returns errorcode xx**

An internal error has occurred. Call for support.

SQLCSL04   **SQL executor returns SQLCODE xx**

SQLCODE xxx was presented when executing the SELECT on the log table.

If you entered a non-standard statement clause for log selection, verify that it has valid SQL syntax.

SQLCSL05   **SELECT list is empty**

The log table contains no rows that satisfy your search criteria.

SQLCSL06   **Insufficient storage. Output list is incomplete.**

There is not enough virtual storage to contain the entire report. Enlarge your virtual storage size or request a more selective report.

24.5 SQLMFM Messages

SQLMF000   **SQLMFM loaded at <address>**
SQLMF000   **Monitoring Session started for today <date>**

Informatory messages issued at startup of the System Monitor.

SQLMF001   **RC xxx when reading SQLMF CTL A**

When reading the SQLMFIN CTL file, the CMS file system returns errorcode xxx.

Verify that the CMS file has valid format.

SQLMF002   **Above control statement invalid**

The control statement listed before this message has no valid format.

SQLMF003   **Too many databases**

SQLMFM cannot monitor more than 255 databases.

SQLMF004   **Database definition missing or invalid**

The DATABASE section of the SQLMFIN CTL file is not valid.

SQLMF005   **Max nr of Dbspaces reached**

SQLMFM cannot monitor more than 10000 Dbspaces per database.

SQLMF006   **CONNECT to <name> returns SQLCODE xxx**

When connecting to one of the databases defined in the SQLMFIN CTL file, SQLCODE xxx has been detected. Connect is retried at each monitoring interval.

SQLMF007   **Error xxx writing SQLMF DSx**

The CMS file system returns errorcode xxx when writing a session dataset. If the errorcode equals 00D, there is no more room on the A-disk of the SQLMFM machine.

SQLMF008   **Error xxx closing SQLMF DSx**

The CMS file system returns errorcode xxx when closing a session dataset.

Call for software support.

SQLMF009   **HNDIUCV RC is xxx**

CMS returns RC xx when establishing the IUCV environment.

Call for software support.

SQLMF010   **IUCV RECEIVE RC is xxx**

CMS returns RC xx when receiving a request from a user executing the SQLMF program.

Call for software support.

SQLMF011   **IUCV REPLY RC is xxx**

CMS returns RC xx when replying to a request from a user executing the SQLMF program.

Call for software support.

SQLMF102   **SQLCODE nnnn IN ROUTINE routine_name**

SQLCODE nnnn occurred when inserting monitor data in the SQL session tables. The routine name specified indicates the table access where the error has occurred.

I000 : SQLMF_SESSION_CTR
I100 : SQLMF_COUNTS
I200 : SQLMF_DBSP_COUNTS
I300 : SQLMF_DBXT_COUNTS
I400 : SQLMF_DBSP_USE
I500 : SQLMF_DBXT_USE
I600 : SQLMF_USER_STATE
I700 : SQLMF_LOG_USE
I800 : SQLMF_CHKP_DELAY
I1000 : SQLMF_CONNECTS
I1400 : SQLMF_DSP_COUNTS
DELETE : applying the "RETAIN" parameter.

SQLMF999   **INTERNAL ERROR n AT OFFSET xxxxxx**
Call for software support.

24.6 Messages issued by the operator command interface

SQLMOPC01 HNDIUCV RC is xxx

CMS returns RC xx when establishing the IUCV environment.

Call for software support.

SQLMOPC02 CMSIUCV RC x connecting to y

CMS presented returncode X when initiating the IUCV communication with the database server Y.

For a description of the IUCV returncodes, refer to page 368.

SQLMOPC03 Timeout waiting on command reply from <database-server>
SQLMOPC04 Ensure that this machine is able to perform SMSG commands

The reply to the DB2/VM operator command did not arrive within the expected time period from the target database server. Since the server returns the reply using the SMSG command and since this command is subject to external security checking (such as RACF), it should be ensured that the server has been authorized to perform SMSG.

24.7 Messages issued by the Recorder Writer

SQLCRW01   HNDIUCV RC is xxx

CMS returns RC xxx when establishing the IUCV environment.

Call for software support.

SQLCSCRW02 RC x after ALSERV ADD

Where x is the returncode from the CP ALSERV ADD macro, as follows:

4   There are no unused entries in the host access list that can be used to establish the new ALE.

8   The specified ASIT does not identify a currently-existing address space for which your virtual machine is authorized for the requested type of access.

SQLCSCRW03 CMS returncode xxx writing lockwait recorder file <fn> <ft>

A CMS file system error occurred. See page 369 for details.

SQLCSCRW04 Recorder file <fn> <ft> is full

This message should not occur. When a recorder file is full, due to a mismatch between the RECORDER MAXSIZE statement and the actual size of the A-disk, the CMS message "A-disk full" is issued, but the recorder wraparound mechanism is applied, to delete the oldest recorder extent.

Call for software support.

SQLCSCRW05 Recorder MAXSIZE is xxx MB

Informatory message stating the computed maximum size of the recorder file when RECORDER MAXSIZE has been specified as a percentage.

SQLCSCRW06 Lock Recorder MAXSIZE is xxx MB

Informatory message stating the computed maximum size of the lockwait recorder file when RECORDER TRACELOCKS has been specified as a percentage.

SQLCSCRW07 Timeout during IUCV CONNECT to xxx

A timeout has occurred when the recorder writer connects to the database server xxx. Ensure that the server is started.

SSPQDM300   **RC x creating x dataspace**

Failure to create a private recorder dataspace.
For an explanation of the returncode, refer to message SQLCSI300.

SSPQDM301   **RC x after ALSERV ADD**

Failure to add the ALET for a newly created private recorder dataspace.
For an explanation of the returncode, refer to message SQLCSI301.

SSPQDM303   **RC x after ALSERV REMOVE**

Failure to remove the ALET for a dropped private recorder dataspace.
For an explanation of the returncode, refer to message SQLCSI303.

SSPQDM304   **Dataspace x is full**

Increase the "private_size" operand on the RECORDER DSPSIZE statement in
SQLCS CONFIG.

24.8 IUCV Returncodes

For a complete description of the IUCV returncodes, refer to the IBM manual CP Programming Services (SC24-5520-02) in the chapter IUCV Function Descriptions - CONNECT function. Use the last 2 digits of the IUCV code displayed to locate the IPRCODE in the manual.

The recoverable IUCV error codes are the following:

1011	The target communicator is not logged on.
1012	The target communicator is logged on but is not enabled for IUCV.
1013	Maximum number of IUCV connections for the source communicator exceeded. Specify a larger MAXCONN in the VM/ESA directory.
1014	Maximum number of IUCV connections for the target communicator exceeded. Specify a larger MAXCONN in the VM/ESA directory.
1015	The source communicator is not authorized (in the VM directory) to connect to the target communicator

For all other IUCV error codes, call for software support.

24.9 CMS File System Returncodes for Read

1   File not found, disk not accessed, or insufficient authority.

2   Invalid buffer address.

3   I/O operation to a minidisk failed.

4   First character of file mode is illegal.

5   Number of records to read is equal to zero.

7   AFT is not marked with a record format of F or of V. If the file was not previously opened, this indicates that the file has an invalid record format.

8   Successful operation, but the buffer was too small to hold all of the requested data.

11   Number of records to read is not exactly one for a file with variable-length records.

12   No records were read because end of file was reached or because the position parameter specified a record number greater than the number of records in the file.

13   Found an invalid displacement in the AFT for a file with variable-length records (this indicates a coding error: it should not occur).

20   Invalid character detected in file name.

21   Invalid character detected in file type.

25   Insufficient free virtual storage available for file system control blocks

26   Position is negative, the number of records to read is negative, or position plus the number of records to process exceeds the file system capacity.

29   Storage group space limit reached.

30   Some error, other than those in this list of codes, occurred while accessing an SFS file. No rollback occurred.

31   Rollback occurred while trying to access an SFS file. The work unit ID on which the rollback occurred is the default work unit ID at the time the file was opened by the first operation to the file.

40   One of the following errors occurred:

          A required CSL routine was dropped.
          A required CSL routine was not loaded.
          There was an error in a user exit routine.
          There was an error calling the user accounting exit routine

42   The variable length record read is invalid.

48   File is empty.

49   External object cannot be opened.

50   File is in DFSMS/VM migrated status and implicit RECALL is set to OFF.

51   Error occurred during DFSMS/VM file recall processing.

55   APPC/VM error.

70   SFS file sharing conflict or minidisk file is already open by DMSOPEN or DMSOPDBK with
          an output intent.

80   I/O error accessing OS dataset.

81   OS read password protected dataset.

82   OS dataset organization is not BSAM, QSAM, or BPAM.

83   OS dataset has more than 16 extents.

84   Attempt to read a file on an OS or DOS formatted minidisk.

99   A required system resource is unavailable for one of the following reasons:

          There is insufficient virtual storage for the file pool server.
          The file pool server is unavailable.
          File is in migrated status and DFSMS is not enabled.

IBM Source: CMS Application Development for assembler
                SC24-5453-02

If you can't find the returncode appearing in the SQL/AF error message in the above list, please refer
to the Chapter CSL / FSF Return Codes in the IBM manual VM System Messages and Codes.

24.10   CMS File System Returncodes for Write

1   Not authorized to write to file.

2   Invalid buffer address.

3   I/O operation to a minidisk failed.

4   First character of file mode is illegal or disk not accessed.

5   Second character of file mode is illegal.

6   The last record number to be written is too large (more than 65535) to fit in a halfword and an extended plist is not specified.

7   Position specifies a record number that is more than one greater than the current number of records in a file with variable-length records.

8   Size of output buffer is not greater than zero or an attempt was made to write a null record to a file with variable length records.

11   FSCB is not marked with a record format of F nor of V.

12   Disk or directory not accessed R/W.

13   Disk is full.

14   Size of output buffer is not evenly divisible by the number of records for a file with fixed-length records.

15   Attempt to alter the record length of a file with fixed-length records.

16   Record format specified not the same as file.

17   Size of output buffer is greater than 65535 for a file with variable-length records.

18   Number of records to write is not exactly one for a file with variable-length records.

20   Invalid character detected in file name.

21   Invalid character detected in file type.

25   Insufficient free storage available for file system control blocks

26   Position specifies a negative record number or number of records to write is negative or position plus the number of records exceeds the file system capacity ($2^{31} - 1$) or logical block number computed by system exceeds the file system capacity ($2^{31} - 1$).

29   The storage group space limit was reached.

30   Some error, other than those in this list of codes, occurred while accessing an SFS file. No rollback occurred.

31   Rollback occurred while trying to access an SFS file.

38   File explicitly opened with read intent.

39   A disk is accessed as a read only extension of another, and a given file exists on the extension disk but not on the parent disk.

40   One of the following errors occurred:

        A required CSL routine was dropped.
        A required CSL routine was not loaded.
        There was an error in a user exit routine.
        There was an error calling the user accounting exit routine

49   External object cannot be opened.

50   File is in DFSMS/VM migrated status and implicit RECALL is set to OFF.

51   Error occurred during DFSMS/VM file recall processing.

55   APPC/VM error.

70   One of the following sharing conflicts occurred:

        The file is locked.
        The file pool server detected a deadlock.
        The file is open for write through SFS OPEN.
        The file is open for write by another user.
        You attempted to write to a file that is currently implicitly open for READ, but the file has been changed since it was originally opened.
        The minidisk file is already open by DMSOPEN or DMSOPDBK when issuing an FSWRITE.

80   I/O error accessing OS dataset.

81   OS read password protected dataset.

82   OS dataset organization is not BSAM, QSAM, or BPAM.

83   OS dataset has more than 16 extents.

84   Attempt to write a file on an OS or DOS formatted minidisk.

99 A required system resource is unavailable for one of the following reasons:

There is insufficient virtual storage for the file pool server.
The file pool server is unavailable.
File is in migrated status and DFSMS is not enabled.

IBM Source: CMS Application Development for assembler
SC24-5453-02

If you can't find the returncode appearing in the SQL/AF error message in the above list, please refer to the Chapter CSL / FSF Return Codes in the IBM manual VM System Messages and Codes.

25 Glossary

access path
To access a table, DB2/VM automatically chooses an access strategy. A table can be accessed by a DBspace scan, by a non-selective index scan or by a selective index scan. The access path adopted is stored in the package. To record the access path for a given SQL statement during its execution, SQL/MF scans the storage-resident DB2 package.

agent
A structure allocated by the database manager to enable the processing of requests from a DB2/VM client. There are always 3 system agents. A fourth system agent is active in a TCP/IP environment. Each active DB2/VM user session is represented by a user agent. SQL/MF monitors user agents only.

blocking
During APPC communications between a database server and a client, the blocking option causes multiple table rows to be transmitted in a single APPC transaction. Without blocking, each table row to be passed requires an APPC transaction. Blocking is a facility to be requested at preprocessing time. It may drastically reduce the communications overhead between client and server. Please note that only cursor-based statements are eligible for blocking. Furthermore, even when blocking has been requested, DB2/VM may force non-blocking, for example, when the statement uses long fields, during a SELECT FOR UPDATE etc. The default blocksize in a DB2/VM-only protocol is 8 KB.

buffer lookup
DB2/VM holds the most recently accessed data in a storage area called the bufferpool to avoid I/O when the same data is accessed repeatedly. Before attempting I/O or retrieval from a dataspace, the database manager performs a buffer lookup to find the requested data.

statement type
The type of the statement is a character string determined by SQL/MF from the call parameters at the time of the SQL request (OPEN, FETCH, CLOSE and so on).

communications wait
The time elapsed between the end of the previous statement and the start of the current statement (expressed in milliseconds). During that time, the agent waits for the arrival of a new SQL request on the APPC communications path. Therefore, the communications wait time includes application processing time by the client.

CPUtime used
The processor time used by the database server to process the current statement (expressed in milliseconds).

data pages read
The number of pages read from disk or from the dataspaces (if the latter facility has been installed).

data pages written
The number of pages written to disk or to the dataspaces (if the latter facility has been installed). Please note that a statement changing the database does not initiate an immediate write to disk. Rather, the database manager flags the corresponding bufferpool page as modified and writes the page to disk when the logical unit of work ends or when another application steals the bufferpool page.

dataspace pagefault
The situation where a required dataspace page is not in main storage and has to be retrieved from the dataspaces. While the page is being retrieved, the database manager can service other users.

DBSS call
The user's SQL request is processed by RDS (the Relational Data System) which calls DBSS (the Database Subsystem) to retrieve data from the database. Complex SQL statements may require multiple DBSS calls during a single RDS call. The (DBSS_call/RDS_call) ratio is a performance analysis parameter.

deadlock
The situation where multiple users lock the database in such a manner that an unending wait situation would arise. The database manager forces one of the users involved.

directory buffer lookup
The buffering technique described under Buffer Lookup above also applies to the retrieval of directory pages.

directory blocks read
The number of directory pages read from disk or from the dataspaces (if the latter facility has been installed).

directory blocks written
The number of pages written to disk or to the dataspaces (if the latter facility has been installed). See the remark under Data pages written above.

dispatcher calls
When an agent must wait for an event to complete, it calls the DB2/VM dispatcher which continues work for other users. A dispatcher call is made when the agent must wait for I/O completion, for unlocking of a locked resource, for the arrival of the next SQL request from the client and so on.

elapsed time
The wall-clock time elapsed between the begin and the end of an SQL statement. In case of cursor-based statements, the opening of the cursor is used as begintime.

hit ratio's

page_buffer_hit_ratio:
shows the efficiency of page buffer pool access as the percentage of page buffer lookups that could be satisfied without needing I/O or dataspace access.

directory_buffer_hit_ratio:
shows the efficiency of directory buffer pool access as the percentage of directory buffer lookups that could be satisfied without needing I/O or dataspace access.

dataspace_hit_ratio:
shows the efficiency of dataspace paging as the percentage of dataspace reads and writes that did not cause a dataspace pagefault.

dataspace_access_hit_ratio:
shows the efficiency of dataspace access as the percentage of buffer lookups that did complete without causing a dataspace pagefault.

host variables
Those elements of the SQL statement that are unknown when the statement is preprocessed by the database server and that are specified at execution time. When showing monitored statements, SQL/MF substitutes the host variable names with their contents.

internal Dbspace
The database manager uses internal ("temporary")Dbspaces for various purposes, such as sorting, joining and so on. Internal DBspaces are always accessed by relational scan. Therefore, unnecessary use of those Dbspaces may have a negative performance impact.

I/O wait time
The accumulated time (in milliseconds) that an agent has waited for the completion of its I/O requests or for the completion of dataspace pagefaults. For tables located in dataspaces, I/O operations are performed asynchronously by the VM control program on request of the database server, not by DB2/VM itself. Therefore, such operations are not counted in the I/O wait time.

isolation level
The isolation level may take the values RR (repeatable read), CS (cursor stability), RS (read stability) or UR (uncommitted read).

lock escalation
The event where a large number of locks is changed by the database manager into a lock of a higher level.

lock wait
The number of times a resource lock request from an agent results in a wait state, because the resource is in use.

lock wait time
The total time (in milliseconds) that an agent had to wait for locked resources.

log pages read
The number of pages read from the DB2/VM log.

log pages written
The number of pages written to the DB2/VM log.

package
The collection of a number of executable SQL statements, as produced by the DB2/VM preprocessor. Previously called access module.

package section
Each statement in a package is called a section and receives a section number during preprocessing. A single section number is allocated by DB2/VM to cursor-based statements (open, fetch, close). SQL/MF adds a dummy section, numbered -1, to contain the processing cost associated with package loading, run privilege checking and auto-reprepping, if applicable.

package unload

During product installation, all packages are unloaded and stored in the SQL_STMNTS table. For packages prepped (or reloaded) while SQLCSI is active, the SQL_STMNTS table is updated automatically. For packages prepped while SQLCSI was inactive, a DBSU program unload will be performed automatically to update the SQL_STMNTS table.

RDS call

An application request is received and processed by the Relational Data System. See also DBSS call above.

VM BLOCKIO request

An access to data that is not in DB2/VM dataspaces but that is retrieved from disk. Note that the corresponding statistic is available with DB2/VM dataspaces only.