DB2 UDB for Linux, UNIX, and Windows Performance and Tuning

Module 1
Tablespace Design for Performance

DISCLAIMER

- This material references numerous hardware and software products by their trade names. References to other companies and their trademarks are for information purposes only, and all trademarks are the properties of their respective companies.
- DB2 is a registered trademark of IBM Corporation.
- AIX is a registered trademark of IBM Corporation.
- UNIX is a registered trademark of the Open Group.

Objectives

- Upon completion of this module you should be able to:
  - Understand the DB2 Storage Model
    - Systems Managed Space
    - Database Managed Space
  - Containers
  - DB2 striping

Objectives Continued

- Understand the following:
  - how to place containers on various types of disk
  - DB2 use of extents, extentsize, and meaning of DB2_PAGE_CONTAINER_TAG, DB2_PARALLEL_IO registry variables
  - Use of Raw Devices

Objectives Continued

- Understand the following:
  - Direct I/O
  - Concurrent I/O
- Use of multi-page file allocation multipage_alloc (emphasis with MDC)
- Synchronous I/O
- Asynchronous I/O

Tablespace Design

- SMS Tablespaces
  - Managed by Operating System
  - Directories
  - Allocated a page at a time
  - Cannot add containers except via redirected restore
  - Can’t separate indexes and data
  - Cannot use raw devices

Tablespace Design

- DMS tablespaces
  - Managed by DB2
  - Allocated extent-at-a-time
  - Formatted ahead of time
  - Separation of indexes and data
  - Add containers on the fly
  - Add container without requiring a rebalance (Begin New Stripe Set)

Tablespace Design – DMS Continued

- Use of Raw devices
- Can extend/resize/drop container
- Can breakout indexes and data into separate bufferpools

Tablespace Basics

- Tablespaces use files or “containers” to store table and index data
- DMS Device containers refers to RAW Devices
  - DB2 takes control of the device
- EXTENTSIZE
  - Number of pages written to a container or prefetched from a container

Tablespace Basics

- PREFETCHSIZE
  - Is the number of pages to be read from a tablespace when prefetching is used
    - Prefetching is used to pre-stage data into the bufferpool when the optimizer determines that data access is sequential
    - Application doesn’t have to wait for the data

Tablespace Basics

- Prefetch size should be set to a multiple of the number of containers being used
  - Example:
    - EXTENTSIZE = 32, 4 containers
    - Set PREFETCHSIZE size to 128
  - Enables 4 extent size prefetch requests to be issued in parallel
- For OLTP environments smaller extent size is preferred

Tablespace Basics

- Overhead and transfer rate
  - These values are used by the optimizer to determine the costs of I/O during query optimization
  - Both values are measured in milliseconds
  - Default for overhead is 24.1
  - Default for transfer rate is 0.9
- These defaults are outdated and should not be used

Tablespace Basics

- For 10K RPM disks, specify OVERHEAD=8 and FOR 4K pages a TRANSFERRATE=0.1
- For 15K RPM disks specify OVERHEAD=6
- If the tablespace is spread across disks with different performance characteristics, choose an average of all the values

Tablespace Design

- When developing your tablespace strategy, some things to think about
- How many tables per tablespace?
- Where should indexes be placed?
- Bufferpool strategy
- Block-based bufferpool considerations
- Recovery and integrity considerations
  - Data and index must be recovered together

Tablespace Design

- From a performance standpoint, one table per tablespace facilitates monitoring and tuning
- Can easily tie tablespace activity to the table!
- At least consider for new tables where behavior and characteristics are not known

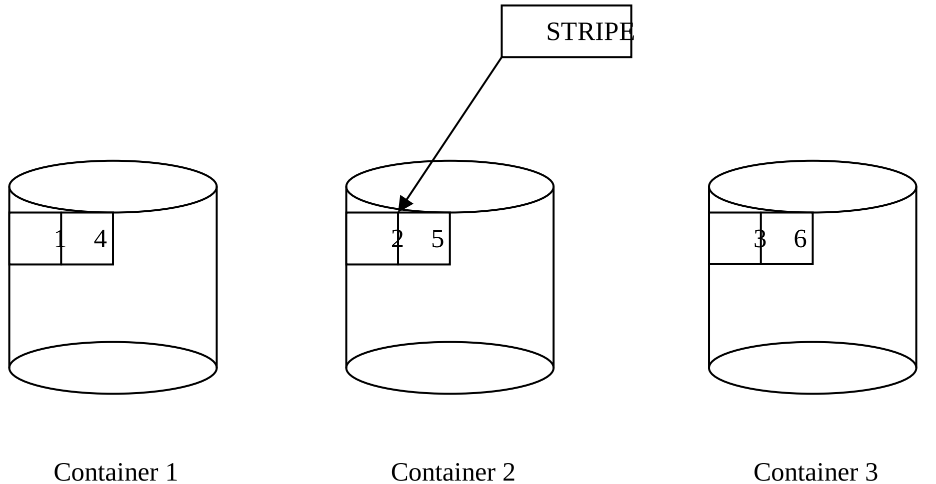
Tablespace Design

- For top performance place indexes in separate tablespace
- Also facilitates separate bufferpool placement
- Which facilitates monitoring and tuning of bufferpools and index tablespaces
  - Access patterns
  - Synchronous/Asynchronous

DB2_STRIPED_ _CONTAINERS

- With this variable set, Pre V8.1 caused DMS container tag to be allocated in an extent and when the extent size is aligned with RAID stripe size provided improved I/O performance
- Now default behavior in V8.1
  - Could use DB2 registry variable DB2_USE_PAGE_CONTAINER_TAG to revert to former behavior, not recommended

DB2 Striping



Container Placement

- Rules of thumb:
  - Spread data over multiple containers over multiple physical disks
  - Multiple adapters and paths for redundancy/performance

Tablespace Design

- “db2 list tablespace”
- “db2 list tablespace containers for 0”

DB2_PARALLEL_IO

- DB2 environmental variable
- Can set to a range of containers or for all
  - “db2set db2_parallel_io=*”
- Set this variable when using RAID-5 storage and single container
  - Enables DB2 to issue parallel read requests

RAW DEVICES

- RAW devices still outperform all other setups
- Logs
- Will not change for short term but things are in the works
- Concurrent I/O!

Storage – The Past

- Enterprise Database servers with onboard disk
- Limited capacity
- Limited Reliability
- Limited Availability

RAID-1

- Disk mirroring
- Writes data in two places
- With dual adapters, high availability
  - Either disk can fail, data is still accessible
  - With duplexing, disk controller can also fail
- High Performance
- Twice the usual number of disks

RAID-5

- Collection of disks in an array (typically 6-7) with parity striping
- Parity data is striped across all disks in the array
- High-end storage solutions use hardware RAID-5 and large cache which reduces write penalty unless cache overrun
- Provides high availability unless 2 disks fail at exact same time

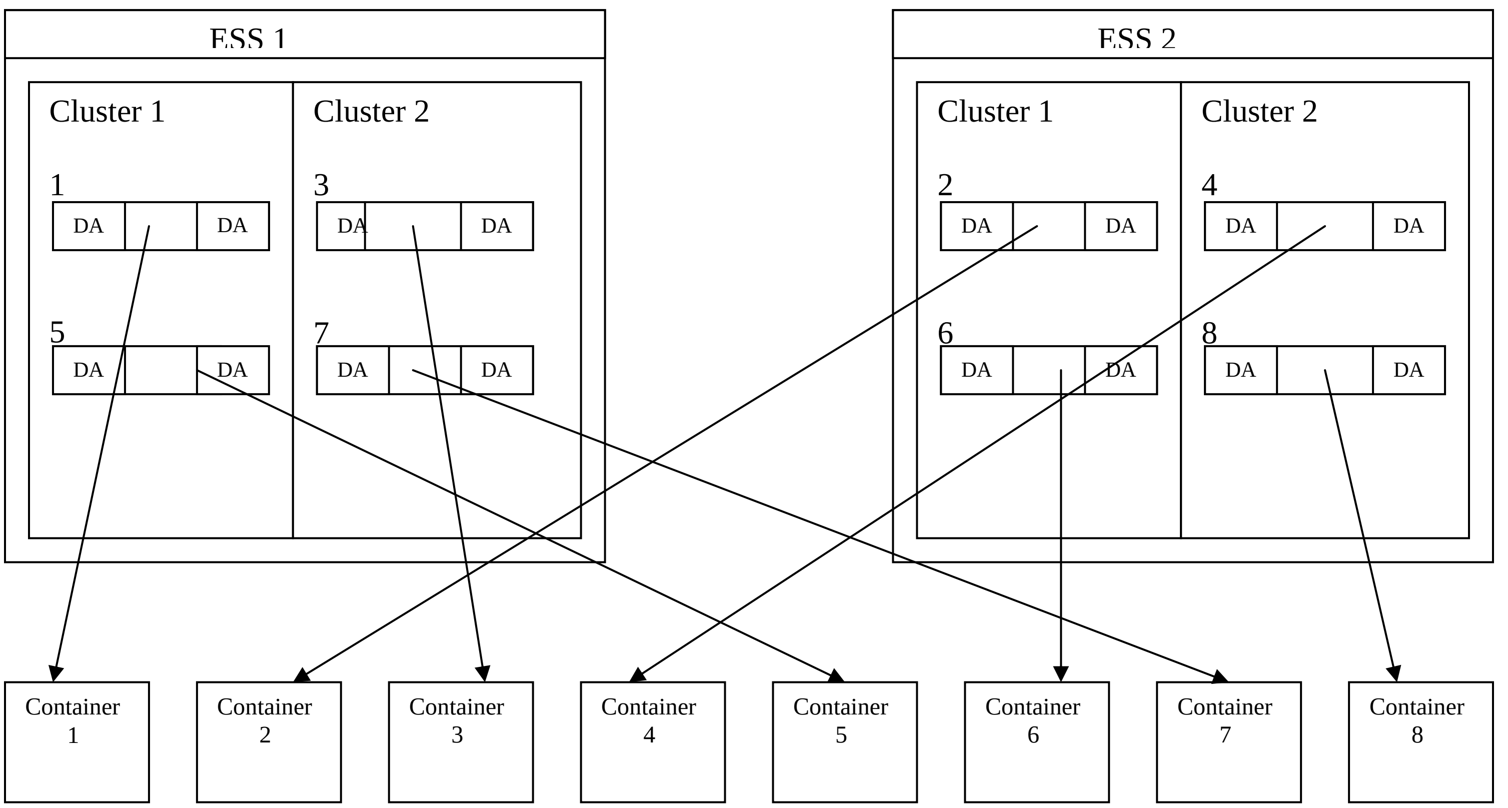
RAID-10

- Implemented as a striped array whose segments are RAID-1arrays
  - Same fault tolerance as RAID-1
  - High I/O rates achieved by striping RAID-1 segments
  - Has same overhead for fault tolerance as with mirroring alone

The Present

- Enterprise Storage
  - Direct Attached
  - SAN
  - NAS

SHARK


ESS 1
Cluster 1
Cluster 2
1
DA
DA
3
DA
DA
5
DA
DA
7
DA
DA
ESS 2
Cluster 1
Cluster 2
2
DA
DA
4
DA
DA
6
DA
DA
8
DA
DA
Container 1
Container 2
Container 3
Container 4
Container 5
Container 6
Container 7
Container 8

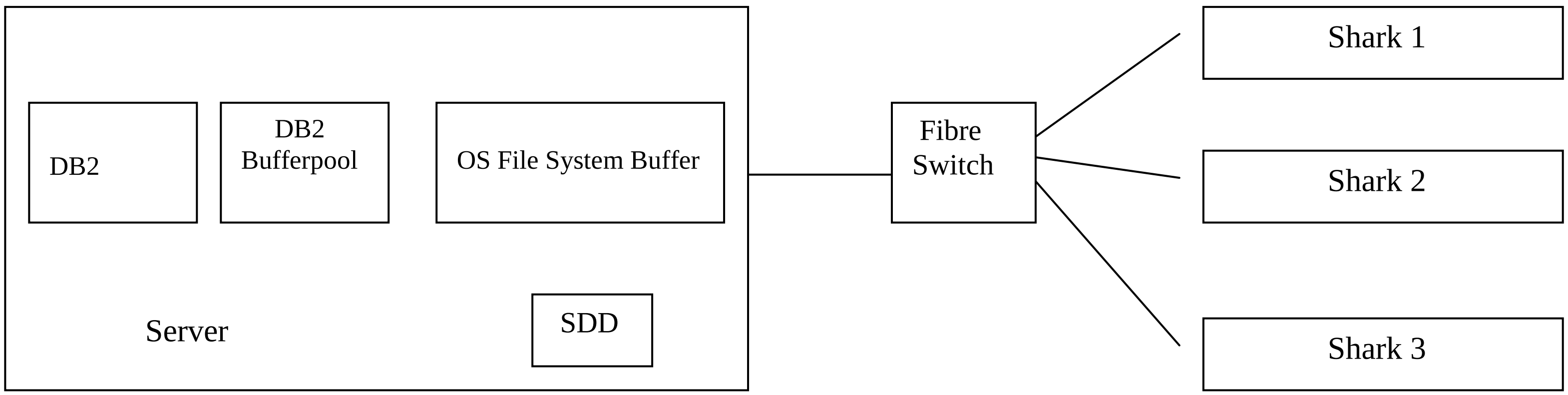
Balanced I/O

- Key for good performance
- Balance I/O across
  - Storage devices (cabinets)
  - Clusters within a cabinet
  - Span disk adapters
  - Engage as many arrays as possible

Present/Future

- SAN
- NAS
  - RAID?

Storage Architecture



SAN

DIRECT I/O

- On AIX, limited Direct I/O support added in DB2 V8.1.4 for all SMS tablespaces except the following:
  - temporary tablespaces
  - Long fields
  - LOBs
- Not yet available for DMS, possibly future release or fixpak
- Previously available on Windows™ for SMS and DMS using the DB2NTNOCACHE registry variable

DIRECT I/O

- Direct I/O improves performance but INODE locking still a serious detriment to performance
- Concurrent I/O introduced in AIX 5L V5.2.0.10 (ML1) May 27, 2003 in the enhanced Journaling File System (JFS2)
- DB2 UDB for Linux, UNIX, and Windows does not support Concurrent I/O as of V8.1.4

DIRECT I/O

- Direct I/O does improve performance by reducing CPU consumption
- But Concurrent I/O is needed to reduce INODE lock contention

CONCURRENT I/O

- Concurrent I/O not yet available for DB2 as of V8.1.4 on AIX
- Is needed to reduce I/O bottleneck
- Uses Direct I/O implicity, and is enabled using similar commands
- Already supported by other relational databases

CONCURRENT I/O

- Reduces INODE lock problems by taking a read-shared lock

ASYNCHRONOUS I/O

- Also know as AIO, enabled by default in DB2 V8.1
- AIO improves the performance of page cleaners
- Also, when using RAW devices, AIO server threads (which are also kernal threads) do not require a context switch as with files and use a “fast path” to the LVM that avoids the use of AIO server threads

Storage Provisioning

- Storage provisioning is concerned with easing the storage administration burden for DBAs and Sytem Administrators
- Future plan
- Based on similar concepts used on mainframe for managing storage
  - System Managed Storage (SMS)
    - Don’t confuse this with System Managed Space (SMS)!

Storage Provisioning

- Designers envision that storage provisioning will generally consist of the following components:
  - Storage templates
  - Dedicated storage pools
  - Quality of Service based on priority of the data
  - Archiving

Storage Provisioning

- Automatic movement of tablespaces based on activity, less important data moved to slower devices and vice versa
- Automatic growing of pools and disk additions

Table Design for Performance

Table Design

- Determine the different kinds of activity that will go against the table
- What are the business priorities?
  - Selects (OLTP response)
  - Updates?
  - Inserts?
- Then tune to meet business objectives

Table Design

- Clustering
  - Regular or MDC?
  - Freespace
  - Data access patterns
  - Consider tradeoffs of not clustering
- Data not stored in any particular order
- If APPEND ON used, can’t have a clustering index
- More reorgs may be required

Table Design

- Consider reducing number of freespace control records (FSCRs) to be searched when looking for freespace for inserts
- Use registry variable to reduce
- If inserts still not fast enough, use APPEND ON
  - Again, this must be based on business priorities

Range Clustered Tables (RCTs)

- DB2 V8.1.4 introduces Range Clustered Tables (RCTs)
  - Implements range partitioning
  - Uses offset or displacement
  - Easier for Oracle™ users to migrate to DB2!

Module 2
Snapshot Monitoring

Objectives

- After completion of this unit you should be able to:
- Identify snapshot monitoring facilities available and be able to enable snapshot monitoring at the instance or application level
- Understand when, why and how to use snapshot monitoring

Objectives

- Understand how DBAs use the output from snapshot monitoring to identify, investigate, and solve performance problems
- Develop a snapshot monitoring strategy for databases you support

Monitoring

- In order to make sure that resources are being used efficiently and to ensure that business requirements are met, continuous monitoring must be practiced
- Use Health Center, Health Monitor, Memory tracker, Snapshot Monitoring and Event Monitoring
- Third Party Vendor Tools

Snapshot Monitoring Architecture

- Controlled by a series of monitoring switches
  - Off by default (except timestamp switch)
- Can be enabled at the instance level or application level
- Each switch adds overhead on database manager

Snapshot Monitoring Architecture

- Controlled by a series of monitoring switches
- Use of snapshot monitoring is DBA “first line of defense”
  - Very difficult to support DB2 without using them
- Kind of like investigating a crime scene, they show evidence of wrong doing but more investigation/analysis required

Snapshot Monitoring

- Authorization Required
  - SYSADM, SYSCNTL, SYSMAINT
- New registry variable in DB2 V8.1.4
  - DB2_SNAPSHOT_NOAUTH
  - "db2set DB2_SNAPSHOT_NOAUTH=ON"
- Allows users without above authority to take snapshot

Snapshot Monitoring

```
Monitor Recording Switches
Switch list for db partition number 0
Buffer Pool Activity Information  (BUFFERPOOL) = ON  03-17-
   2003 15:02:57.169849
Lock Information                        (LOCK) = ON  03-17-
   2003 15:02:57.169849
Sorting Information                     (SORT) = ON  03-17-
   2003 15:02:57.169849
SQL Statement Information          (STATEMENT) = ON  03-17-
   2003 15:02:57.169849
Table Activity Information             (TABLE) = ON  03-17-
   2003 15:02:57.169849
Take Timestamp Information         (TIMESTAMP) = ON  03-17-
   2003 15:02:57.169849
```

Unit of Work Information (UOW) = ON 03-17-2003 15:02:57.169849

Snapshot Monitoring

This page intentionally left blank

Snapshot Monitoring

Database Manager Switch	**Database Switch**	**Information of Interest**
DFT_MON_BUFFERPOOL	Bufferpool	Logical and physical reads, Asynchronous I/O activity, Information with which to compute hit ratios.
DFT_MON_LOCK	Lock	Locks held by applications, lock waits, escalations, deadlocks
DFT_MON_SORT	Sort	Amount of sortheap used, sort overflows, number of sorts, sort time
DFT_MON_STMT	Statement	APPLID, connect time, sorts, DML activity, locks held, bufferpool activity
DFT_MON_TABLE	Table	Read and write activity
DFT_MON_UOW	Uow	Completion status, start and end times.
*DFT_MON_TIMESTAMP	Timestamp	Timestamp for time dependent functions

*New in V8

Snapshot Monitoring

- Snapshot monitor switches can be turned on and off as needed. Use the UPDATE DBM CONFIGURATION USING DFT_ON_TABLE ON or UPDATE MONITOR SWITCHES USING TABLE ON command.

Snapshot Monitoring

- It is important to note that monitor switches must be enabled before issuing a snapshot command
- If the appropriate monitor switches are not enabled at either the instance or database level, DBM and database information is available but not much other information is available
- Refer to the Monitoring Matrix for complete details

GET SNAPSHOT FOR		Tables	Tablespaces	Memory Pools	Bufferpool & I/O	Lock Summary	Lock Detail	Sorts	Agents	CPU Utilization	Rows Read/Selected	Pkg./Sect./Cat.	Application State	SQL Stmt. activity	Log Usage	Dynamic SQL
								P	A							
				A	S	P		P			A	A		A	A	
				A	S	P		P	A	S	A	A	A			
					S											
all			A	A	S	P		P	A	S	A	A	A	A	A	P
database				A	S	P		P			A	A		A	A	
bufferpool					S											
Applicati-ons	On <database>			A	S	P		P	A	S	A	A	A	A		
tables																
tablespace			A		S											
locksl						P	S						A			
Dynamic SQL																P

A- always collected

S- collected only when monitor switch is ON

P- collected when switch is ON, partially collected when switch is OFF

Snapshot Monitoring

- There are eight snapshot levels available as follows:
  - Database – Records information at the database level
  - Database Manager – Records information at the instance level
  - Application – Records application information
  - Bufferpools – Records bufferpool activity
  - Tablespace – Records tablespace activity
  - Table – Records table activity
  - Lock – Records lock information for locks held by applications
  - Dynamic SQL cache – Records point-in-time statement information from the SQL statement cache

Snapshot Monitoring

Snapshot Type	**Command**
Snapshot for locks	“db2 get snapshot for locks on sample”
Database Manager Snapshot	“db2 get snapshot for DBM”
Database Snapshot	“db2 get snapshot for database on SAMPLE”
Tablespace Snapshot	“db2 get snapshot for tablespaces on SAMPLE”
Bufferpool Snapshot	“db2 get snapshot for bufferpools on SAMPLE”
Applications	“db2 get snapshot for applications on SAMPLE”
Dynamic SQL	“db2 get snapshot for dynamic sql on SAMPLE”

Frequently Used Snapshot Commands

Database Manager Snapshot

- A DBM snapshot can be taken by issuing the following command from the CLP:

  db2 get snapshot for database manager

Snapshot Monitoring

- The Database Manager Snapshot output has changed significantly in v8. Heap memory areas are now separately reported for the various heaps. These are identified by the element "Memory Pool Type" followed by the heap being reported. From the previous example,we can see that the package and catalog cache entries have changed.

Snapshot Monitoring

- Key Database Manager snapshot elements to review and to monitor on a regular basis are:
  - Post threshold sorts
  - Pipe sort requests requested
  - Pipe sort requests rejected
  - Agents waiting for a token
  - Agents stolen from other applications
  - Max agents overflow
  - Hash joins after heap threshold exceeded
  - Gateway connection pool agents stolen
  - Size of package cache
  - Size of catalog cache
  - Size of lock managers heap
  - Size of database heap

Database Snapshot

- Key Database snapshot elements to review, monitor, and track on a regular basis are:
  - Lock waits
  - Time database waited on locks
  - Deadlocks detected
  - Lock escalations
  - Lock timeouts
  - Sort overflows
  - Bufferpool data physical reads
  - Bufferpool data writes
  - Bufferpool index physical reads
  - Dirty page steal cleaner triggers

Database Snapshot (continued)

- Key Database snapshot elements to review, monitor, and track on a regular basis are:
  - Dirty page threshold cleaner triggers
  - Direct reads
  - Direct writes
  - Database files closed
  - Failed statement operations
  - Internal rollbacks
  - Internal rollbacks due to deadlocks
  - All row metrics
  - Package cache overflows
  - Catalog cache overflows
  - Number of hash join overflows

DBM and Database Performance Indicators

Monitoring Element	Tuning Tips
Lock waits	The total time an application waited for locks. Use with time database waited on locks to compute average time waiting for a lock. This should be < 10 ms. Look for application sequencing problems, packages bound with RR, and lock escalations to X table locks. Get snapshot on application and locks and find all locks associated with application holding the most locks.
Time database waited on locks	The total amount of elapsed time that applications were waiting for locks. Use last reset time and computed average time applications waited for locks. If lock wait and this parameter are high, you have concurrency issues.
Dead locks detected	Monitor and find application sequencing problems if deadlocks occur frequently.
Lock escalations	Not necessarily a problem but if occur constantly investigate application problems, review size of locklist and max locks.
Lock timeouts	Set locktimeout to 10-30 seconds and monitor. If too many lock timeouts occur, review applications running during this time, review reasons they are timing out and correct application problem, then and only then consider increasing locktimeout.
Sort Overflows	Sort overflows should be < 3% in OLTP. This is difficult to achieve in DW/BI environments. Eliminate sorts through proper indexing and clustering. Since we can't eliminate sorts in DW/BI environments, then tune temporary table space container placement and create multiple containers on separate disks to maximize opportunity for parallel I/O.
Bufferpool data physical reads	In OLTP seek to minimize. In DW/BI, sort overflow tuning I/O tip should be used.
Bufferpool data writes	Bufferpool data write occur to free a page in the bufferpool so another page can be read, and to flush the bufferpool. Consider increasing the size of the bufferpool if bufferpool data writes is high in proportion to asynchronous page writes.
Bufferpool index physical reads	Same as bufferpool data physical reads.

DBM and DatabasePerformance Indicators

Dirty page cleaner triggers	Consider increasing size of bufferpool and number of I/O cleaners. Consider decreasing changpthres if you cannot increase size of bufferpool.
Dirty page threshold cleaner triggers	Indicates number of times chngpgs_thres has been reached and dirty pages written asynchronously do disk. Start with the default and decrease to 20-30% in OLTP environment.
Database files closed	Try to keep this at 0. Unnecessary closing and opening of files incurs unneeded overhead.
Failed statement operations	Can be an indicator of application problems. Investigate with application staff and resolve. Not necessarily a problem but if high frequency is a possible indicator of locking, application, or other problems.
Package cache overflows	Package cache overflows to utilheap, locklist and other dbheap memory. Increase package cache until no overflows occur, but do not over allocate.
Catalog cache overflows	Catalog cache overflows cause table descriptors, etc. to be flushed as needed resulting in I/O if descriptors need to be brought back in. Set catalogcache_sz so that at least 90% hit ratios are observed.
Number of hash join overflows	Hash joins overflow from sortheap through bufferpool to temporary space. Increase sortheap and eliminate unnecessary sorts via elimination and through clustering techniques.

Application Snapshots

- Use application snapshots to monitor details of application activity. This snapshot enables you to identify applications consuming large amounts of CPU and memory.
- Locking, bufferpool, and agent activity are provided for each active application. This information can be used in conjunction with the “*db2 list applications show detail*” command to identify problems.

Tablespace Snapshot

- Tablespace snapshots are very useful for identifying categories of tablespace activity such as asynchronous and synchronous read and write activity, bufferpool activity, logical and physical reads from which a tablespace bufferpool hit ratio can be computed, and direct reads and writes
- Top tablespaces in the database can be identified and targeted for tuning
- “db2 get snapshot for tablespaces on sample”

Table Snapshot

- Table snapshots are useful for identifying the most active tables via rows read and rows written, and tables with page reorgs and overflows
- This snapshot can be used in conjunction with the tablespace snapshot to identify active tables per tablespace

Bufferpools

- Accessing memory is extremely fast, as most memory chips commercially available deliver performance that is measured in nanoseconds, often 70ns or faster. Contrast this to the time typically required to access disk storage devices, which is commonly measured in milliseconds. Good disk performance is typically measured in the 3ms to 7ms range, suggesting that accessing memory for data is about 1000 times faster than accessing disk storage

Bufferpool Snapshot

- For demonstration purposes this is an example of a bufferpool snapshot without the bufferpool snapshot switch enabled
- As you can tell from this page and the next, basically no data is available without the monitoring switch enabled

Bufferpool Snapshot continued

This page intentionally left blank

Bufferpool Snapshot

- Issued “db2 get monitor switches” after not receiving any output from previous snapshot
- Status of monitoring switches:
  – Monitor Recording Switches

Switch list for db partition number 0

Buffer Pool Activity Information (BUFFERPOOL) = OFF

Lock Information (LOCK) = OFF

Sorting Information (SORT) = OFF

SQL Statement Information (STATEMENT) = OFF

Table Activity Information (TABLE) = OFF

Take Timestamp Information (TIMESTAMP) = ON 01-11-2004 12:04:23.004198

Unit of Work Information (UOW) = OFF

Bufferpool Snapshot

- Issued the command to enable the bufferpool monitoring switch
  - “db2 update monitor switches using bufferpool on”

  Monitor Recording Switches

  Switch list for db partition number 0

  Buffer Pool Activity Information (BUFFERPOOL) = ON 01-12-2004 03:48:55.499393

  Lock Information (LOCK) = OFF

  Sorting Information (SORT) = OFF

  SQL Statement Information (STATEMENT) = OFF

  Table Activity Information (TABLE) = OFF

  Take Timestamp Information (TIMESTAMP) = ON 01-11-2004 12:04:23.004198

  Unit of Work Information (UOW) = OFF

- Then issued the “db2 get snapshot for bufferpools on sample” snapshot command (Refer to Example 7)

Bufferpool Snapshot

- Key Performance Indicators Continued

Bufferpool Snapshot

- Key Performance Indicators Continued

Bufferpool Snapshot

- Block-based bufferpools
  - New in DB2 UDB V8.1
- Causes non-contiguous pages to be prefetched into contiguous area of memory
  - BLOCKSIZE and NUMBLOCKPAGES specified on create or alter bufferpool command
  - NUMBLOCKPAGES used to specify amount of bufferpool to be reserved for sequential blocks
    - Cannot exceed 98% of bufferpool size
- Use for bufferpools with sequential “characteristics”

Lock Snapshot

- The snapshot for locks shows all locks held in the database by application handle and application id, along with the number of locks held and the status of the application
- Used in conjunction with “list applications show detail” command may enable you to identify and solve locking problems
- More than likely, a deadlock event monitor with details will be more helpful

Dynamic SQL Snapshot

- Dynamic SQL snapshots add overhead to the database manager and should be used only as needed. Output should be written to a file for detailed analysis. Upi can use Dynamic SQL snapshots to find and investigate SQL statements with high costs, high number of rows read, and sorts.

Resetting Monitoring Switches

- An application can reset monitor switches, which in effect resets the counters to 0 for the application issuing the reset command. Note: An application in this respect could be the CLP, Command Center, Third Party Vendor Tool, or a user-written application.
- This can be accomplished by an application issuing the RESET MONITOR FOR DATABASE <database name> command. The GET MONITOR SWITCH command can be used to display the current status of monitoring switches. The RESET MONITOR ALL command can be used to reset the monitor switches for <u>all databases in an instance</u>.

Snapshot Monitoring

- Every application has its own copy of the snapshot monitor values. Resetting the monitor switches only effects the counters of the application that issues the reset.

Snapshot Monitoring via New SQL Functions in V8.1

- New in v8 is the ability to issue snapshot commands via SQL functions. These 15 new functions make it easy for you to write programs that issue snapshot commands and process the snapshot data. They can also be issued via the CLP. Previously this was only available via the administrative API.
- This is a real breathrough in monitoring capability in DB2 and along with write-to-table event monitors provides a historical repository for use in reviewing performance problems, trend and historical analysis

SQL Snapshot Functions

- Below is the syntax and examples can be found in the next few pages:

SNAPSHOT_BP(<dbname>, <partition>)

Returns bufferpool information

```
SELECT DIRECT_READS FROM
    TABLE(SNAPSHOT_BP('GUNNDB', -2)) AS S;
```

SQL Snapshot Functions

Function	Definition of Output Data
SNAPSHOT_AGENT	Application information associated with agents.
SNAPSHOT_APPL_INFO	General application level identification for each application connected to the database.
SNAPSHOT_APPL	Application information. Counters, status information and most recent SQL statement (statement monitor switch must be on)
SNAPSHOT_BP	Physical and logical reads, asynchronous and synchronous writes, counters.
SNAPSHOT_CONTAINER	Tablespace container information.
SNAPSHOT_DATABASE	Database information, counters, sorts, lock escalations, memory heaps.
SNAPSHOT_DBM	Database Manager information, sort overflows, dbheap, locklist heap, other memory heaps.
SNAPSHOT_FCM	Database manager level information regarding FCM resources.
SNAPSHOT_DYN_SQL	Dynamic SQL from SQL statement cache.
SNAPSHOT_FCMNODE	Database manager information regarding FCM for a particular partition.
SNAPSHOT_LOCK	Information at the database level and application level for each application connected to the database.
SNAPSHOT_LOCKWAIT	Lock wait information for applications.
SNAPSHOT_STATEMENT	Application and statement information including most recent SQL statement executed.
SNAPSHOT_SUBSECT	Application information regarding the subsections of access plans for the applications connected to the database.
SNAPSHOT_TABLE	Table activity information at the database and application level for each application connected to the database. Table activity information at the table level for each table that was accessed by an application connected to the database.
SNAPSHOT_TBS	Information about table space activity the database level, the application level for each application connected to the database, and the table space level for each table space that has been accessed by an application connected to the database.
SNAPSHOT_SWITCHES	Database manager monitor switch settings.
SNAPSHOT_QUIESCER	Information about quiescers at the table space level.

SQL Snapshot Functions

- SQL snapshot functions provide DBAs with a way to integrate snapshot history into overall monitoring strategy
- Can be included in a script and run periodically throughout the day, recording snapshot output in DB2 tables
- Can be used for problem determination and for trending and historical purposes

SQL Snapshot Table DDL

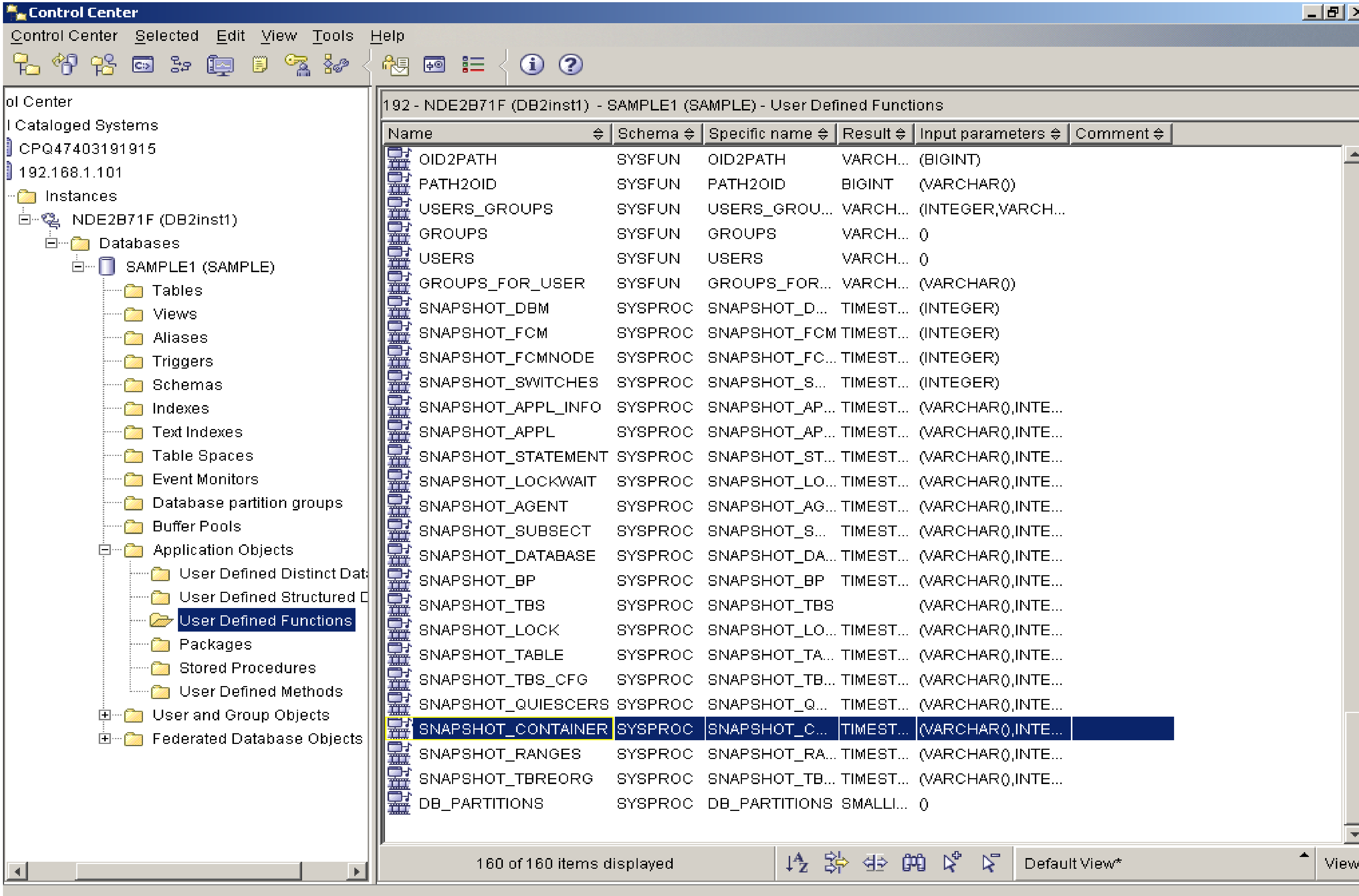
Column	**Data Type**
SNAPSHOT_TIMESTAMP	TIMESTAMP
POOL_DATA_L_READS	BIGINT
POOL_DATA_P_READS	BIGINT
POOL_DATA_WRITES	BIGINT
POOL_INDEX_L_READS	BIGINT
POOL_INDEX_P_READS	BIGINT
POOL_INDEX_WRITES	BIGINT
POOL_READ_TIME	BIGINT
POOL_WRITE_TIME	BIGINT
POOL_ASYNC_DATA_RD	BIGINT
POOL_ASYNC_DT_WRT	BIGINT
POOL_ASYNC_IX_WRT	BIGINT
POOL_ASYNC_READ_TM	BIGINT
POOL_ASYNC_WR_TIME	BIGINT
POOL_ASYNC_DT_RDRQ	BIGINT
DIRECT_READS	BIGINT
DIRECT_WRITES	BIGINT
DIRECT_READ_REQS	BIGINT
DIRECT_WRITE_REQS	BIGINT
DIRECT_READ_TIME	BIGINT
DIRECT_WRITE_TIME	BIGINT
POOL_ASYNC_IX_RDS	BIGINT
POOL_DATA_ESTORE	BIGINT
POOL_INDEX_ESTORE	BIGINT
POOL_INDEX_ESTORE	BIGINT
POOL_DATA_ESTORE	BIGINT
UNREAD_PREF_PGS	BIGINT
FILES_CLOSED	BIGINT
BP_NAME	CHAR (18)
DB_NAME	CHAR (8)
DB_PATH	VARCHAR (255)
INPUT_DB_ALIAS	CHAR (8)

SQL Snapshot Functions

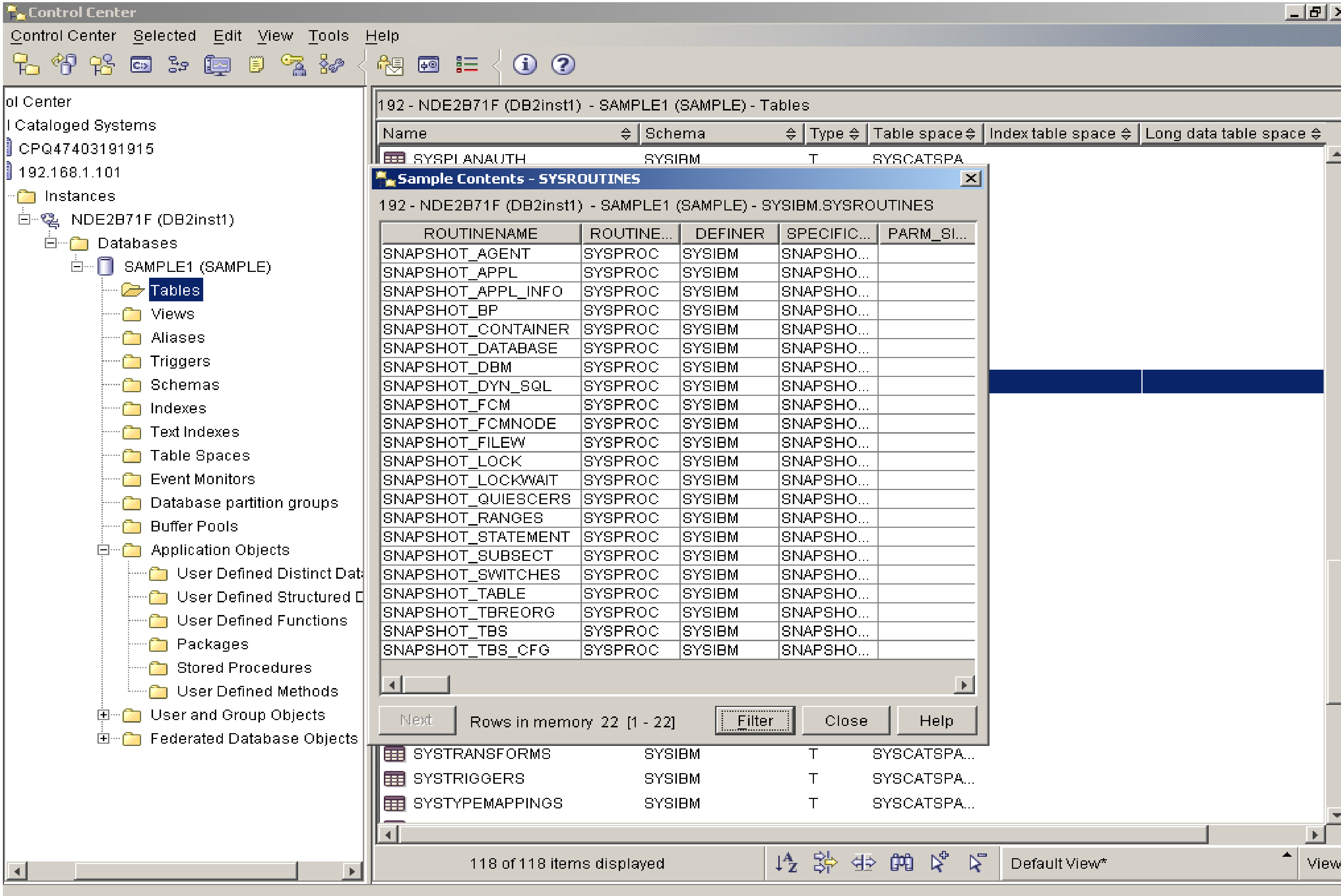
- Sample insert statement for using an SQL snapshot function and inserting the output into a DB2 table:

```
INSERT INTO BP_SNAP
SELECT * FROM TABLE( SNAPSHOT_BP( ‘sample', -1 )) as SNAPSHOT_BP;
```

SQL Snapshot Functions



SQL Snapshot Functions



SQL Snapshot Functions

Snapshot Monitoring Summary

- Snapshot monitoring shows us performance data at a point-in-time. With snapshots, we may or may not capture the information of interest depending on the time we take the snapshot, and whether or not the event we are trying to capture is running or has completed and the frequency of the snapshot taken.
- Many times snapshot data gives you enough data to identify a suspected problem and an event monitor is required to capture the complete picture. Hence event monitors should be used when snapshot data is inconclusive and further data capture is required.

Snapshot Monitoring Summary

- As demonstrated, Tablespace and Bufferpool activity is better analyzed using snapshots which provide adequate data to properly monitor and tune bufferpools and tablespace activity. However, for other data such as SQL Statements and connection and application activity, event monitoring is requried in addition to snapshot monitoring

Snapshot Monitoring Workshop

- Refer to your student workbook and examples for the Snapshot Monitoring Workshop. You have 45 minutes to complete the workshop.

Module 3
Event Monitoring

Objectives

- Upon completion of this module you should be able to:
  - Understand DB2 Event Monitoring and the different types of event monitors
  - Be able to create, run and stop event monitors
  - Be able to format and interpret event monitor output
- Be able to discern when to use snapshot monitoring, event monitoring, and the correct type of event monitor

Event Monitoring

- Event monitoring is used in DB2 to capture complete data regarding DB2 events. Unlike snapshots, which are capturing data at a point-in-time, event monitors are like a trace and record data for all event types at event completion. This has some implications that you need to be aware of such as:
  - More overhead than snapshot monitoring
  - Data captured to files or pipes is not in a viewable format
    - Has to be formatted using db2eva utility
    - Write to table data is viewable and requires no formatting besides your user queries
- Event monitoring is needed in many cases to accurately identify performance problems

Event Monitoring

- Create Event Monitor Options
- Options can be specified on the CREATE EVENT MONITOR statement to control the amount of information collected, to determine how the event monitor stops and starts, to specify the location and size of output files or pipes, types and size of buffers, and scope of the event monitor (local or global).

Event Monitoring

- Authorization Required
  - SYSADM or DBADM

Event Monitoring

- Event monitors can be created that write to three possible types of destinations
  - Write to pipe (requires a process to be reading the pipe) but low overhead
  - Write to file(s) – number and size of files specified on create event monitor command
  - Write to table – powerful and flexibile but these tables are logged just like any other DB2 persistent table

Event Monitoring

- Write to table – powerful and flexible but these tables are logged just like any other DB2 persistent table!
- However, write-to-table event monitors much easier to use than other types

Write-to-table Event Monitors

- Can use to support development efforts and once you have developed “canned” SQL to query the tables, it is very easy to review and asses performance and adequacy of SQL and resources
- Third party vendor query tools also can be used to speed analysis of data

Event Monitoring

- The following command can be used to create a connection event monitor that uses default values and writes-to tables:

```
CREATE EVENT MONITOR PGUNN
        FOR CONNECTIONS
    WRITE TO TABLE;
```

Event Monitoring

- Event monitors capture and record data as events complete. DB2 provides eight types of event monitors. They are:
    - ❑Database
    - ❑Tables
    - ❑Deadlocks
    - ❑Tablespaces
    - ❑Bufferpools
    - ❑Connections
    - ❑Statements
    - ❑Transactions

Event Monitoring

- **Tablespace Events**
  - Records an event record for each active tablespace when the last application disconnects from the database.
- **Bufferpools Events**
  - Bufferpool event monitor record an event record for bufferpools when the last application disconnects from the database.

Event Monitoring

- **Connection Events**
  - Connection event monitors record an event record for each database connection event when an application disconnects for the database.
- **Statement Events**
  - Statement event monitors record an event for every SQL statement issued by an application (for both dynamic and static SQL).
- **Transactions Events**
  - Transaction event monitors record an event record for every transaction when it completes (indicated by a COMMIT or ROLLBACK statement).

SQL Events

- 25-33% of tuning lies in database, tablespace, and bufferpool tuning
- 67-75% or more tuning opportunities lie within individual SQL statements
- Reduce the cost of most expensive SQL, and entire application runs faster.

Statement Event Monitor

- create event monitor STMNTS
  for statements
  write to file '/home/db2inst1/event/out'
  maxfiles 5 maxfilesize 1000 blocked append
  manualstart;
- set event monitor STMNTS state = 1;

Statement Events

- set event monitor STMNTS state = 0;
- Now that we have captured SQL Event Data:
  - Use DB2EVMON to format it
  - Use Control Center tool to look at it
  - Third party vendor tool

db2evmon

- Format the data as follows:
- db2evmon –path '/home/db2inst1/event/out' > stmtrpt.txt

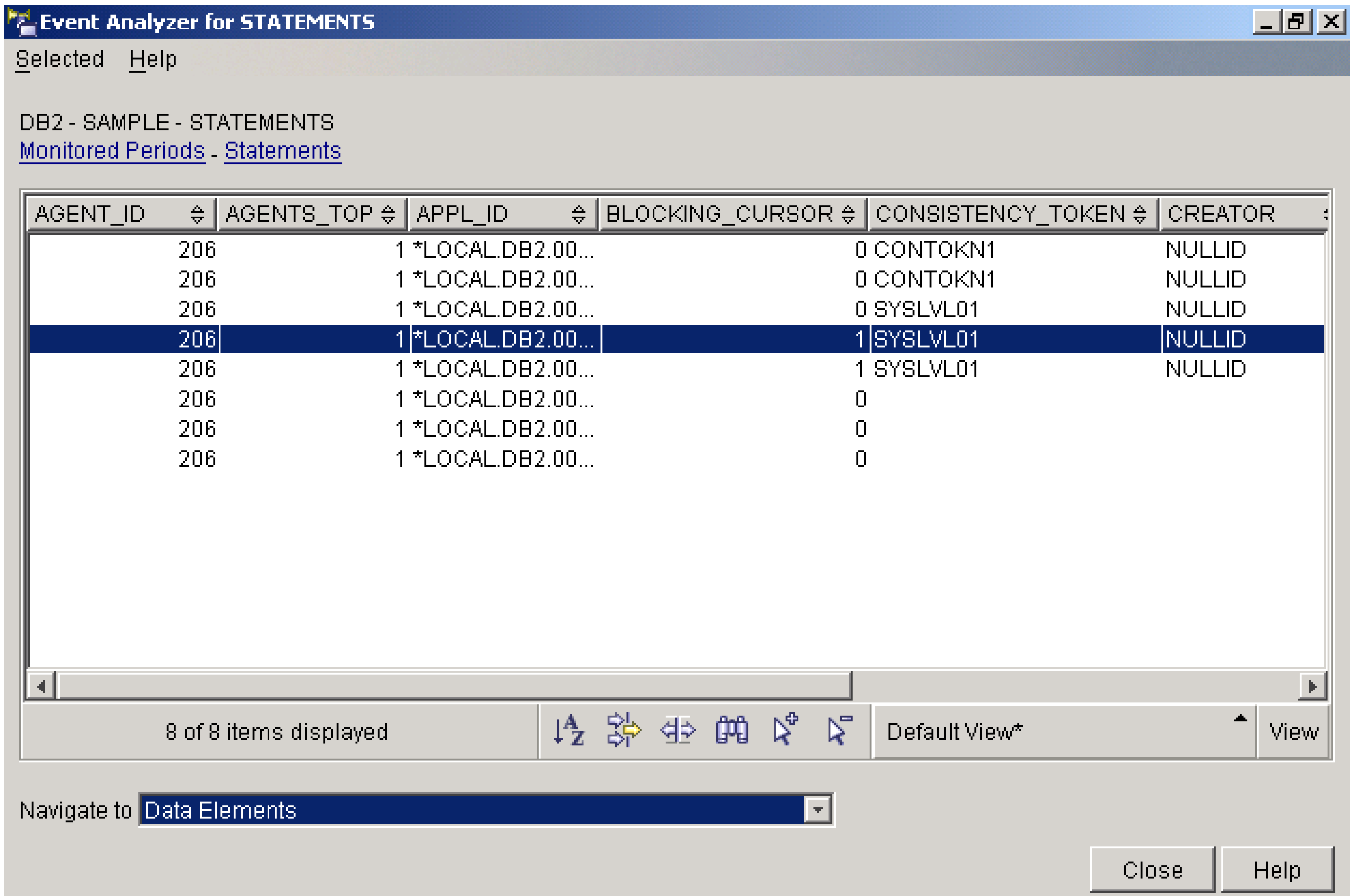
- or -

- db2evmon –db SAMPLE –evm STMNTS > stmtrpt.txt
- edit/view/more stmtrpt.txt
- db2eva

db2evmon output

```
3541) Statement Event ...
  Appl Handle: 297
  Appl Id: A02CCD85.042B.991130164258
  Appl Seq number: 0001
  -------------------------------------------
  Type     : Dynamic
  Operation: Close
  Section  : 297
  Creator  : NULLID
  Package  : SQLL1B0N
  Cursor   : SQLCURCS297
  Text     : SELECT MERCHANT.MECNTRY FROM MCUSTINFO MCUSTINFO, MERCHANT MERCHANT, SHOPPER SHOPPER
             WHERE SHOPPER.SHRFNBR = MCUSTINFO.MCSHNBR and MERCHANT.MERFNBR = MCUSTINFO.MCMENBR
             and SHOPPER.SHRFNBR = 111111
  -------------------------------------------
  Start Time: 11-30-1999 16:43:58.867373
  Stop Time:  11-30-1999 16:43:58.875953
  Number of Agents created: 1
  User CPU:   0.000000 seconds
  System CPU: 0.010000 seconds
  Fetch Count: 1
  Sorts: 2
  Total sort time: 0
  Sort overflows: 0
  Rows read: 2
  Rows written: 0
  Internal rows deleted: 0
  Internal rows updated: 0
  Internal rows inserted: 0
```

Formatting Event Monitor Output



Statement Event Monitor

- The statement event output will show a great deal of cost information including CPU, Sort, Fetches, Rows Read/Written, and internal rows DML. Elapsed time can be computed by finding the difference between Start and Stop times.
- Bufferpool detail (logical, physical, async reads/writes) is not available at statement event level. It is also not readily apparent which tablespaces are accessed.

Connection Event Monitors

- Connection events provide true cost of sort
- Use in conjunction with statement event monitors to identify suboptimal SQL and sort problems
- Use during periods of suspected activity to identify problems, then deactivate monitors
- Can be run more often but volume of data produced and overhead can be prohibitive

Deadlocks Event Monitor With Details

- New Deadlock event monitor with details
  - Activated by default at DB2 installation
  - DB2DETAILDEADLOCK
- Deactivate this right away as it imposes significant overhead
- Use only as needed
- New “with details” option provides finer granularity for offending SQL, object, and lock details

Write-to-table Event Monitors

- When a write-to-table event monitor is created, by default the following thirteen event monitor tables are created:
    - CONNHEADER
    - DEADLOCK
    - DLCONN
    - CONTROL
    - DLLOCK
    - STMT
    - SUBSECTION
    - XACT
    - CONN
    - DB
    - BUFFERPOOL
    - TABLESPACE
    - TABLE

Write-to-table Event Monitors

Event type	**Target table names**	**Available information**
DEADLOCKS	CONNHEADER DEADLOCK DLCONN CONTROL	Connection metadata Deadlock data Applications and locks involved in deadlock Event monitor metadata
DEADLOCKS WITH DETAILS	CONNHEADERE DEADLOCK DLCONN DLLOCK CONTROL	Connection metadata Deadlock data Applications involved in deadlock Locks involved in deadlock Event monitor metadata
STATEMENTS	CONNHEADER STMT SUBSECTION CONTROL	Connection metadata Statement data Statement data specific to subsection Event monitor metadata
TRANSACTIONS	CONNHEADER XACT CONTROL	Connection metadata Transaction data Event monitor metadata
CONNECTIONS	CONNHEADER CONN CONTROL	Connection metadata Connection data Event monitor metadata
DATABASE	DB CONTROL	Database manager data Event monitor metadata
BUFFERPOOLS	BUFFERPOOL CONTROL	Buffer pool data Event monitor metadata
TABLESPACES	TABLESPACE CONTROL	Tablespace data Event monitor metadata
TABLES	TABLE CONTROL	Table data Event monitor metadata

Write-to-table Event Monitors

- General Consideration for Write-to-Table Event Monitors

- When the CREATE EVENT MONITOR statement is issued, all event monitor target tables are created. If the creation of a table fails for any reason, an error is passed back to the application program and the CREATE EVENT MONITOR statement fails.
- During CREATE EVENT MONITOR processing, if a table already exists, but is **not defined** for use by another event monitor, no table is created, and processing continues. A warning is passed back to the application program.

Write-to-table Event Monitors

- Can also use “filters” with event monitors

Event Monitoring

- The following command can be used to create a connection event monitor that uses default values and writes-to tables:

```
CREATE EVENT MONITOR PGUNN
          FOR CONNECTIONS
    WRITE TO TABLE;
```

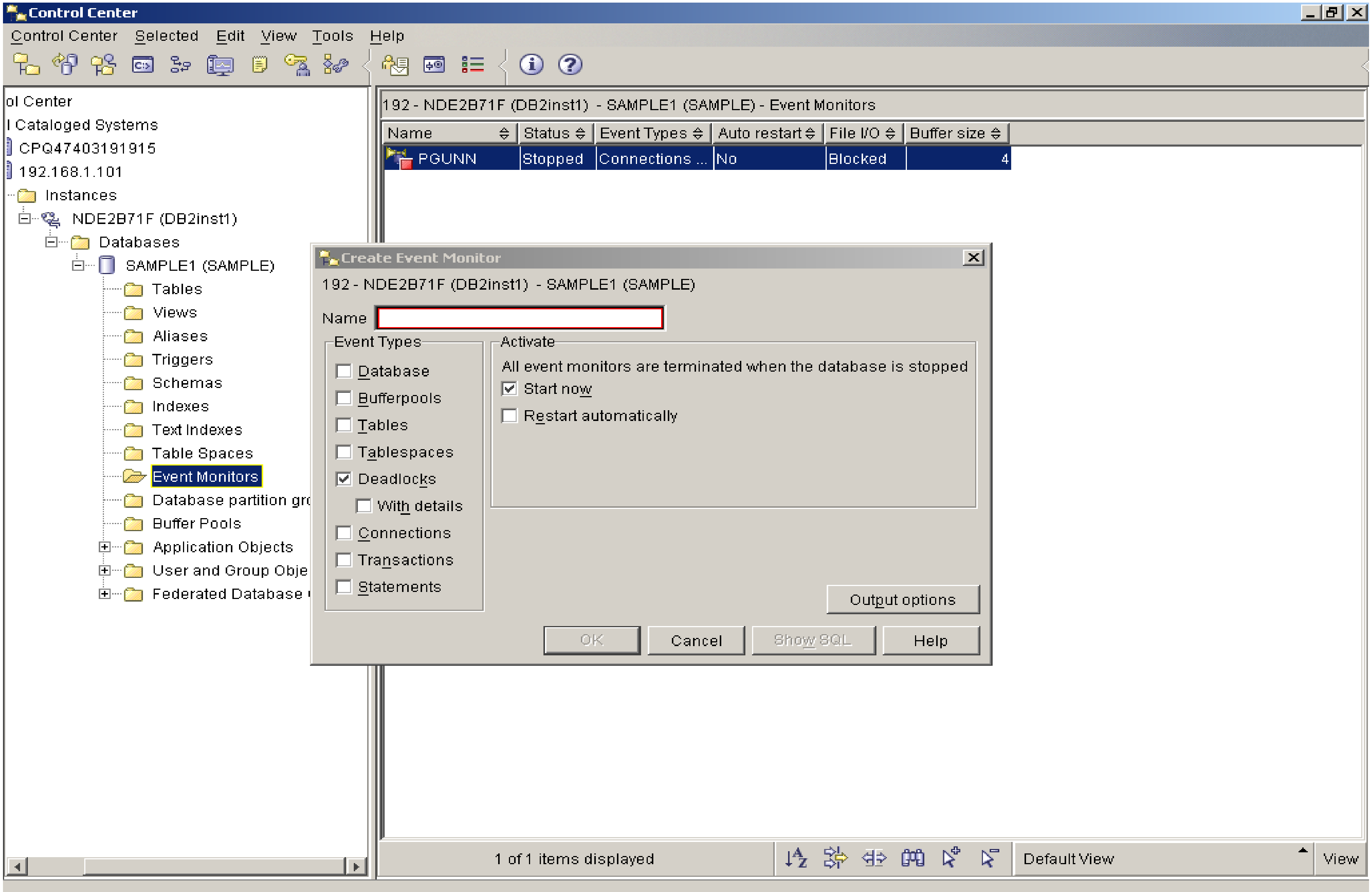
Creating Event Monitors

- create event monitor BADONE
  for statements
  write to file ‘/home/db2inst1/events/eventout’
  maxfiles 10 maxfilesize 1000 blocked append
  manualstart;
- Activate the event monitor by setting its state to 1
- set event monitor BADONE state = 1;
- Deactivate the event monitor by setting state to 0

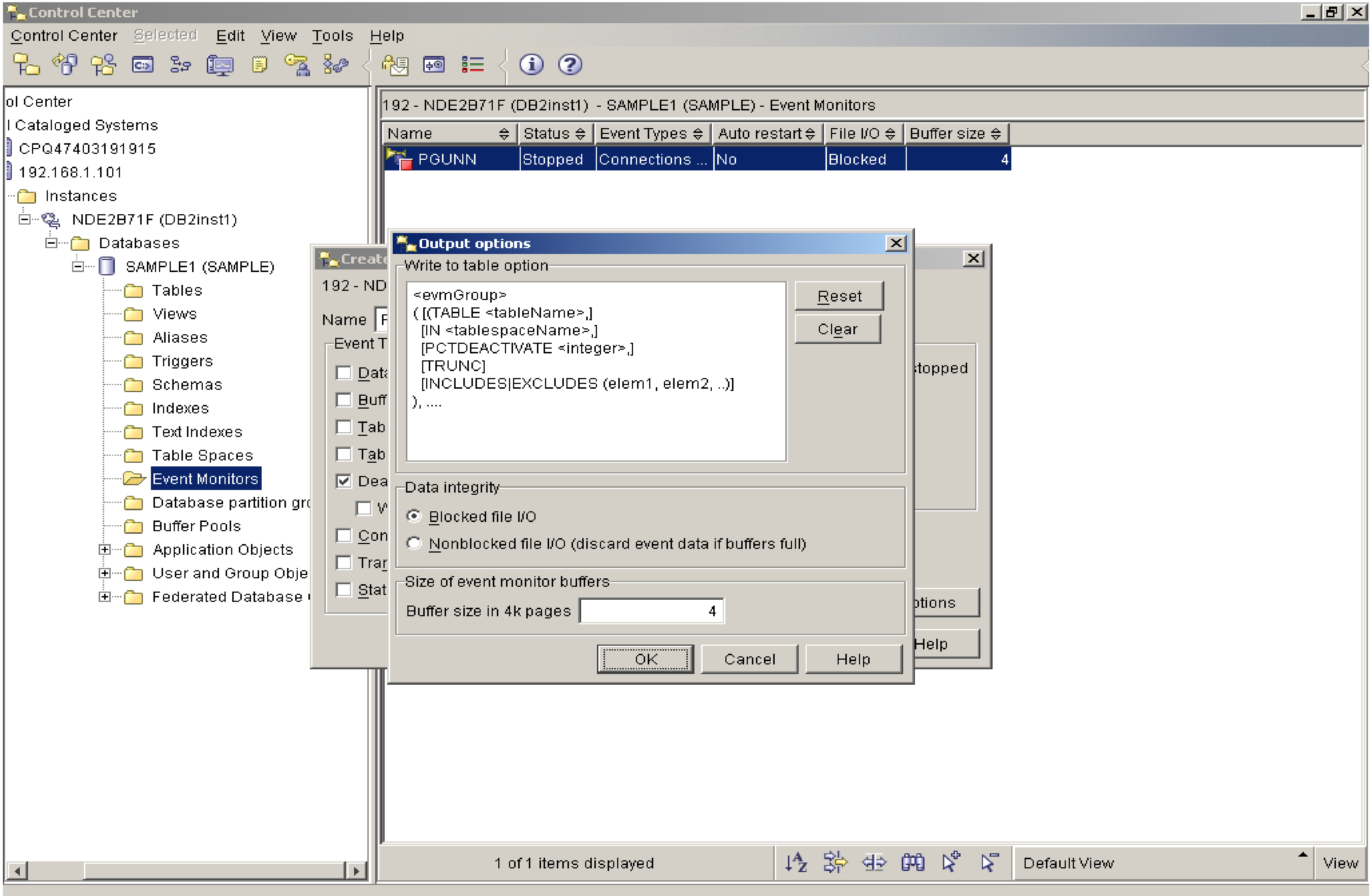
Write-to-table Event Monitors

- Can create via command line as previously outlined or via Control Center

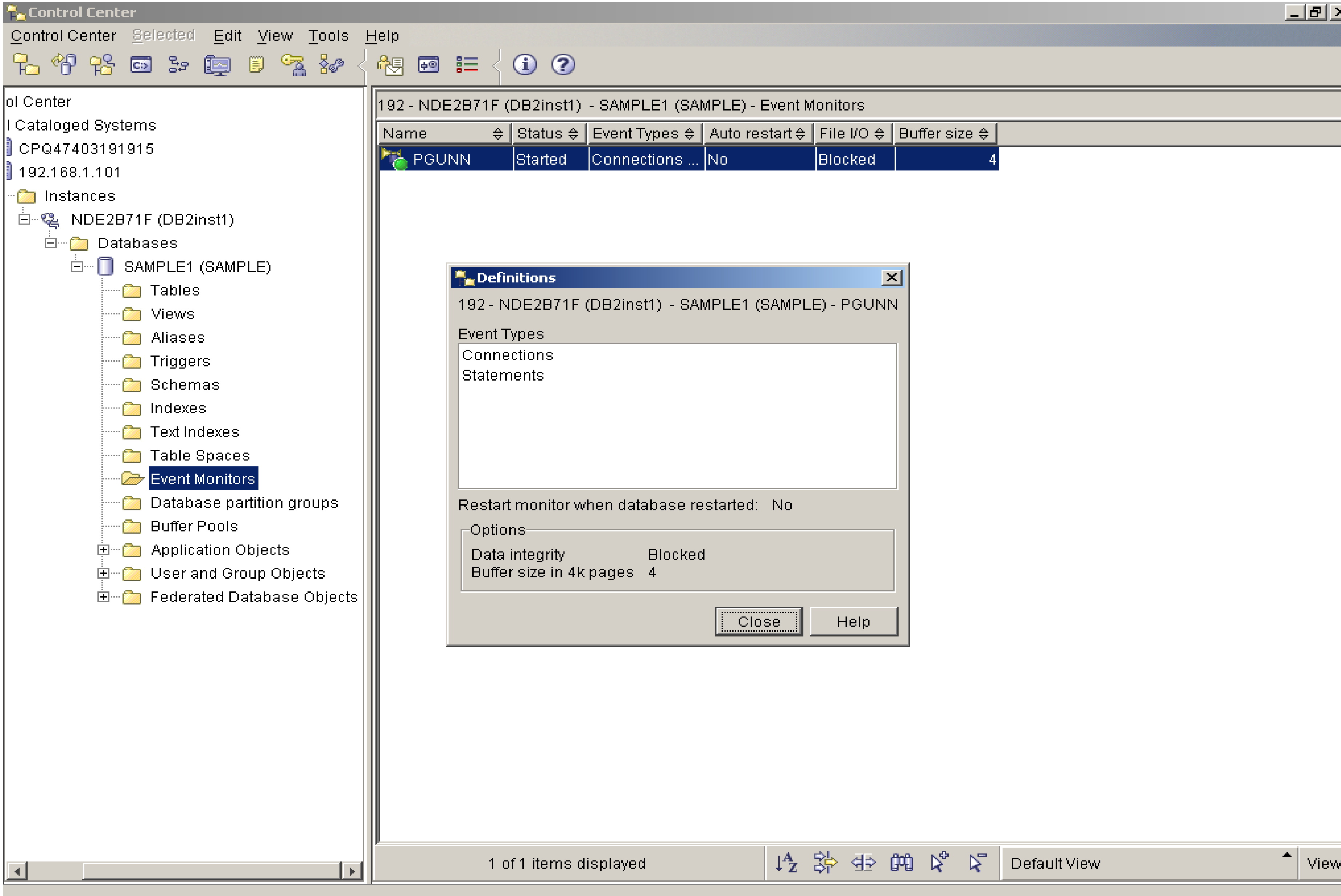
Write-to-table Event Monitors



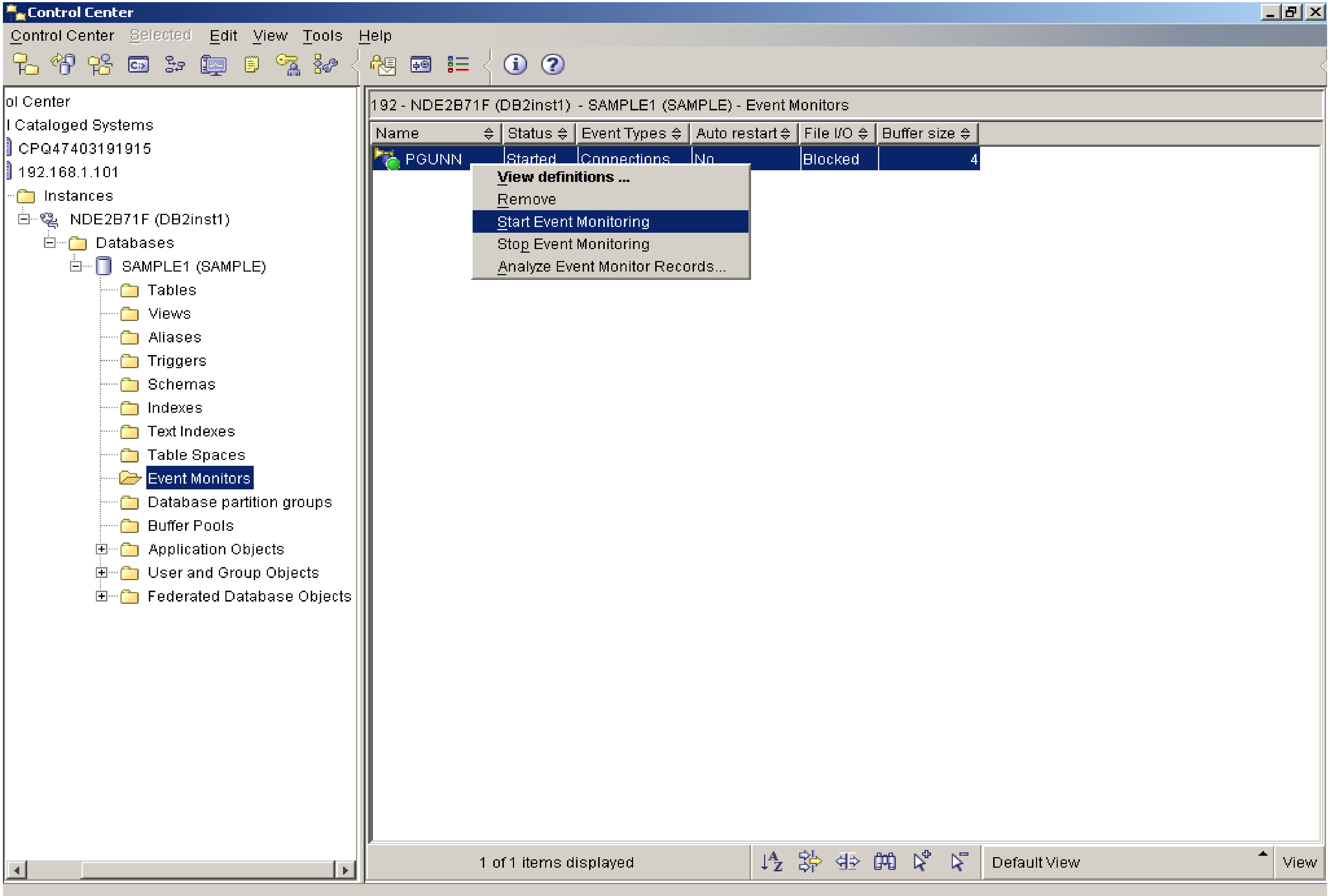
Write-to-table Event Monitors



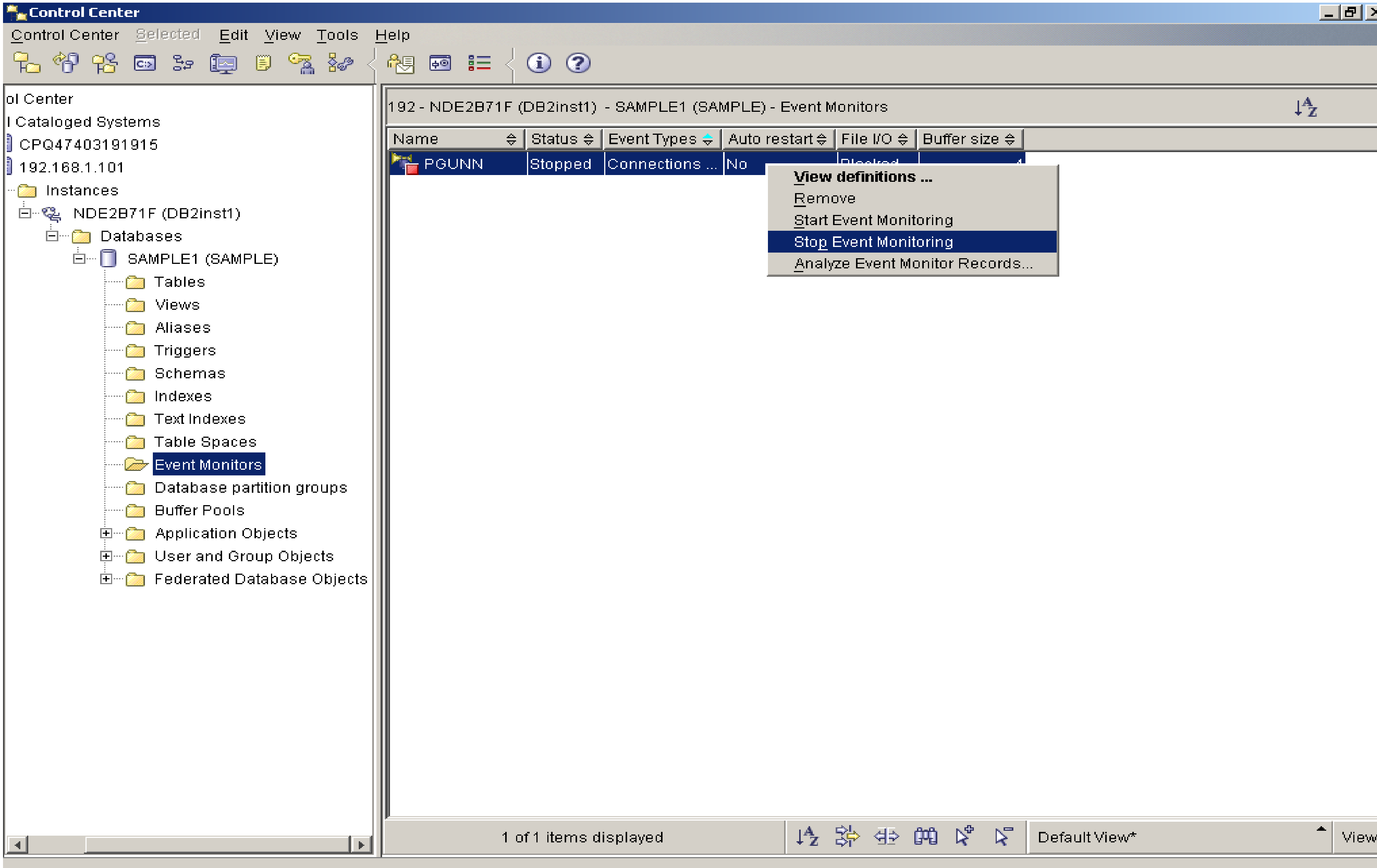
Write-to-table Event Monitors



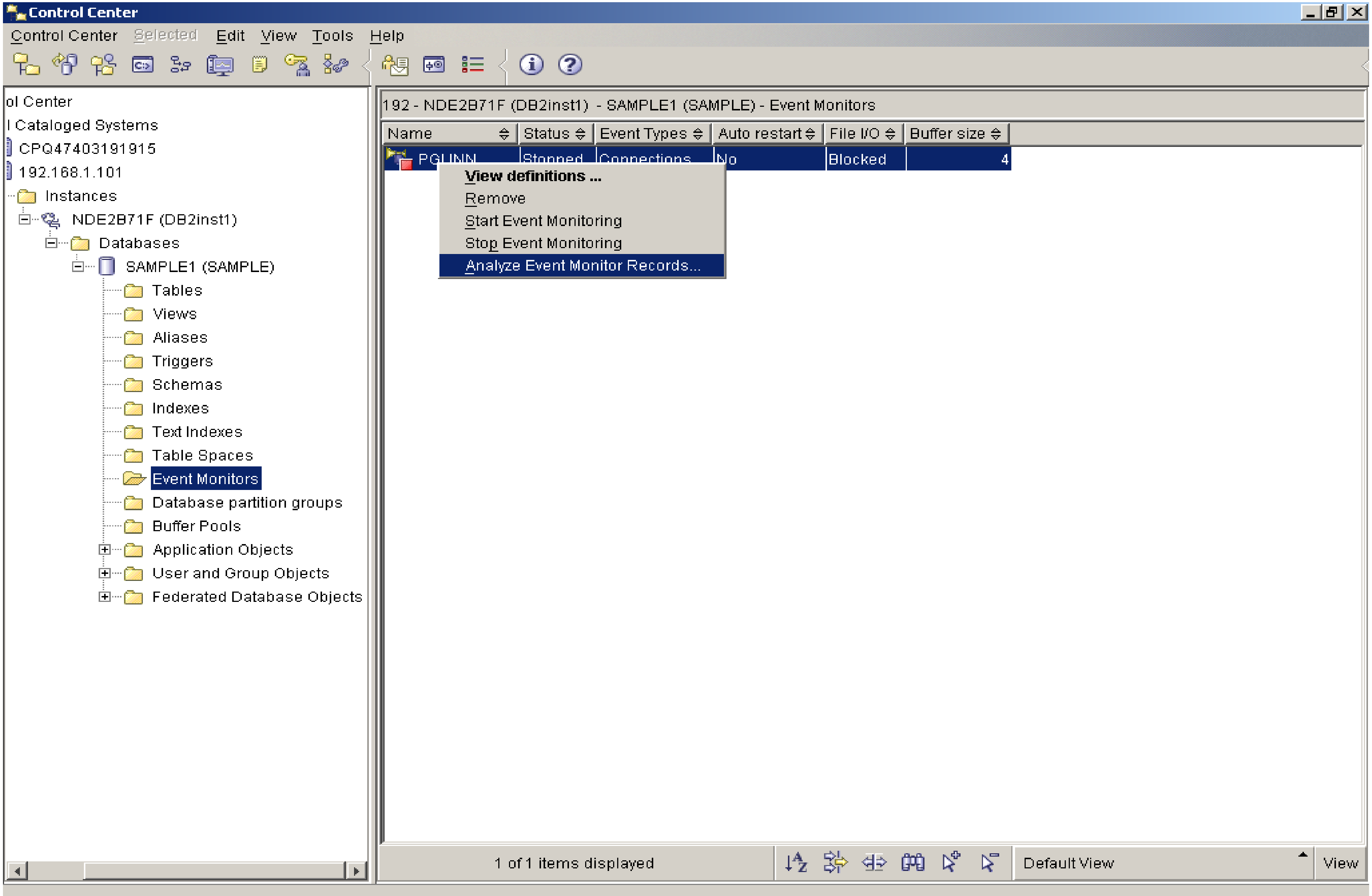
Write-to-table Event Monitors



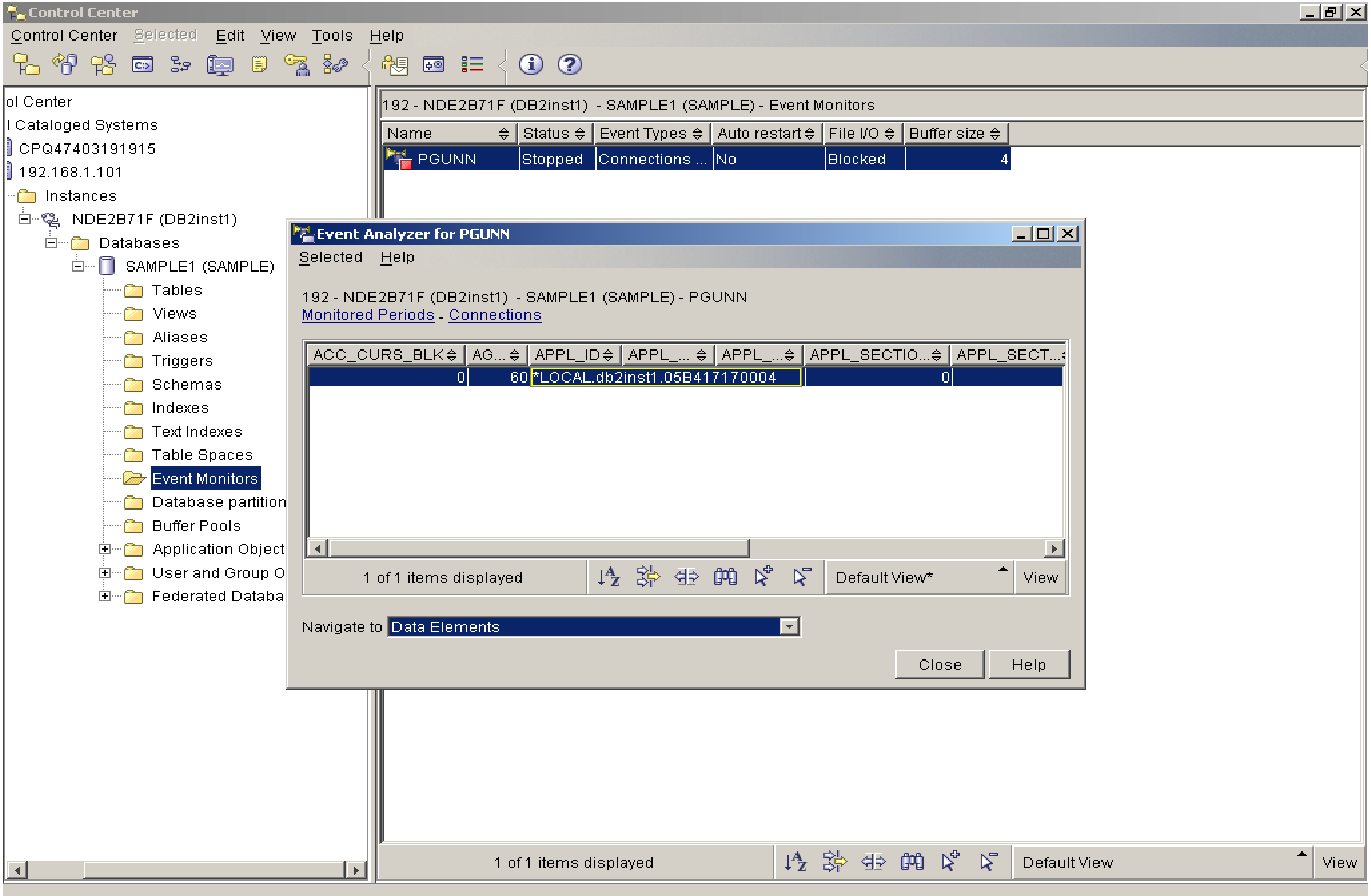
Write-to-table Event Monitors



Write-to-table Event Monitors



Write-to-table Event Monitors



Write-to-table Event Monitors

Data Elements

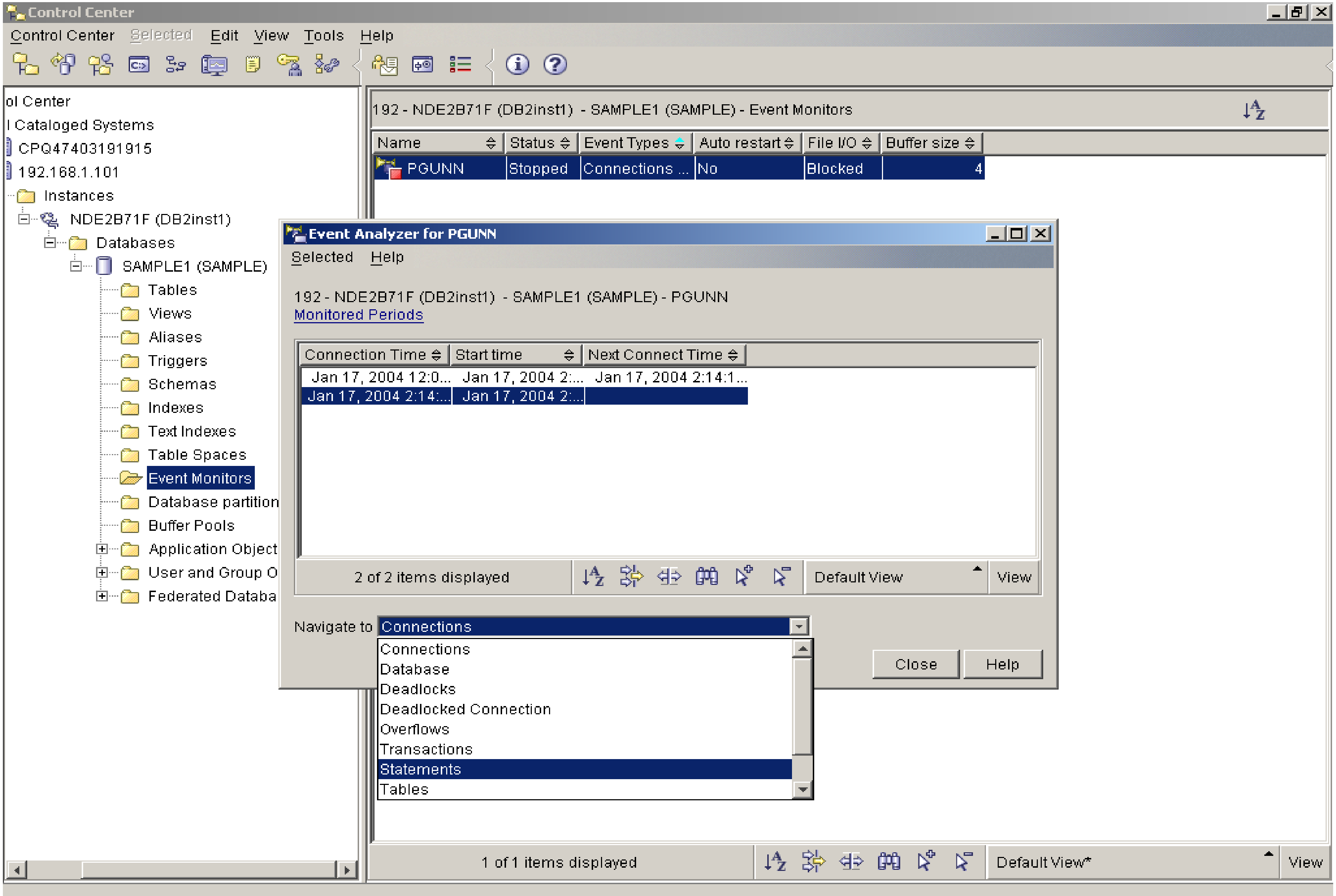
Data Element	Value
ACC_CURS_BLK	0
AGENT_ID	130
APPL_ID	C0A80168.L405...
APPL_PRIORITY	0
APPL_PRIORITY_TYPE	65536
APPL_SECTION_INSERTS	0
APPL_SECTION_LOOKUPS	2
APPL_STATUS	11
AUTHORITY_LVL	18292030
BINDS_PRECOMPILES	0
CAT_CACHE_HEAP_FULL	0
CAT_CACHE_INSERTS	0
CAT_CACHE_LOOKUPS	0
CAT_CACHE_OVERFLOWS	0
CAT_CACHE_SIZE_TOP	0
COMMIT_SQL_STMTS	2
COORD_NODE	130
DDL_SQL_STMTS	0
DEADLOCKS	0
DIRECT_READ_REQS	0
DIRECT_READ_TIME	0
DIRECT_READS	0
DIRECT_WRITE_REQS	0
DIRECT_WRITE_TIME	0
DIRECT_WRITES	0
DISCONN_TIME	Jan 17, 2004 2:...
DYNAMIC_SQL_STMTS	2
ELAPSED_EXEC_TIME	123
EVMON_FLUSHES	0
FAILED_SQL_STMTS	0
HASH_JOIN_OVERFLOWS	0
HASH_JOIN_SMALL_OVERFLOWS	0
INT_AUTO_REBINDS	0
INT_COMMITS	0
INT_DEADLOCK_ROLLBACKS	0
INT_ROLLBACKS	0
INT_ROWS_DELETED	0
INT_ROWS_INSERTED	0
INT_ROWS_UPDATED	0
LOCK_ESCALS	0
LOCK_TIMEOUTS	0
LOCK_WAIT_TIME	0
LOCK_WAITS	0
PARTIAL_RECORD	0

Write-to-table Event Monitors

Data Elements

Data Element	Value
PKG_CACHE_INSERTS	0
PKG_CACHE_LOOKUPS	1
POOL_DATA_FROM_ESTORE	0
POOL_DATA_L_READS	1
POOL_DATA_P_READS	0
POOL_DATA_TO_ESTORE	0
POOL_DATA_WRITES	0
POOL_INDEX_FROM_ESTORE	0
POOL_INDEX_L_READS	1
POOL_INDEX_P_READS	0
POOL_INDEX_TO_ESTORE	0
POOL_INDEX_WRITES	0
POOL_READ_TIME	0
POOL_WRITE_TIME	0
PREFETCH_WAIT_TIME	0
PRIV_WORKSPACE_NUM_OVERF...	0
PRIV_WORKSPACE_SECTION_IN...	1
PRIV_WORKSPACE_SECTION_LO...	5
PRIV_WORKSPACE_SIZE_TOP	37440
REJ_CURS_BLK	0
ROLLBACK_SQL_STMTS	1
ROWS_DELETED	0
ROWS_INSERTED	0
ROWS_READ	1
ROWS_SELECTED	1
ROWS_UPDATED	0
ROWS_WRITTEN	0
SELECT_SQL_STMTS	1
SEQUENCE_NO	0005
SHR_WORKSPACE_NUM_OVERF...	0
SHR_WORKSPACE_SECTION_IN...	0
SHR_WORKSPACE_SECTION_LO...	0
SHR_WORKSPACE_SIZE_TOP	0
SORT_OVERFLOWS	0
STATIC_SQL_STMTS	3
SYSTEM_CPU_TIME	0
TOTAL_HASH_JOINS	0
TOTAL_HASH_LOOPS	0
TOTAL_SORT_TIME	0
TOTAL_SORTS	0
UID_SQL_STMTS	0
UNREAD_PREFETCH_PAGES	0
USER_CPU_TIME	20000
X_LOCK_ESCALS	0

Write-to-table Event Monitors



Write-to-table Event Monitors

Event Analyzer for PGUNN

Selected Help

192 - NDE2B71F (DB2inst1) - SAMPLE1 (SAMPLE) - PGUNN

Monitored Periods - Statements

AGENT_ID	AGENTS_TOP	APPL_ID	BLOCKING_CURSOR	CONSISTENCY_TOKEN	CREATOR	CURSOR_NAME	EVMON_FLUSHES	FETC
60	1	*LOCAL.db2inst...	0				0	
60	1	*LOCAL.db2inst...	0				0	
60	1	*LOCAL.db2inst...	0				0	
130	1	C0A80168.L405...	0				0	
135	1	*LOCAL.db2inst...	0				0	
135	1	*LOCAL.db2inst...	0				0	
130	1	C0A80168.L405...	0	SYSLVL01	NULLID	SQL_CURSH200...	0	
130	1	C0A80168.L405...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	
130	1	C0A80168.L405...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	
130	1	C0A80168.L405...	0				0	
130	1	C0A80168.L405...	0				0	
130	1	C0A80168.L405...	0				0	
134	1	C0A80168.LF05...	0	CONTOKN1	NULLID		0	
134	1	C0A80168.LF05...	0	CONTOKN1	NULLID		0	
134	1	C0A80168.LF05...	0	SYSLVL01	NULLID	SQL_CURSH200...	0	
134	1	C0A80168.LF05...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	
134	1	C0A80168.LF05...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	
134	1	C0A80168.LF05...	0				0	
134	1	C0A80168.LF05...	0				0	
134	1	C0A80168.LF05...	0				0	
123	1	*LOCAL.db2inst...	0	CONTOKN1	NULLID		0	
123	1	*LOCAL.db2inst...	0	AAAAAJHR	NULLID		0	
123	1	*LOCAL.db2inst...	0				0	
123	1	*LOCAL.db2inst...	0	AAAAAJHR	NULLID		0	
123	1	*LOCAL.db2inst...	0				0	
123	1	*LOCAL.db2inst...	0	AAAAAJHR	NULLID	SQLCUR201	0	
123	1	*LOCAL.db2inst...	1	AAAAAJHR	NULLID	SQLCUR201	0	
123	1	*LOCAL.db2inst...	1	AAAAAJHR	NULLID	SQLCUR201	0	
123	1	*LOCAL.db2inst...	1	AAAAAJHR	NULLID	SQLCUR201	0	
123	1	*LOCAL.db2inst...	0				0	
123	1	*LOCAL.db2inst...	0	AAAAAJHR	NULLID		0	
123	1	*LOCAL.db2inst...	0				0	

34 of 34 items displayed

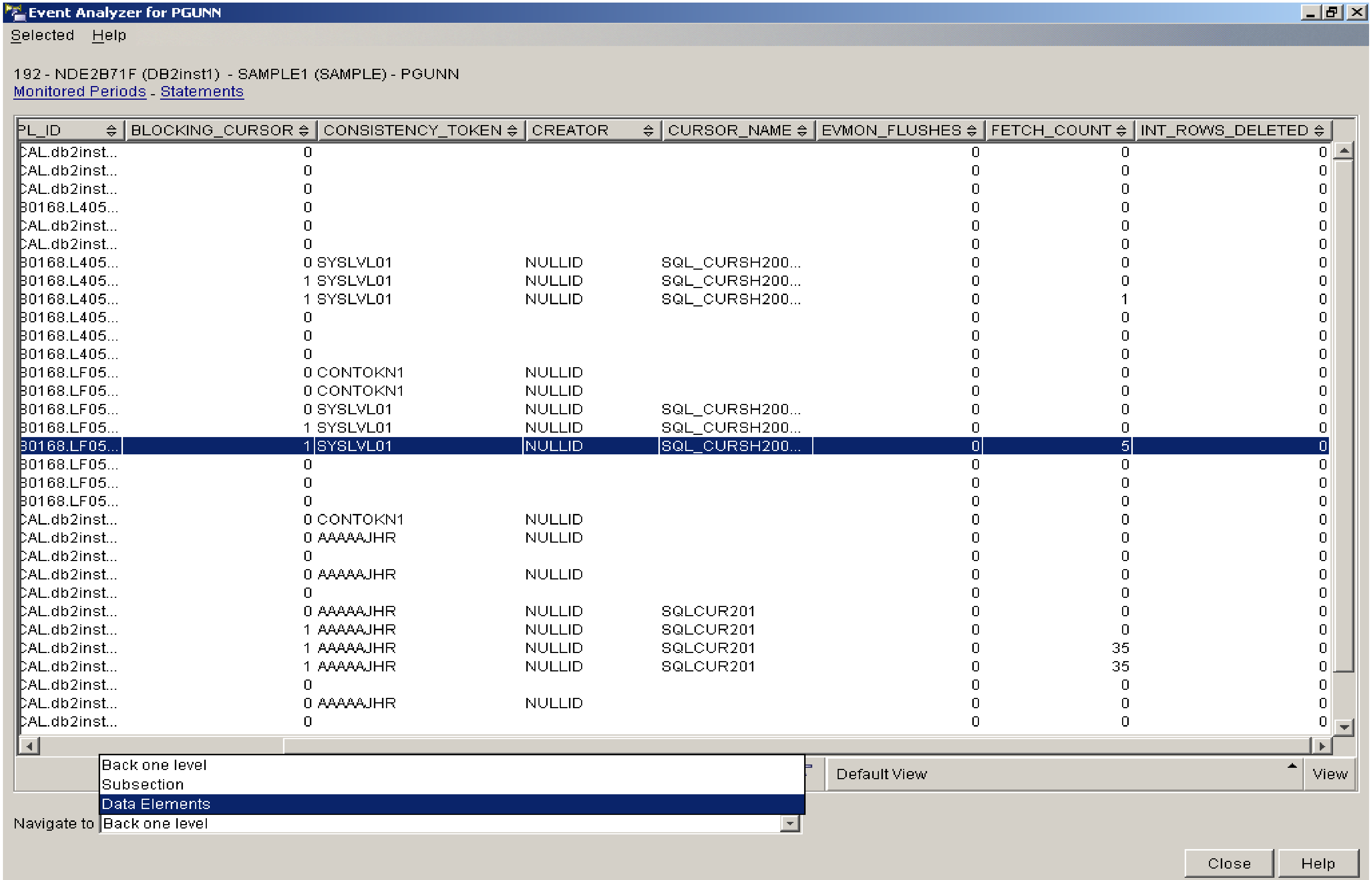
Default View

View

Navigate to Back one level

Close Help

Write-to-table Event Monitors



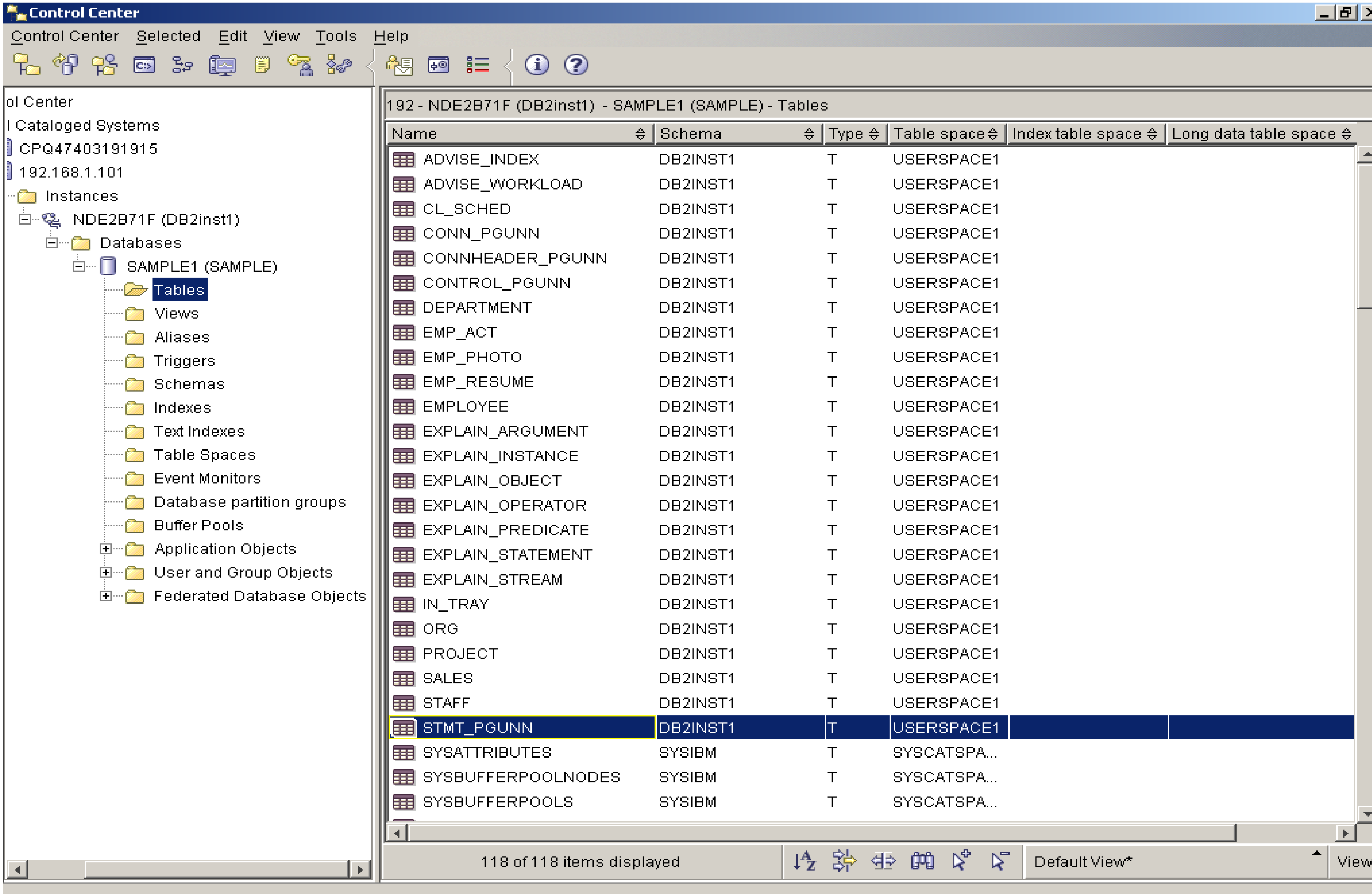
PL_ID	BLOCKING_CURSOR	CONSISTENCY_TOKEN	CREATOR	CURSOR_NAME	EVMON_FLUSHES	FETCH_COUNT	INT_ROWS_DELETED
CAL.db2inst...	0				0	0	0
CAL.db2inst...	0				0	0	0
CAL.db2inst...	0				0	0	0
30168.L405...	0				0	0	0
CAL.db2inst...	0				0	0	0
CAL.db2inst...	0				0	0	0
30168.L405...	0	SYSLVL01	NULLID	SQL_CURSH200...	0	0	0
30168.L405...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	0	0
30168.L405...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	1	0
30168.L405...	0				0	0	0
30168.L405...	0				0	0	0
30168.L405...	0				0	0	0
30168.LF05...	0	CONTOKN1	NULLID		0	0	0
30168.LF05...	0	CONTOKN1	NULLID		0	0	0
30168.LF05...	0	SYSLVL01	NULLID	SQL_CURSH200...	0	0	0
30168.LF05...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	0	0
30168.LF05...	1	SYSLVL01	NULLID	SQL_CURSH200...	0	5	0
30168.LF05...	0				0	0	0
30168.LF05...	0				0	0	0
30168.LF05...	0				0	0	0
CAL.db2inst...	0	CONTOKN1	NULLID		0	0	0
CAL.db2inst...	0	AAAAAJHR	NULLID		0	0	0
CAL.db2inst...	0				0	0	0
CAL.db2inst...	0	AAAAAJHR	NULLID		0	0	0
CAL.db2inst...	0				0	0	0
CAL.db2inst...	0	AAAAAJHR	NULLID	SQLCUR201	0	0	0
CAL.db2inst...	1	AAAAAJHR	NULLID	SQLCUR201	0	0	0
CAL.db2inst...	1	AAAAAJHR	NULLID	SQLCUR201	0	35	0
CAL.db2inst...	1	AAAAAJHR	NULLID	SQLCUR201	0	35	0
CAL.db2inst...	0				0	0	0
CAL.db2inst...	0	AAAAAJHR	NULLID		0	0	0
CAL.db2inst...	0				0	0	0

Write-to-table Event Monitors

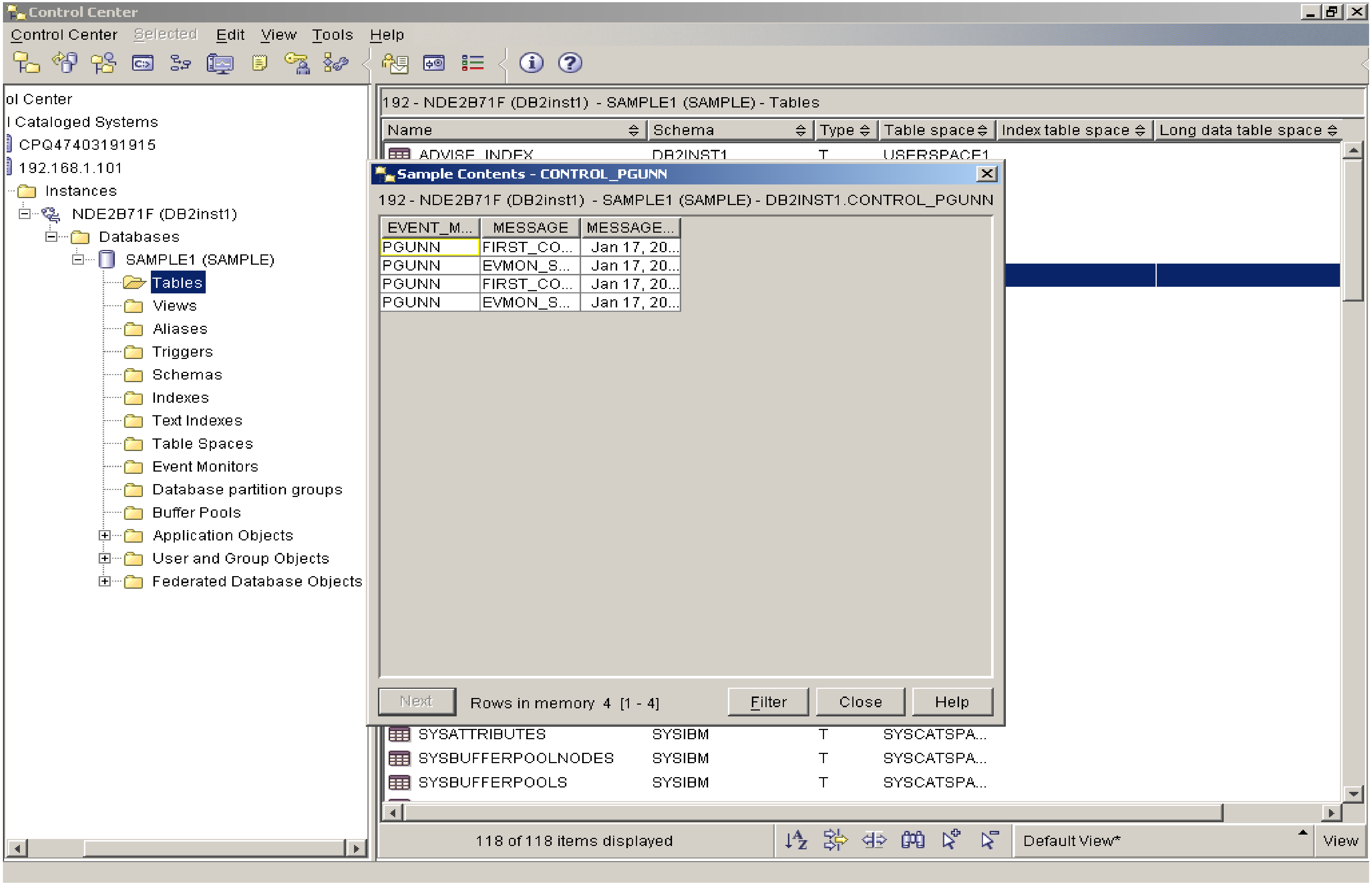
Data Elements

Data Element	Value
AGENT_ID	123
AGENTS_TOP	1
APPL_ID	*LOCAL.db2inst...
BLOCKING_CURSOR	1
CONSISTENCY_TOKEN	AAAAAJHR
CREATOR	NULLID
CURSOR_NAME	SQLCUR201
EVMON_FLUSHES	0
FETCH_COUNT	35
INT_ROWS_DELETED	0
INT_ROWS_INSERTED	0
INT_ROWS_UPDATED	0
PACKAGE_NAME	SQLC2E03
PACKAGE_VERSION_ID	...
PARTIAL_RECORD	0
ROWS_READ	35
ROWS_WRITTEN	0
SECTION_NUMBER	201
SEQUENCE_NO	0006
SORT_OVERFLOWS	0
SQL_REQ_ID	436
SQLCABC	136
SQLCAID	SQLCA
SQLCODE	0
SQLERRD1	0
SQLERRD2	0
SQLERRD3	0
SQLERRD4	0
SQLERRD5	0
SQLERRD6	0
SQLERRM	
SQLERRP	SQL08010
SQLSTATE	00000
SQLWARN	
START_TIME	Jan 17, 2004 2:...
STMT_OPERATION	6
STMT_TEXT	select * from staff
STMT_TYPE	2
STOP_TIME	Jan 17, 2004 2:...
SYSTEM_CPU_TIME	0
TOTAL_SORT_TIME	0
TOTAL_SORTS	0
USER_CPU_TIME	0

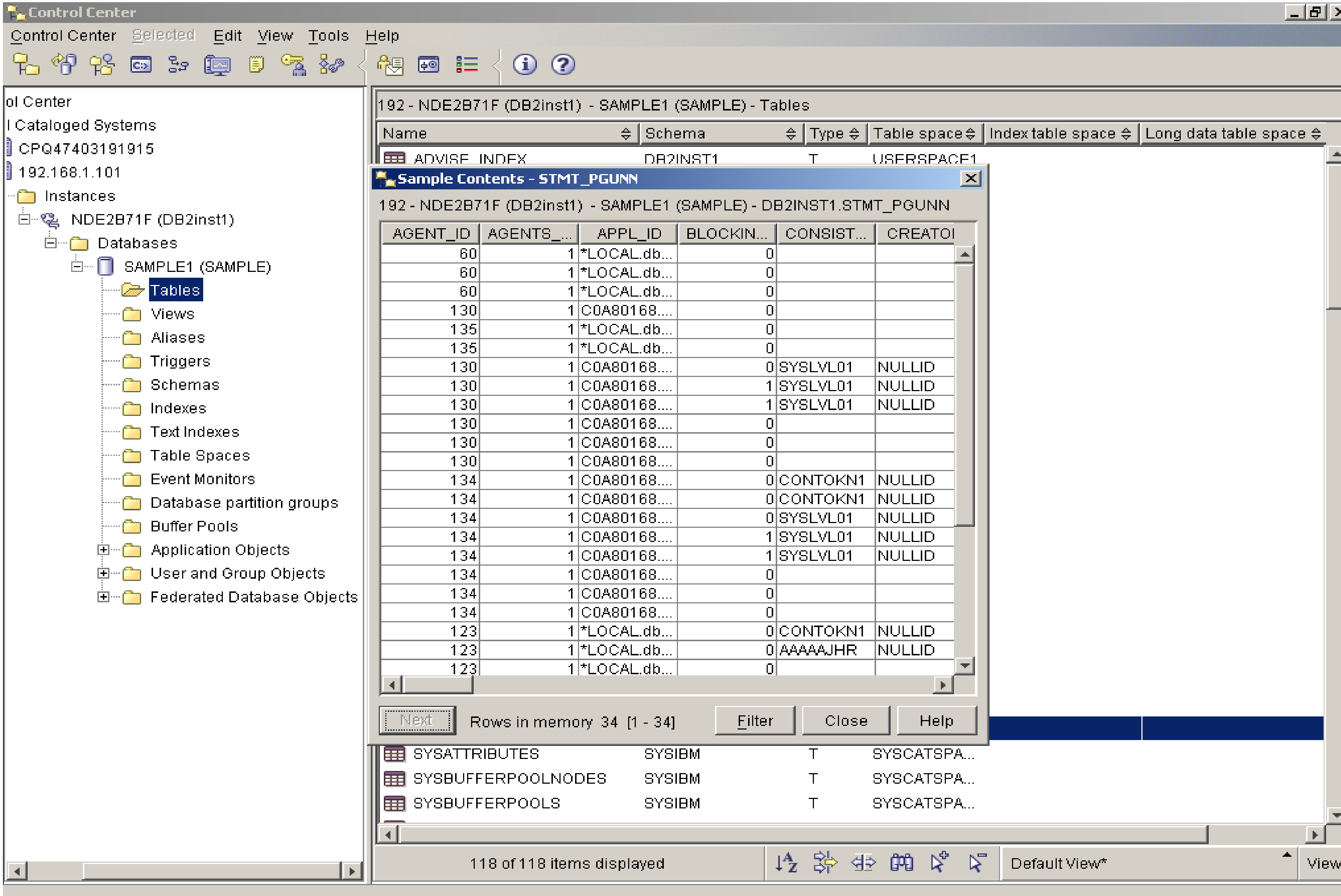
Write-to-table Event Monitors



Write-to-table Event Monitors



Write-to-table Event Monitors



Write-to-table Event Monitors

- This page intentionally left blank

Write-to-table Event Monitors

- This page intentionally left blank

Module 4
Tuning Sorts

Objectives

- Upon completion of this module you should be able to:
  - Identify and eliminate sorts
  - Understand sort terminology
  - Use event monitors to identify SQL statements causing sort problems
- Understand sort overflows and learn how to prevent them from occurring

Sorts

- We humans need data returned in some type of order
- Sorts occur to accomplish this when data is not in order, indexes are not available to return the data in order, or distinct is used to eliminate duplicates
- Max or Min functions not supported by index
- Sorts are usually a major problem in all environments

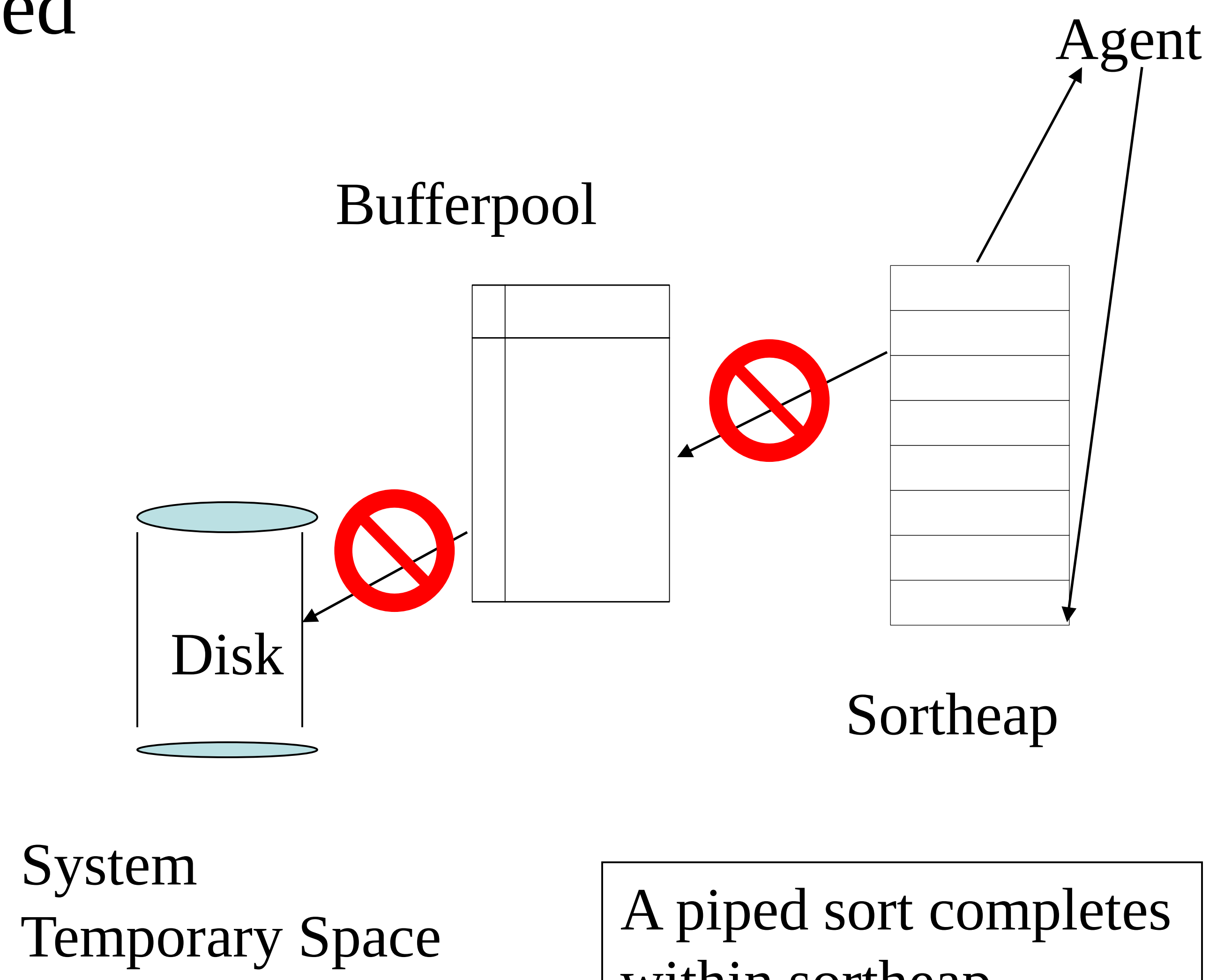
Key Sort Indicators

- SORT VOCABULARY
  - Sort Size
  - Sort Heap
  - Sort Overflows
  - Sort Heap Threshold
  - Sort Capacity
  - Post Threshold Sorts
  - Private Sort
  - Shared Sort
  - SHEAPTHRES
  - SHEAPTHRES_SHR

Configuration Settings that affect your SORT physical memory

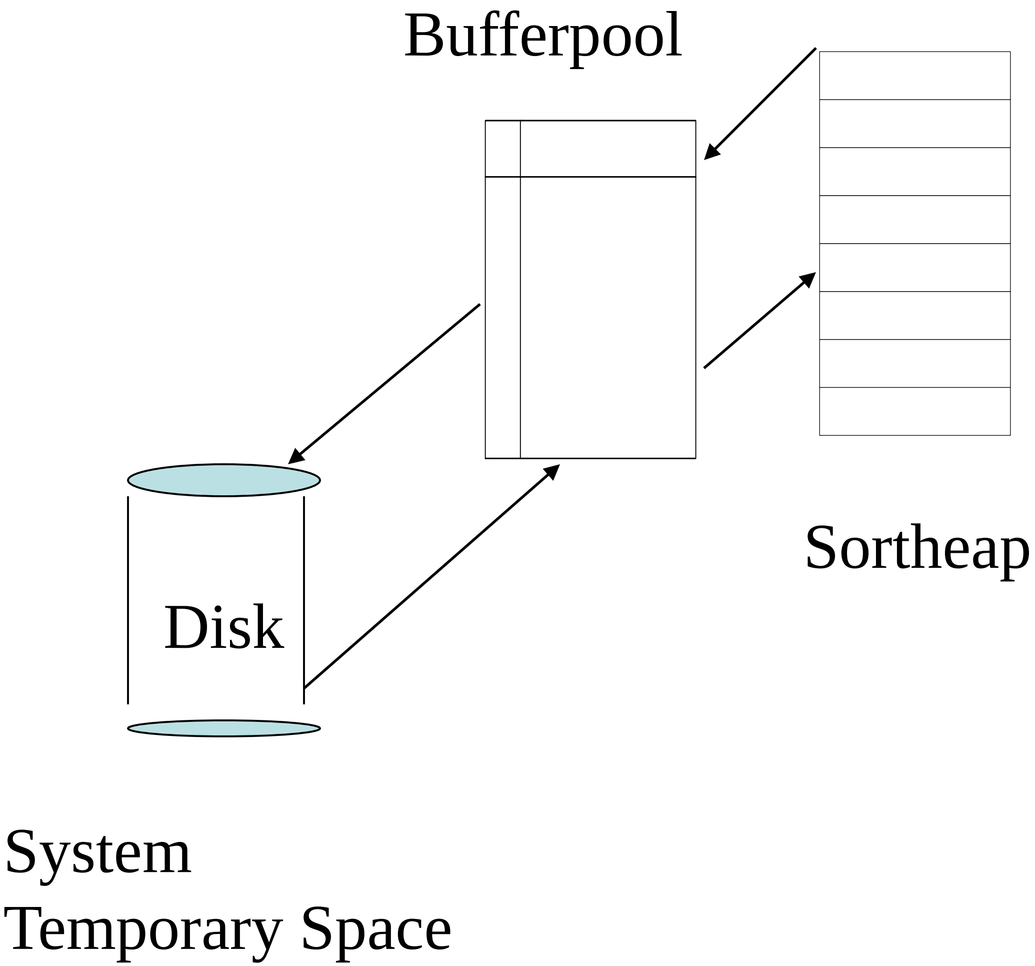
- INTRA_PARALLEL =
  - YES, Shared Memory
    - Decision Support Databases
    - Large Data Volume Queries
  - NO, Private Memory
    - OLTP/Web based/Mixed Applications (DEGREE = 1)
- DB2MEMDISCLAIM (AIX) = YES
- DB2MEMMAXFREE = 8192000
- SHEAPTHRES and SHEAPTHRES_SHR
- BUFFERPOOL SIZES

NonOverflowed Sort (Piped)



A piped sort completes within sortheap without overflowing to bufferpool and disk

Overflowed Sort



Sampling SORT Performance

- Sources of SORT Data
  - Database Snapshots
  - Application Snapshots
  - Connection Events
  - SQL Events
- DB2 “GET SNAPSHOT FOR DATABASE ON *DBNAME*”
- DB2 “GET SNAPSHOT FOR DATABASE MANAGER”

- Database Snapshot

```
Total sort heap allocated = 0
Total sorts               = 237
Total sort time (ms)      = 10190
Sort overflows            = 29
Active sorts              = 0

Number of hash joins      = 0
Number of hash loops      = 0
Number of hash join overflows = 0
Number of small hash join overflows
   = 0
```

- Database Manager Snapshot

```
Sort heap allocated       = 0
Post threshold sorts      = 0
Piped sorts requested     = 181
Piped sorts accepted      = 181
```

SORT Formulas

- % SHEAPTHRES Allocated = Sort Heap Allocated * 100 / SHEAPTHRES
  - If > 90%, larger SHEAPTHRES or smaller SORTHEAP
- If Post Threshold Sorts > 0 (rule above violated), same remedies
- Average Sort Time = Total Sort Time(ms)/Total Sorts
- % SORT Overflows = # Overflows * 100 / Total Sorts
  - If OLTP & If > 3%, cure sorts or increase SORTHEAP
- Sorts / TX, Sorts / SQL, Sort Time / TX, “TCA”
- High Water Sort Heap Used, Average Sort Heap Used

Measuring SORT Performance

- Connection Events provide **true cost** of Sort activity for an Application
- create event monitor GUNCONN for connections write to file 'e:\tmp\dbaudit\conn\' maxfiles 1 maxfilesize 1024 blocked replace manualstart;
- Set event monitor GUNCONN state = 1;
- Set event monitor GUNCONN state = 0;

- SQL Events provide **true cost** of Sort activity for an **individual statement**
- create event monitor GUNSQL for statements write to file 'e:\tmp\dbaudit\sql\' maxfiles 1 maxfilesize 2048 blocked replace manualstart;
- set event monitor GUNSQL state = 1;
- set event monitor GUNSQL state = 0;

- db2evmon -path e:\tmp\dbaudit\DIRNAME

Attendee Notes

- File -
  - Blocked - ensures no data loss at the expense of possible performance delays, should the buffer become full
  - Nonblocked - Fastest capture of event data, but event data could be lost if buffers become full
- Pipes
  - Memory address used to pass event data from buffers to application program reading pipe, extremely fast. “nonblocked” is only option
- Write-to-table event monitors new in V8.1

Sort Consumption

- dynexpln -g -d DBNAME -q “Select * from … order by …”

```
|  Insert Into Sorted Temp Table  ID = t1
|  |  #Columns = 31
|  |  #Sort Key Columns = 1
|  |  |  Key 1: NAME (Ascending)
|  |  Sortheap Allocation Parameters:
|  |  |  #Rows     = 1981
|  |  |  Row Width = 260
|  |  Piped
```

- ((260+8) x 1981) = 530,908 bytes / 4096 = 129.6 4K Pages = 530,908 / 1,048,576 = .51 MB …. 50% of default 256 4K SORTHEAP

Attendee Notes

- Sorts that cannot be completed within a SORTHEAP will overflow to TEMPSPACE via the bufferpool. Recognizing that some number of Overflow sorts will invariably occur, there are several tuning opportunities to optimize overflow behavior.
  - Dramatic sort performance improvements can be achieved by adhering to essential principles:
    - Container Placement
    - Number of Containers
    - Prefetch Sizes
    - TEMPSPACE Bufferpool

Design Advisor

db2advis -d peiprd -I wildsortsql.txt > wildsqlixadvout.txt

execution started at timestamp 2002-08-12-10.25.44.141157

found [1] SQL statements from the input file

Calculating initial cost (without recommmended indexes) [23866.660156] timerons

Initial set of proposed indexes is ready.

Found maximum set of [1] recommended indexes

Cost of workload with all indexes included [75.079346] timerons

total disk space needed for initial set [4.747] MB

total disk space constrained to   [-1.000] MB

1 indexes in current solution

[23866.6602] timerons (without indexes)

[75.0793] timerons (with current solution)

[%99.69] improvement

Trying variations of the solution set.--

-- execution finished at timestamp 2002-08-12-10.25.45.932376--

-- LIST OF RECOMMENDED INDEXES

-- ===========================

-- index[1],   4.747MB

CREATE INDEX WIZ1 ON "PUSER   "."T_FILE" ("FILE_STATUS_NUM" DESC) ;

-- ===========================--

Design Workload tool is finished.

Tuning Sort Summary

- Tuning sorts is a major part of your job as a DBA. Queries and reports require data to be in order to support business rules and objectives
- SQL requires sorts to be performed if indexes or clustering are not being used to return data in the order required
- Sorts that aren’t tuned can overflow to disk and significantly degrade response time and consume excessive CPU

Tuning Sort Summary

- OLTP and Web-based applications should not use sorts that overflow due the stringent response time requirements that must be met
  - Sort overflows should be 0 or close to 0 in this case
- BI/DW and Mixed environments perform larger sorts due to the complexity of the SQL and amount of data, hence, sort overflows are likely to occur
- Use the same tuning techniques to eliminate sorts as in OLTP but also tune the I/O subsystem to improve sorts that overflow

Event Monitoring and Sort Tuning Workshop

- Refer to your student workbook and examples for the Event Monitoring and Sort Tuning Workshop. You have 1 hr to complete the workshop.

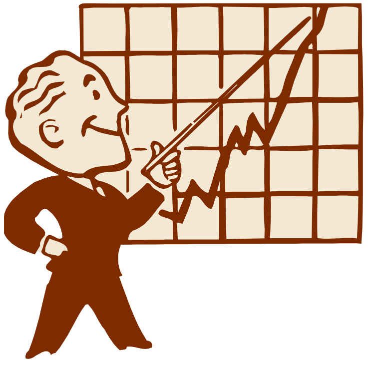
Module 5
DB2 Architecture

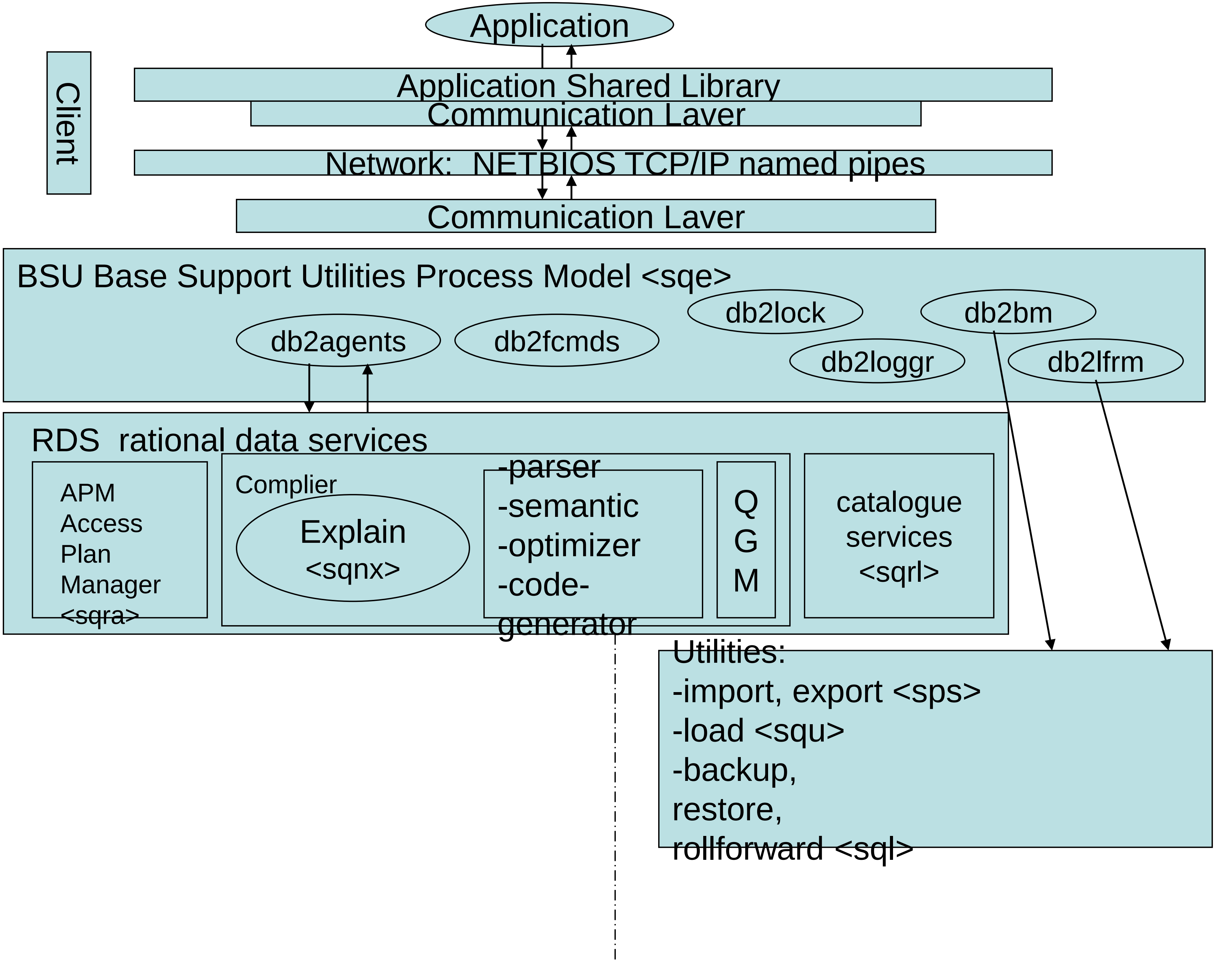
OBJECTIVES

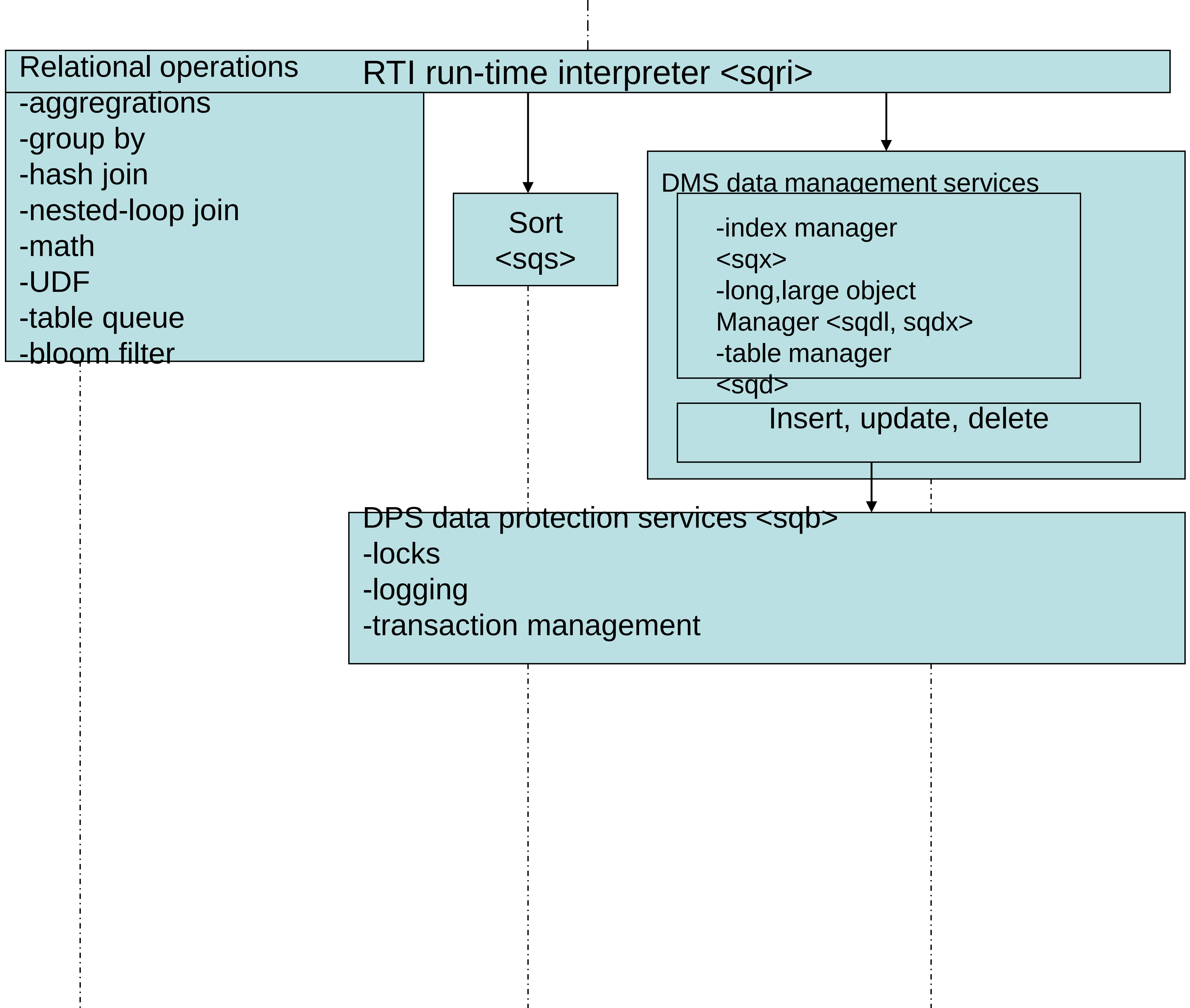
- DB2 Architecture
- Why Tune?
- How is memory used?
- Configuration Parameters
- Agent Related Parameters
- Summary

Introduction

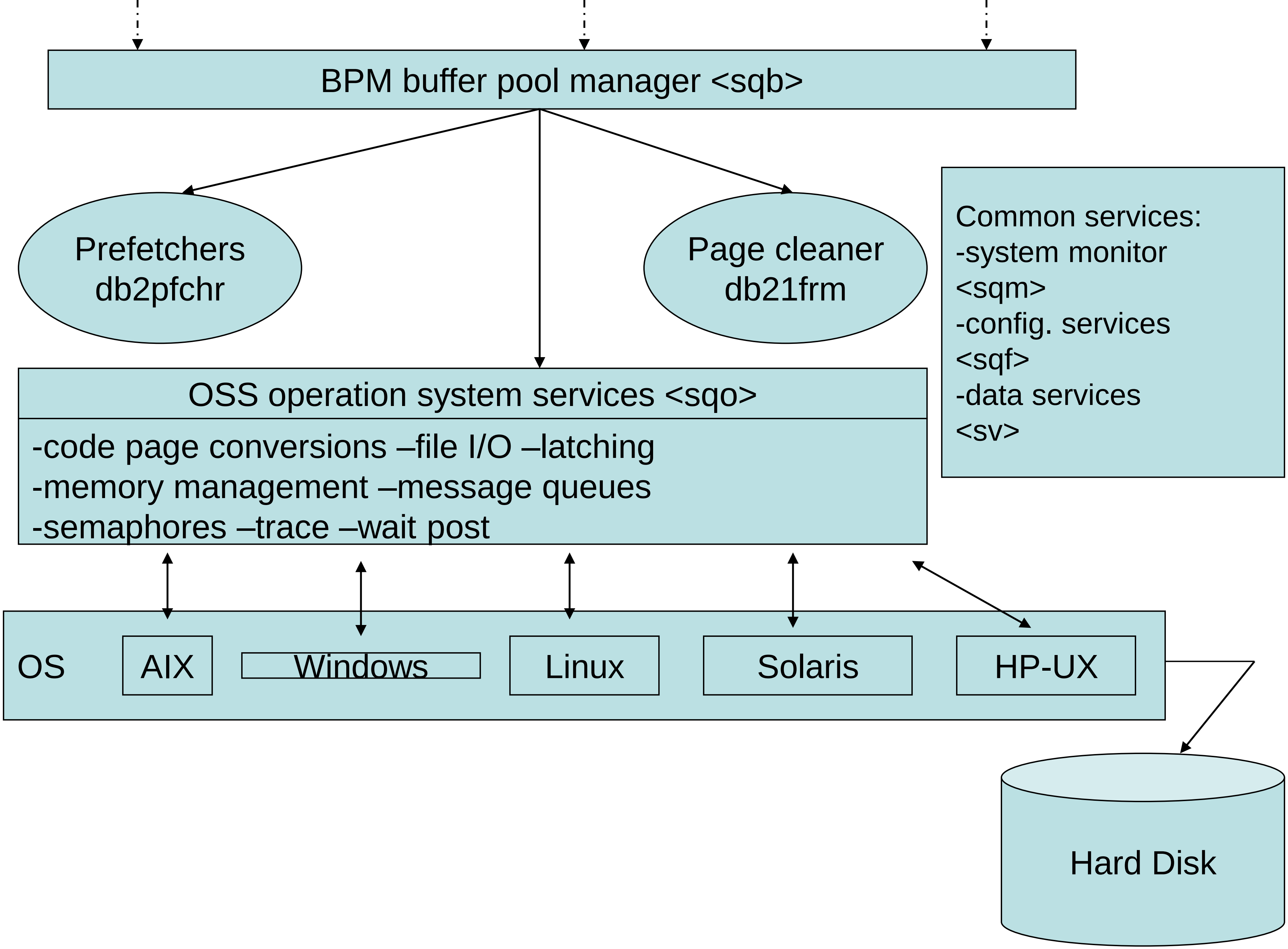
- Successful tuning requires knowledge of DB2 processing and available monitoring facilities



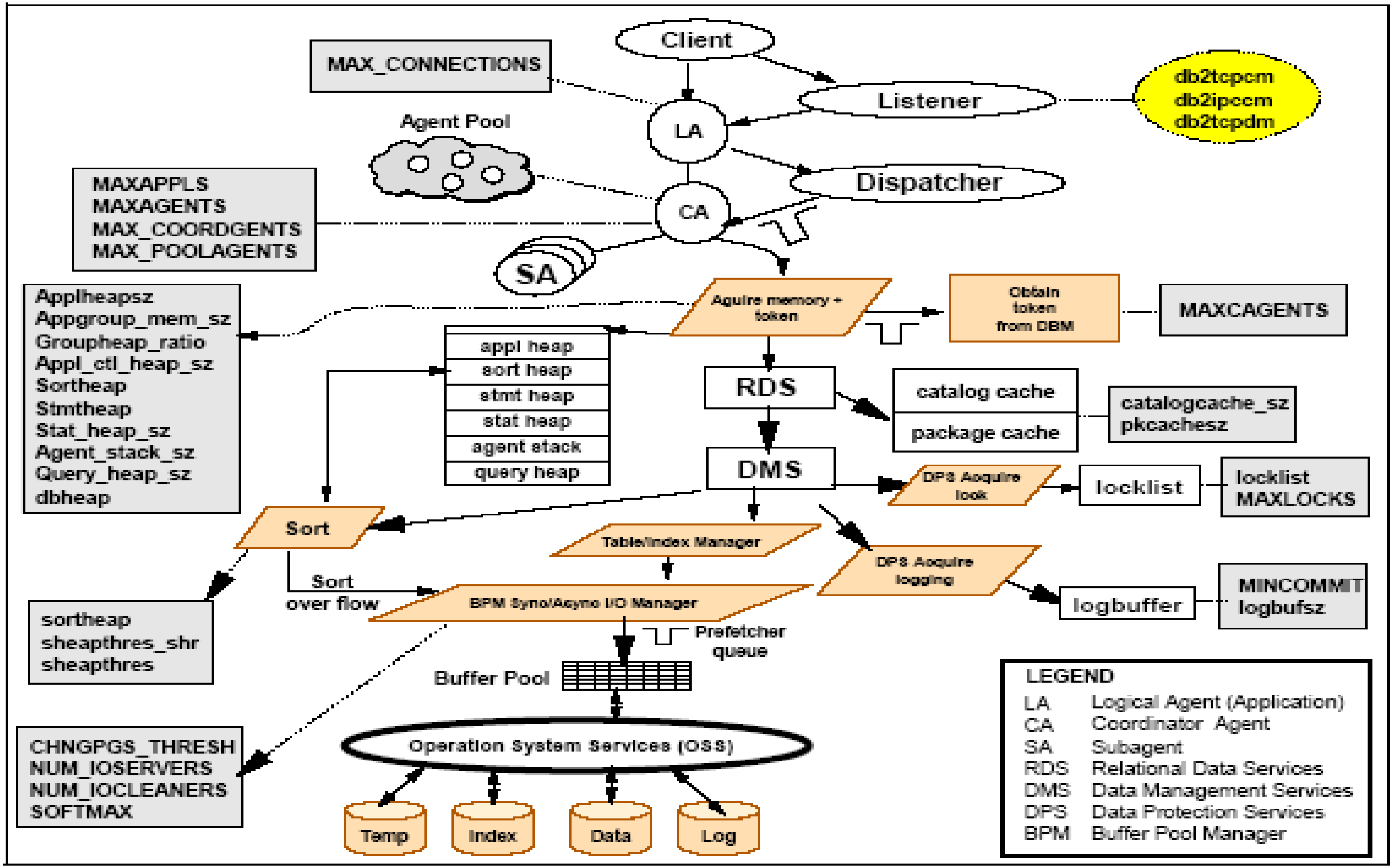

Application
Client
Application Shared Library
Communication Laver
Network: NETBIOS TCP/IP named pipes
Communication Laver
BSU Base Support Utilities Process Model <sqe>
db2lock
db2bm
db2agents
db2fcmds
db2loggr
db2lfrm
RDS rational data services
APM
Access
Plan
Manager
<sqra>
Complier
Explain
<sqnx>
-parser
-semantic
-optimizer
-code-
generator
Q
G
M
catalogue
services
<sqrl>
Utilities:
-import, export <sps>
-load <squ>
-backup,
restore,
rollforward <sql>



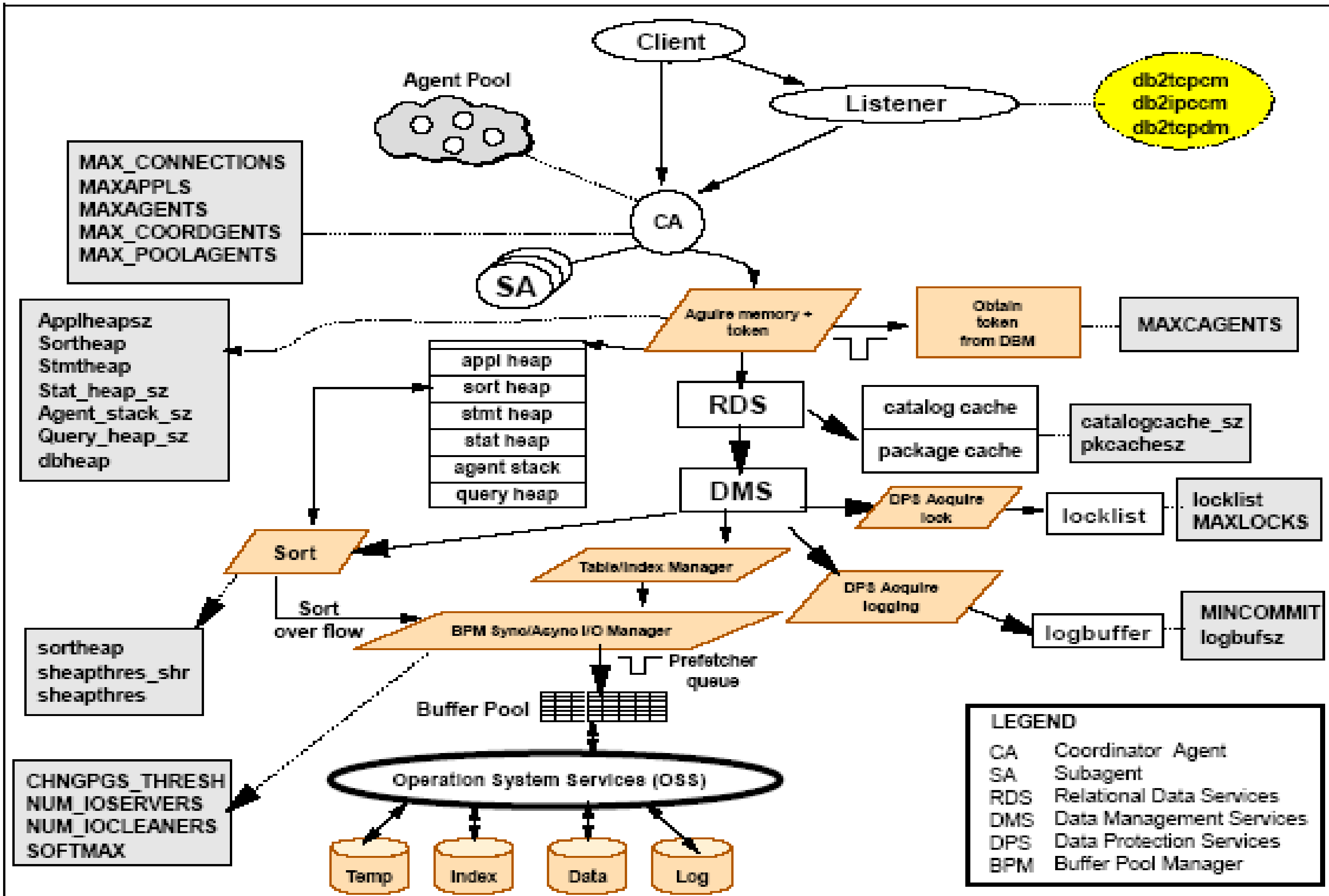
Relational operations
-aggregrations
-group by
-hash join
-nested-loop join
-math
-UDF
-table queue
-bloom filter
RTI run-time interpreter <sqri>
Sort
<sqs>
DMS data management services
-index manager
<sqx>
-long,large object
Manager <sqdl, sqdx>
-table manager
<sqd>
Insert, update, delete
DPS data protection services <sqb>
-locks
-logging
-transaction management


BPM buffer pool manager <sqb>
Prefetchers
db2pfchr
Page cleaner
db21frm
Common services:
-system monitor
<sqm>
-config. services
<sqf>
-data services
<sv>
OSS operation system services <sqo>
-code page conversions –file I/O –latching
-memory management –message queues
-semaphores –trace –wait post
OS
AIX
Windows
Linux
Solaris
HP-UX
Hard Disk

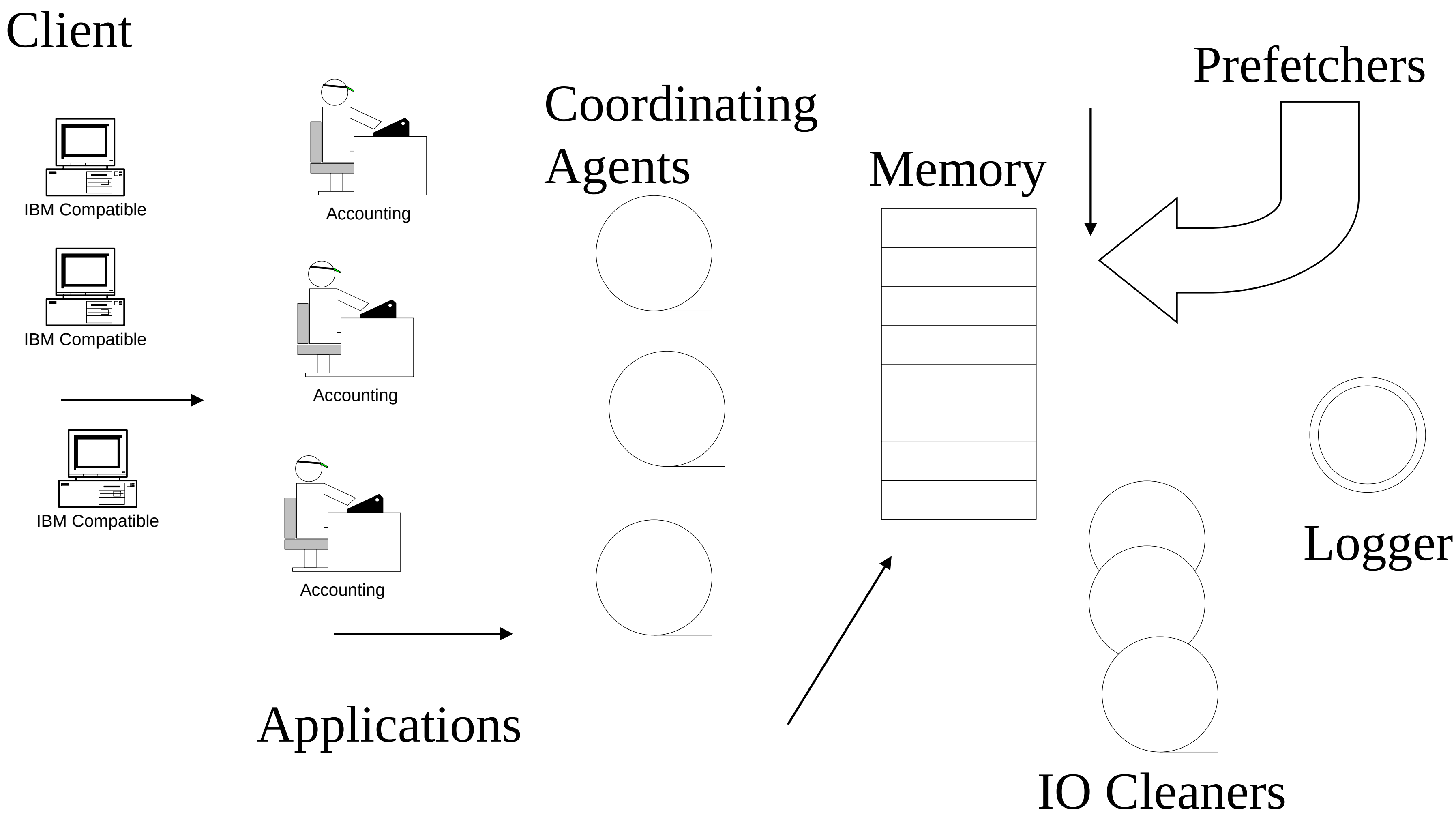
DB2 Process Model



DB2 Process Model



DB2 UDB Process Model



DB2 Process Model

- The DB2 Process Model consists of clients running applications, with agents doing work on behalf of applications. Data is prefetched into the bufferpool by prefetchers. As the bufferpool fills with dirty pages, IO Cleaners write dirty pages asnchronously to disk. The logger process writes undo/redo information to the log, and commits it from the log buffer at commit.

DB2 Prefetchers

- DB2 Prefetchers prefetch data into the bufferpool, in advance of the application needing it, when sequential access is being used. This prevents the application from waiting while data is retrieved from disk.
- The DB CFG NUM_IOSERVERS parameter specifies how many prefetchers will be available to DB2.

DB2 Prefetchers

- As agents request that pages be brought into the bufferpool, prefetch requests are placed on a common prefetch queue. Requests are processed from the queue using FIFO. At times, a prefetcher may not be available. In this case the agent does the prefetch requests which causes the application to wait until the request completes. This is not good!

DB2 Prefetchers

- As a rule of thumb, configure as many prefetchers as the number of disks. DB2 attempts to create a prefetcher for each container to obtain parallel I/O.

DB2 IOCLEANERS

- The DB CFG NUM_IOCLEANERS controls how many IOCLEANERS are available for a database. I/O cleaners are used to write dirty pages to disk. The CHNGPGS_THRES controls when IOCLEANERS are activated or woken up. The default threshold is 60%. That is if 60% of pages in the bufferpool are dirty, IOCLEANERS will write dirty pages to disk asynchronously.

DB2 IOCLEANERS

- This behavior is good, as we want asynchronous writes to occur. If the bufferpool fills up with dirty pages and an IOCLEANER is not available, the agent will have to cause the synchronous write to disk to occur which results in applications having to wait till the write completes. THIS IS NOT GOOD. Configure the NUM_IOCLEANERS parameter to the number of CPUs and monitor over time

Coordinating Agents

- A Coordinating agent is created for each application connected to the database. The coordinator agent does work on behalf of the application. The coordinator agent may create subagents if parallelism is enabled and if work can be done in parallel.

Bufferpools

- Bufferpools are used by DB2 to cache frequently accessed pages in memory. For a dedicated database server, up to 75% of memory should be dedicated to bufferpools as properly sized and configured bufferpools can have a significant effect on performance

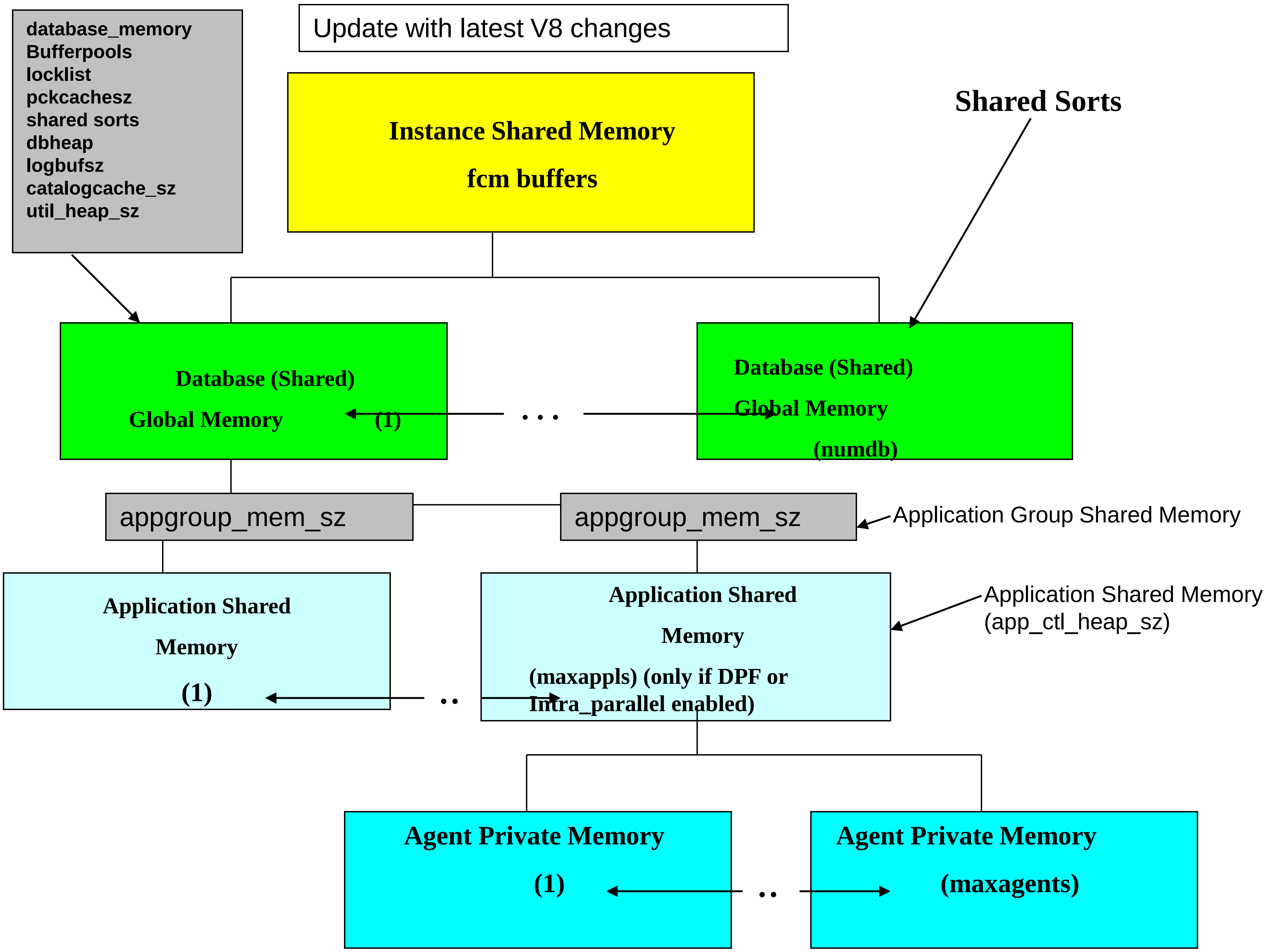
DB2 Logger

- The DB2 Logger is responsible to log redo/undo records and to write committed work to the log buffer and disk. The DB2 Logger works in conjunction with the Bufferpool Manager to ensure that updated pages are written to disk. In V8, we have true dual logging at the database level. This should be enabled at all times.

Module 6
Memory Model and Configuration Parameters

Objectives

- Upon completion of this module you should be able to:
  - Understand the DB2 Memory Model
  - Understand the heaps and parameters that affect how memory is used
  - Understand DB2 agent parameters
  - Be able to determine when and why parameters require tuning



database_memory
Bufferpools
locklist
pckcachesz
shared sorts
dbheap
logbufsz
catalogcache_sz
util_heap_sz
Update with latest V8 changes
Instance Shared Memory
fcm buffers
Shared Sorts
Database (Shared)
Global Memory
(1)
. . .
Database (Shared)
Global Memory
(numdb)
appgroup_mem_sz
appgroup_mem_sz
Application Group Shared Memory
Application Shared
Memory
(1)
..
Application Shared
Memory
(maxappls) (only if DPF or
Intra_parallel enabled)
Application Shared Memory
(app_ctl_heap_sz)
Agent Private Memory
(1)
..
Agent Private Memory
(maxagents)

Application Groups

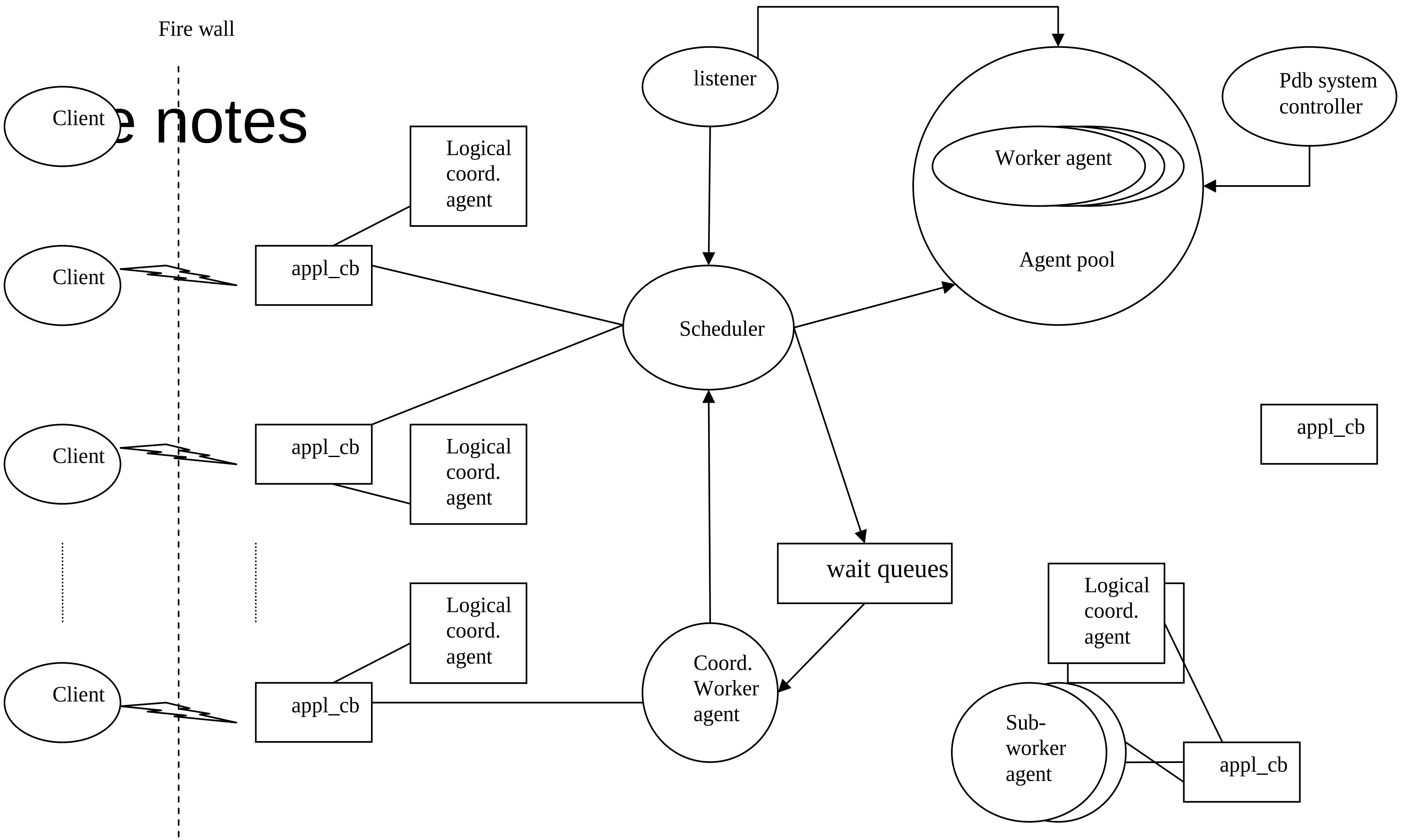
- See notes

Application Groups

- See notes

Connection Concentrator

- e notes



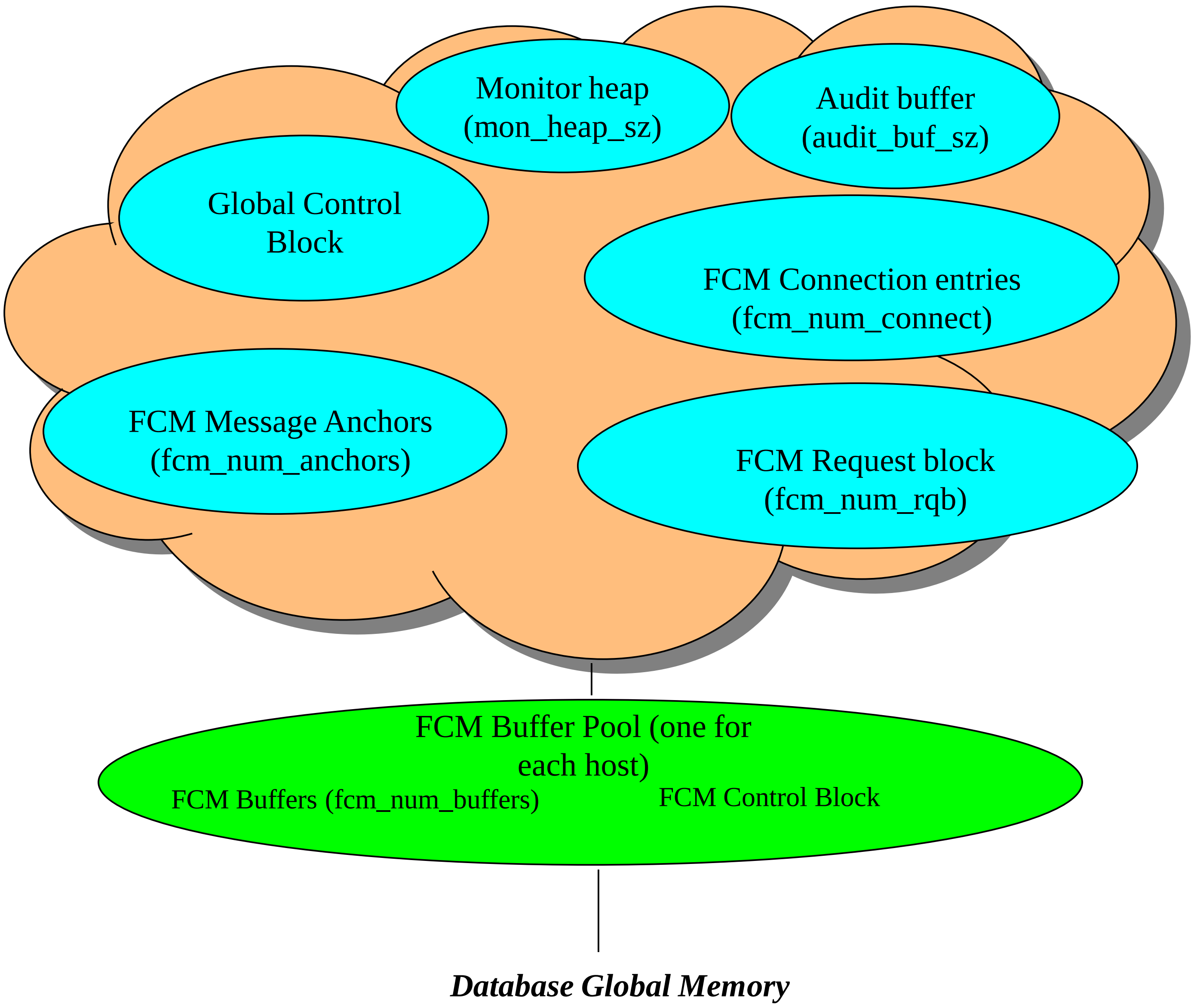
Dynamic Configuration Parameters

- Deferred
  - Get DBM or DB CFG
- Immediate
- Transaction boundary

Autonomous Computing

- Self-managing and Resource Tuning (SMART)
- Configuration parameters that can be set to automatic

Database Manager Shared Memory



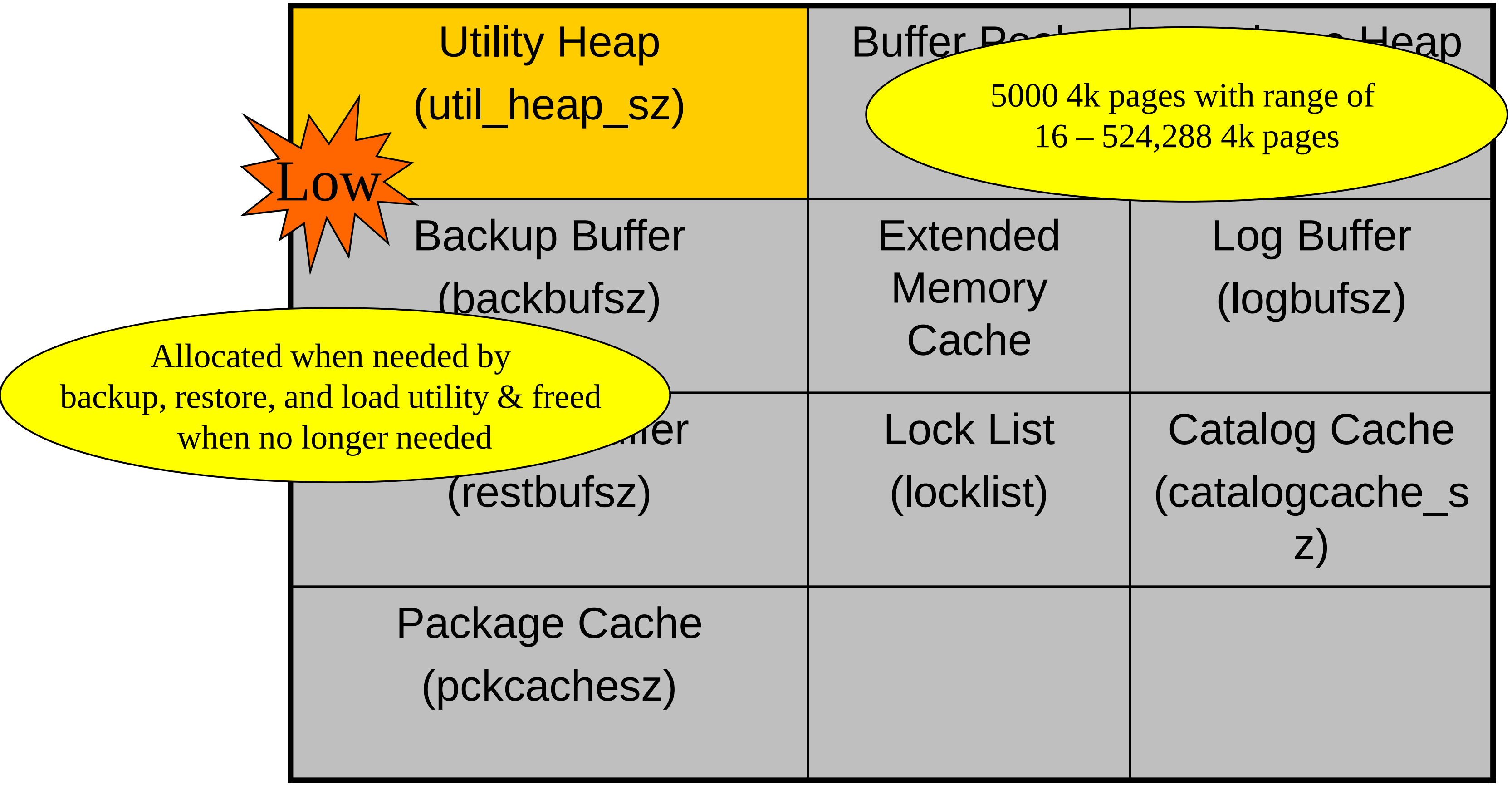
Database Manager Shared Memory

Database Global Memory

Utility Heap (util_heap_sz)	Buffer Pools (buffpage)	Database Heap (dbheap)
Backup Buffer (backbufsz)	Extended Memory Cache	Log Buffer (logbufsz)
Restore Buffer (restbufsz)	Lock List (locklist)	Catalog Cache (catalogcache_sz)
Package Cache (pckcachesz)	Sort Heap – Shared Sort (sortheap)	

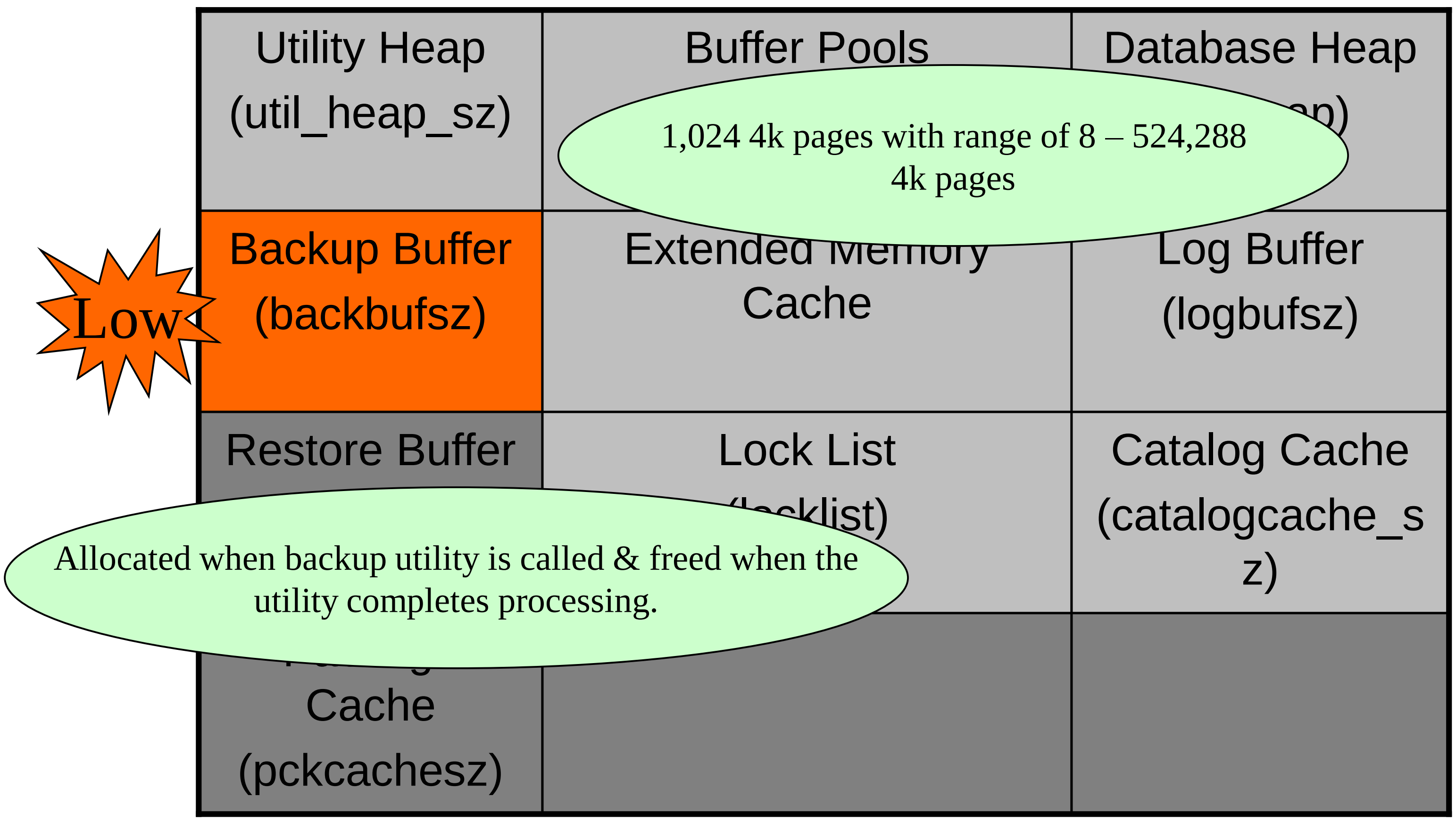
Database Manager Shared Memory

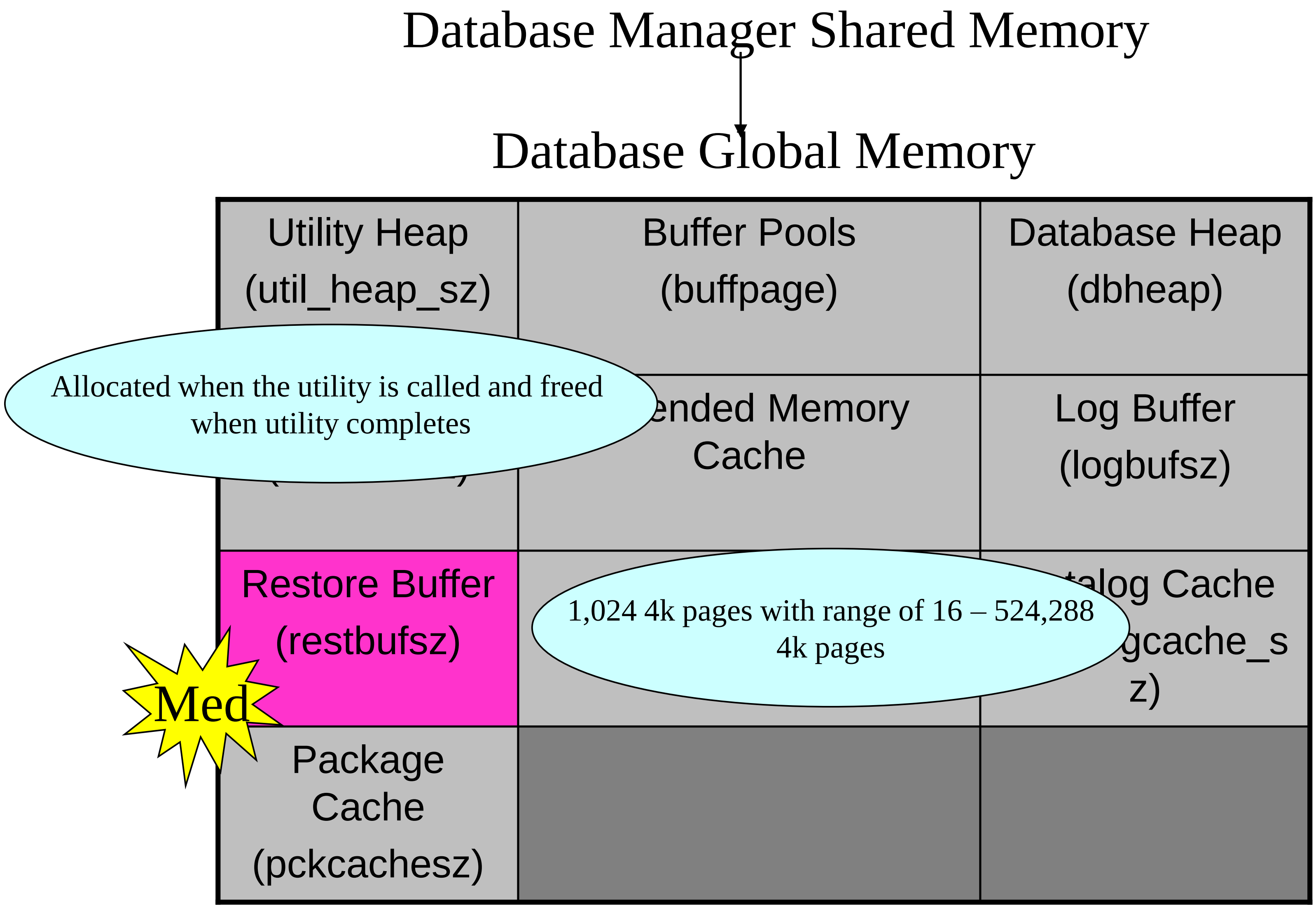
Database Global Memory

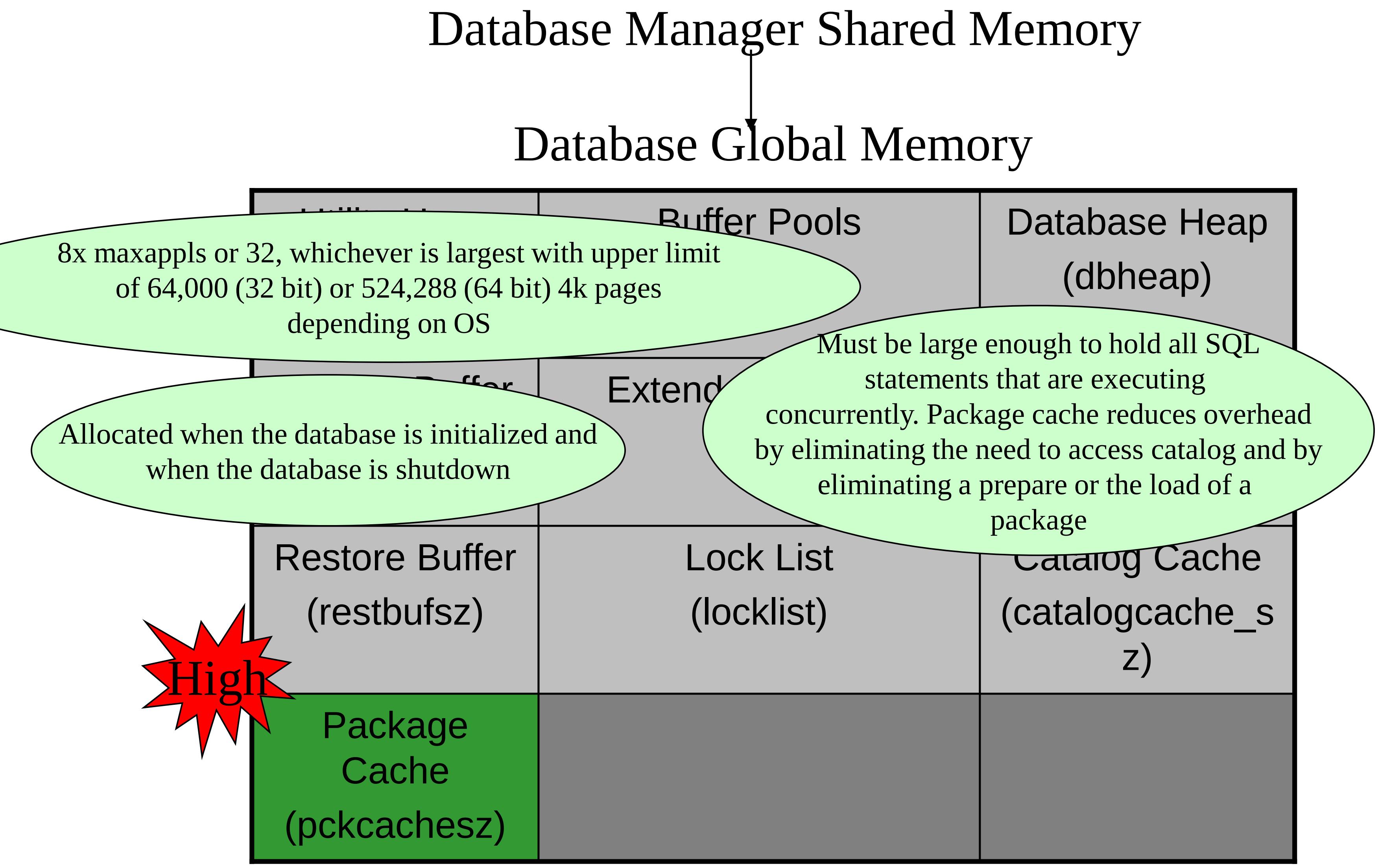


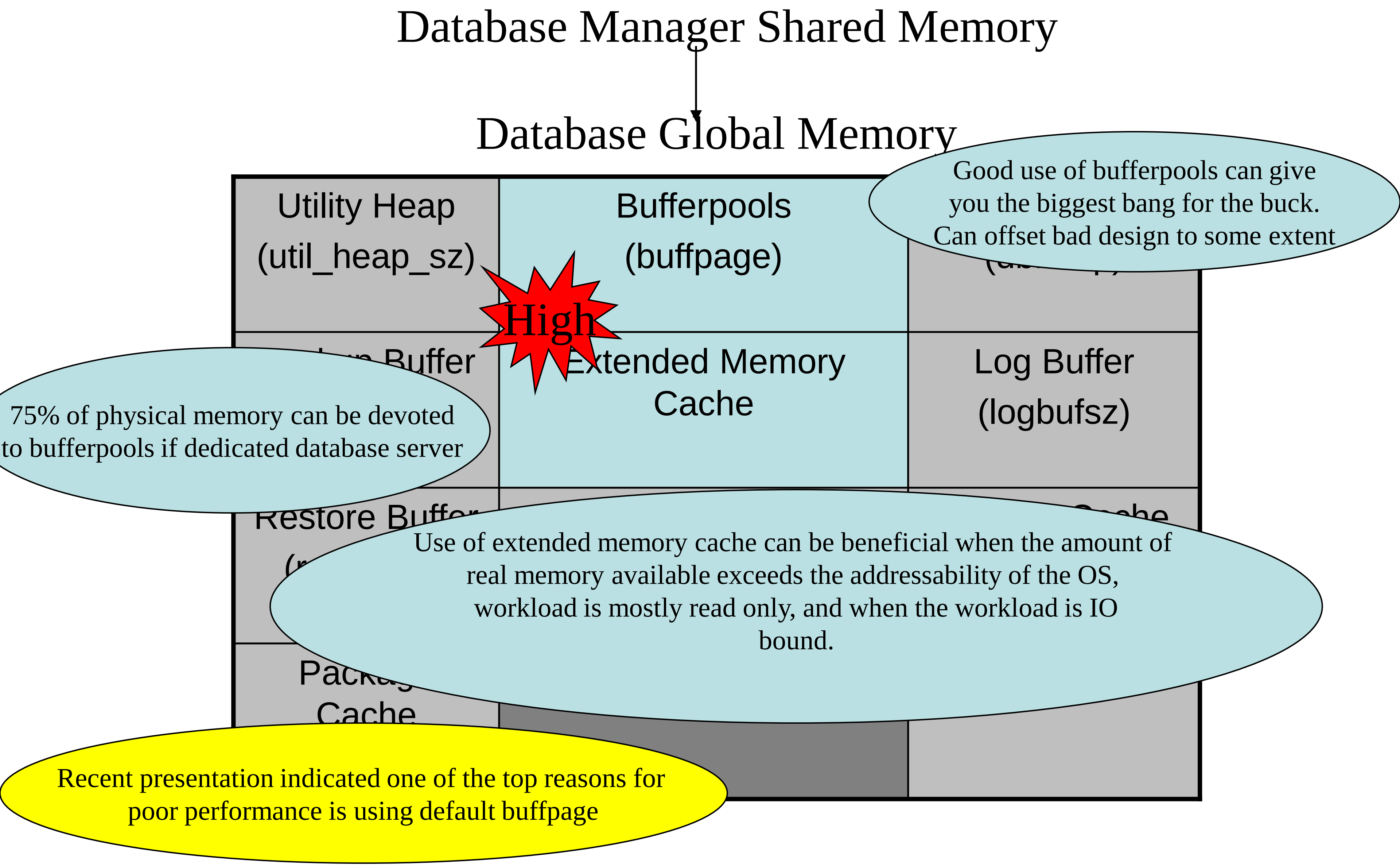
Database Manager Shared Memory

Database Global Memory



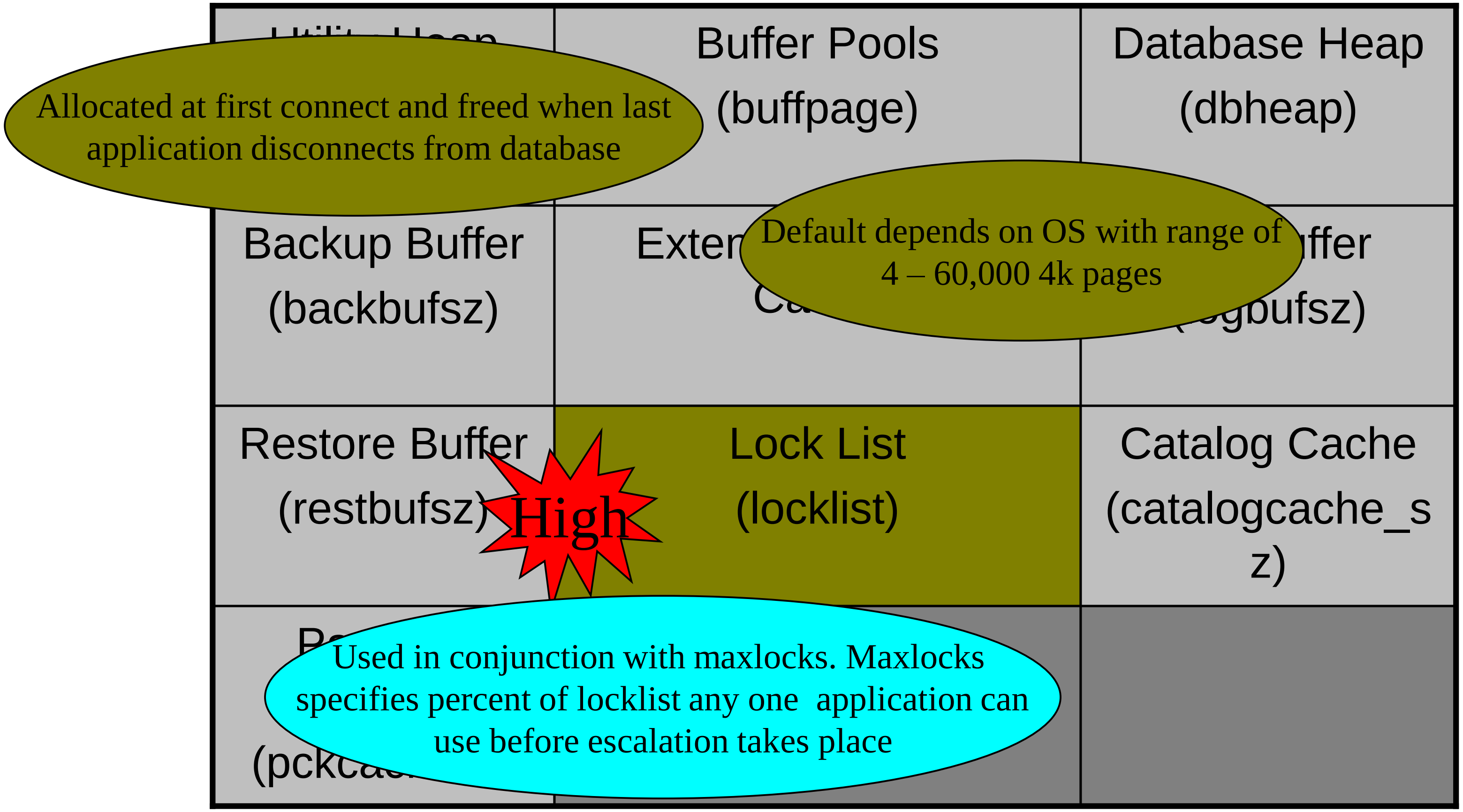

Database Manager Shared Memory
Database Global Memory
Utility Heap
(util_heap_sz)
Buffer Pools
(buffpage)
Database Heap
(dbheap)
Allocated when the utility is called and freed when utility completes
ended Memory
Cache
Log Buffer
(logbufsz)
Restore Buffer
(restbufsz)
1,024 4k pages with range of 16 – 524,288 4k pages
talog Cache
gcache_s
z)
Med
Package
Cache
(pckcachesz)


Database Manager Shared Memory
Database Global Memory
Buffer Pools
Database Heap
(dbheap)
8x maxappls or 32, whichever is largest with upper limit of 64,000 (32 bit) or 524,288 (64 bit) 4k pages depending on OS
Allocated when the database is initialized and when the database is shutdown
Must be large enough to hold all SQL statements that are executing concurrently. Package cache reduces overhead by eliminating the need to access catalog and by eliminating a prepare or the load of a package
Restore Buffer
(restbufsz)
Lock List
(locklist)
Catalog Cache
(catalogcache_sz)
High
Package Cache
(pckcachesz)


Database Manager Shared Memory
Database Global Memory
Utility Heap
(util_heap_sz)
Bufferpools
(buffpage)
High
Extended Memory
Cache
Log Buffer
(logbufsz)
Restore Buffer
Package
Cache
Good use of bufferpools can give you the biggest bang for the buck. Can offset bad design to some extent
75% of physical memory can be devoted to bufferpools if dedicated database server
Use of extended memory cache can be beneficial when the amount of real memory available exceeds the addressability of the OS, workload is mostly read only, and when the workload is IO bound.
Recent presentation indicated one of the top reasons for poor performance is using default buffpage

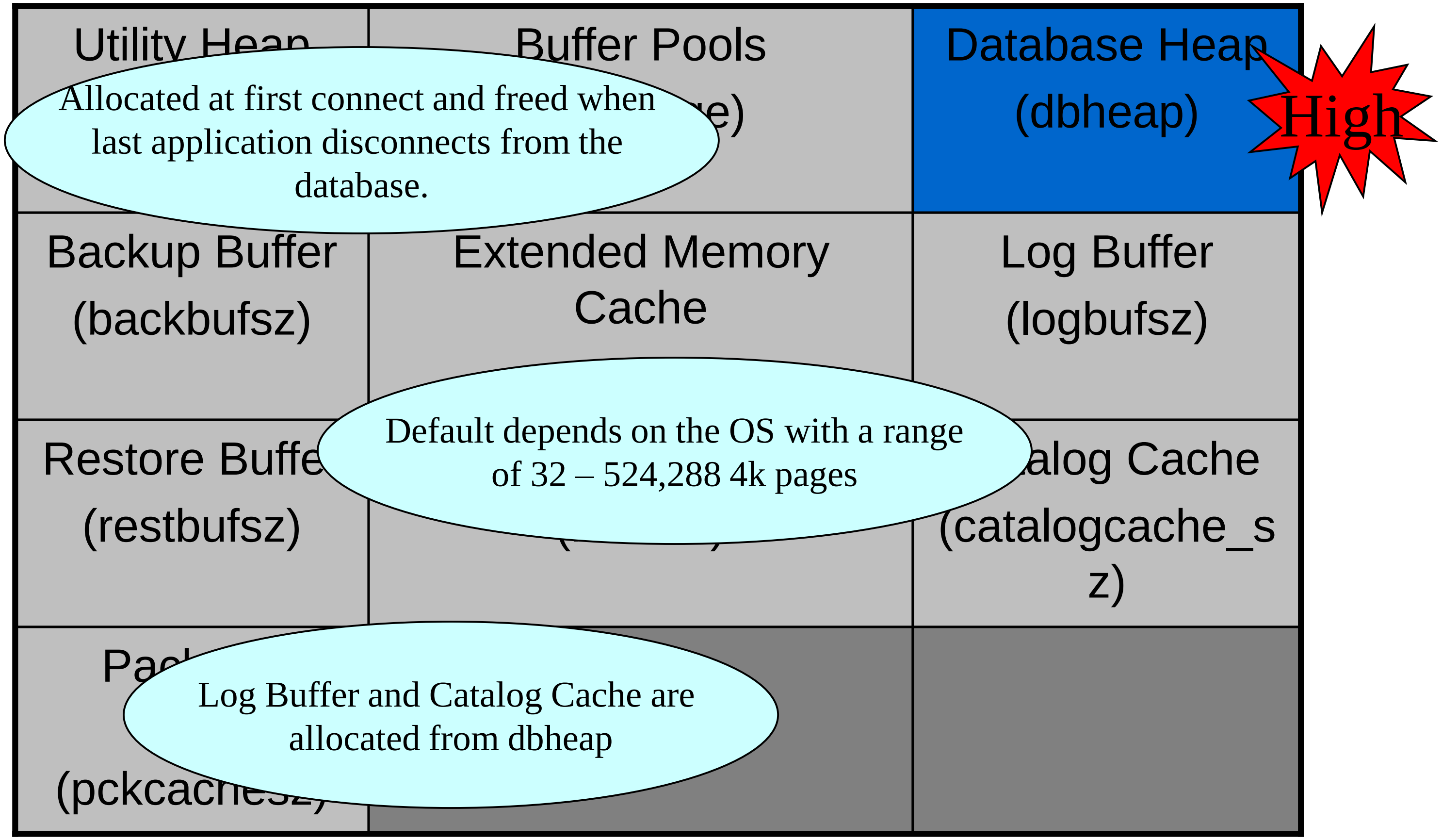
Database Manager Shared Memory

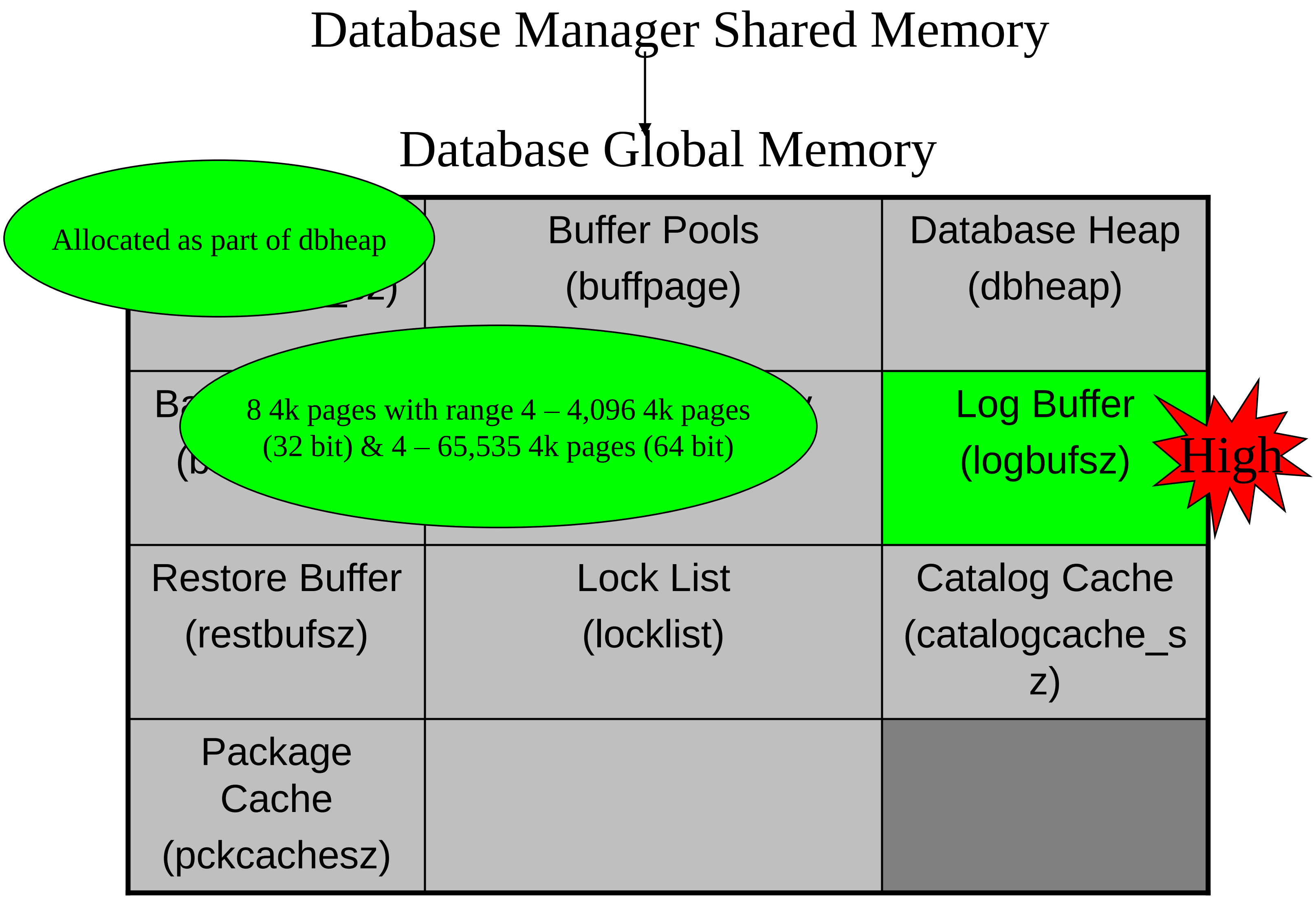
Database Global Memory



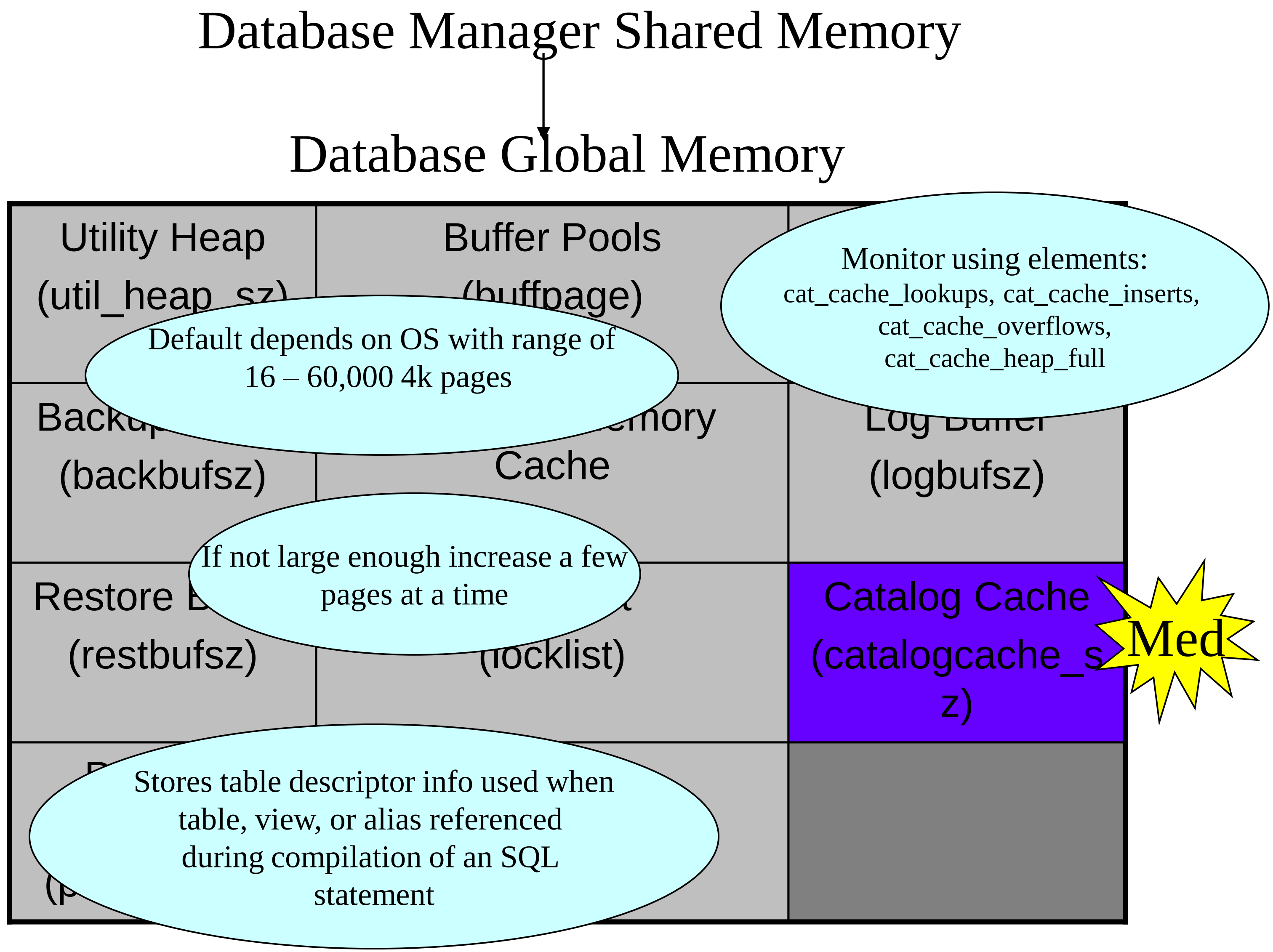
Database Manager Shared Memory

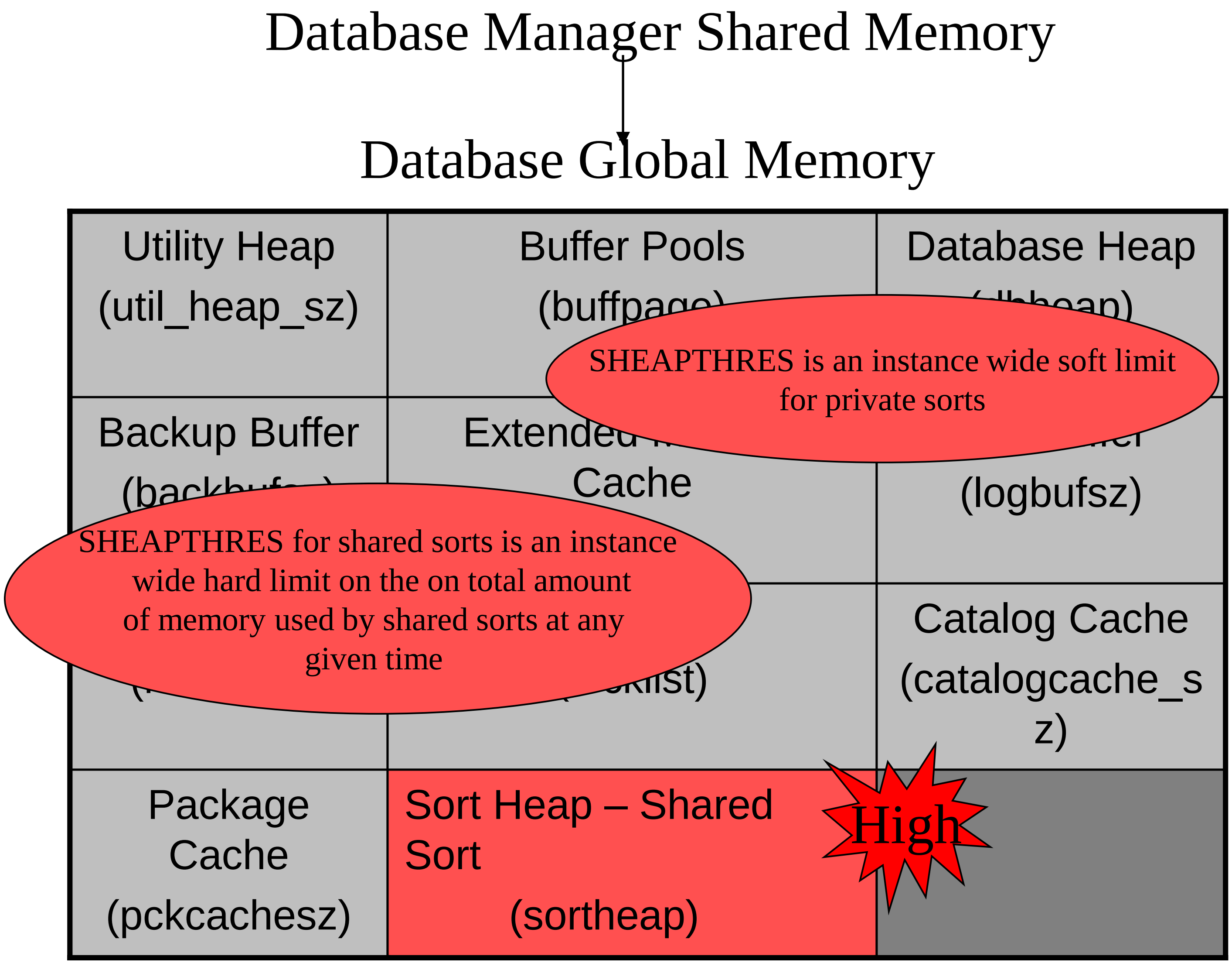
Database Global Memory

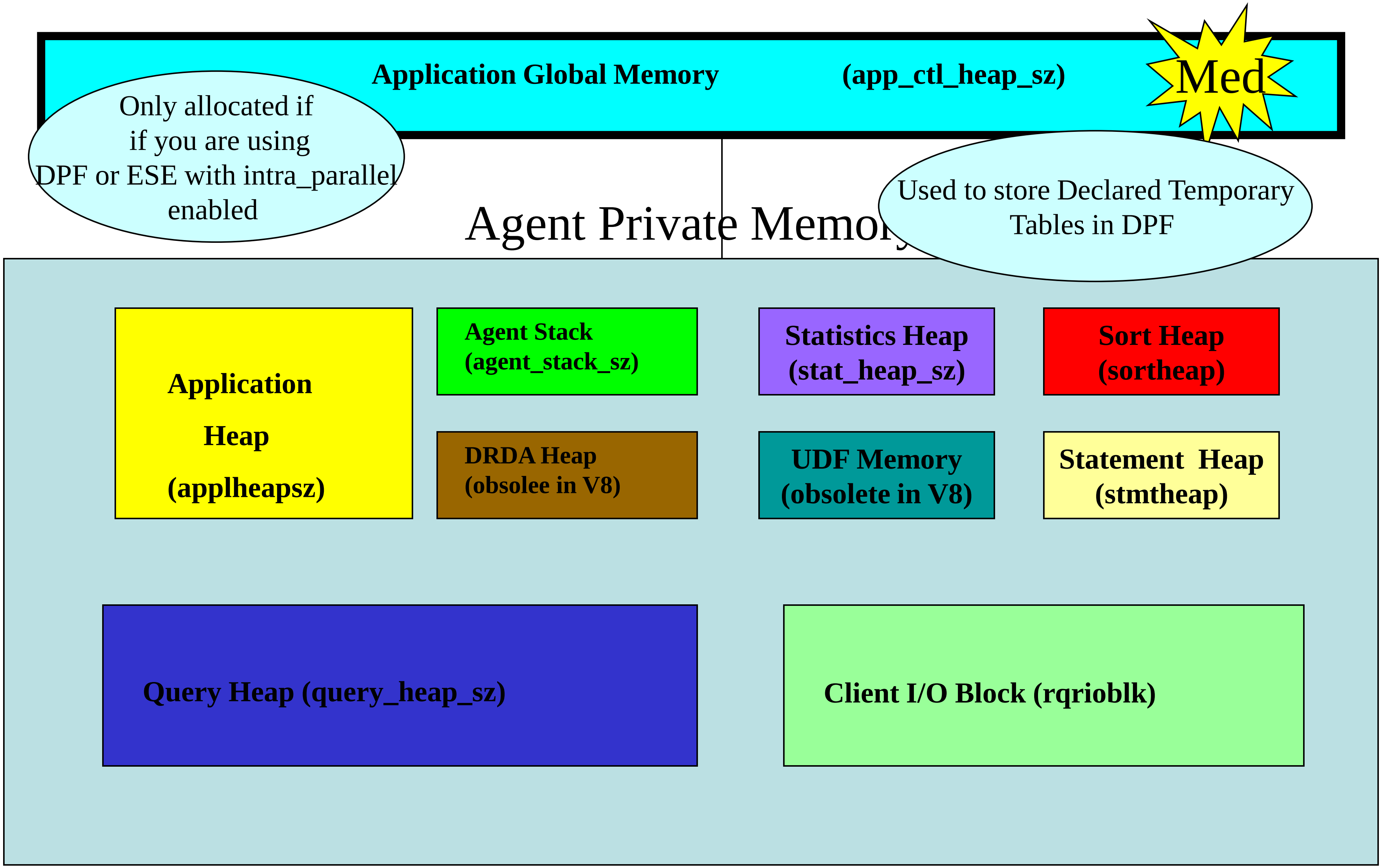




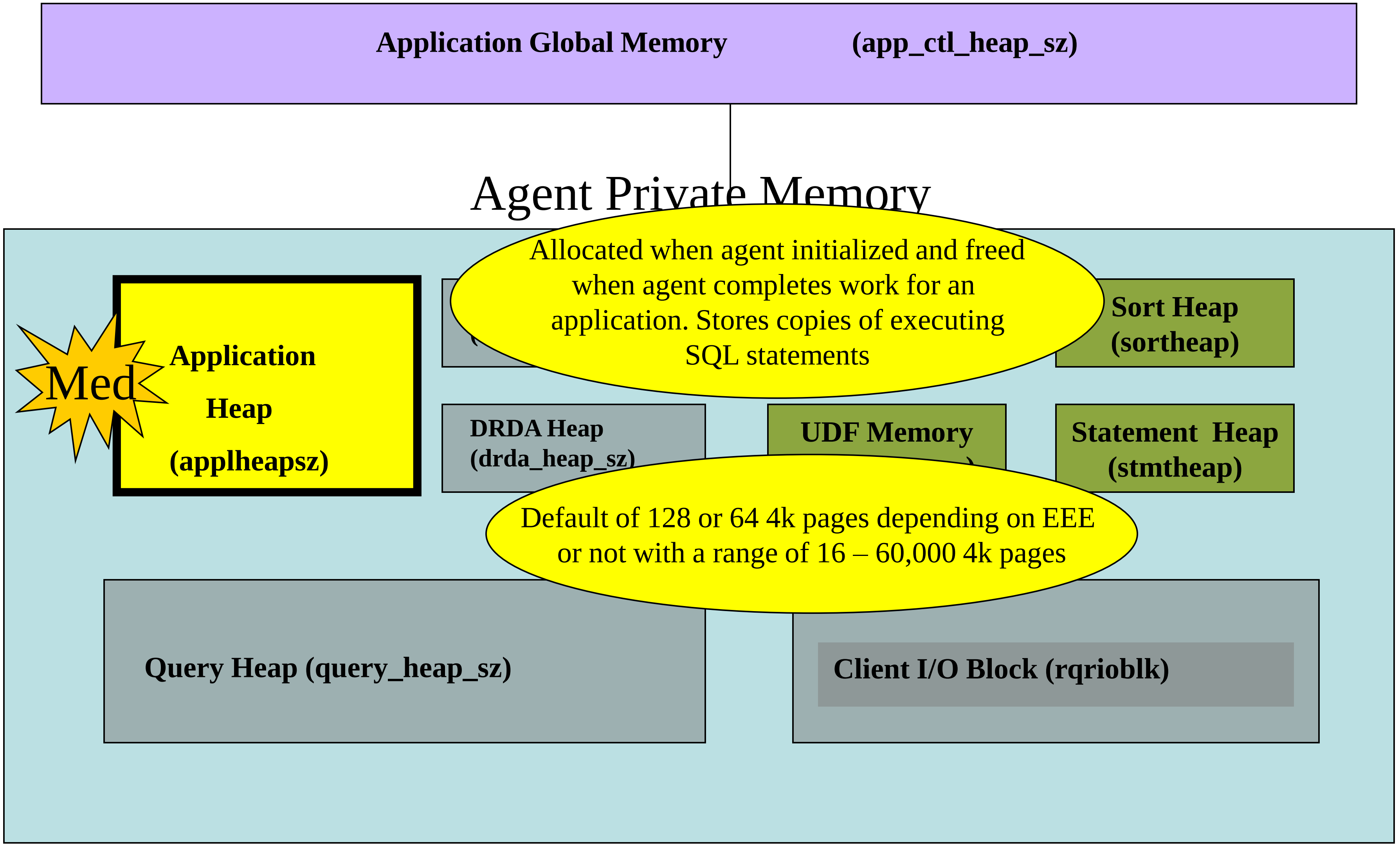
Database Manager Shared Memory
Database Global Memory
Allocated as part of dbheap
Buffer Pools
(buffpage)
Database Heap
(dbheap)
8 4k pages with range 4 – 4,096 4k pages (32 bit) & 4 – 65,535 4k pages (64 bit)
Log Buffer
(logbufsz)
High
Restore Buffer
(restbufsz)
Lock List
(locklist)
Catalog Cache
(catalogcache_sz)
Package Cache
(pckcachesz)

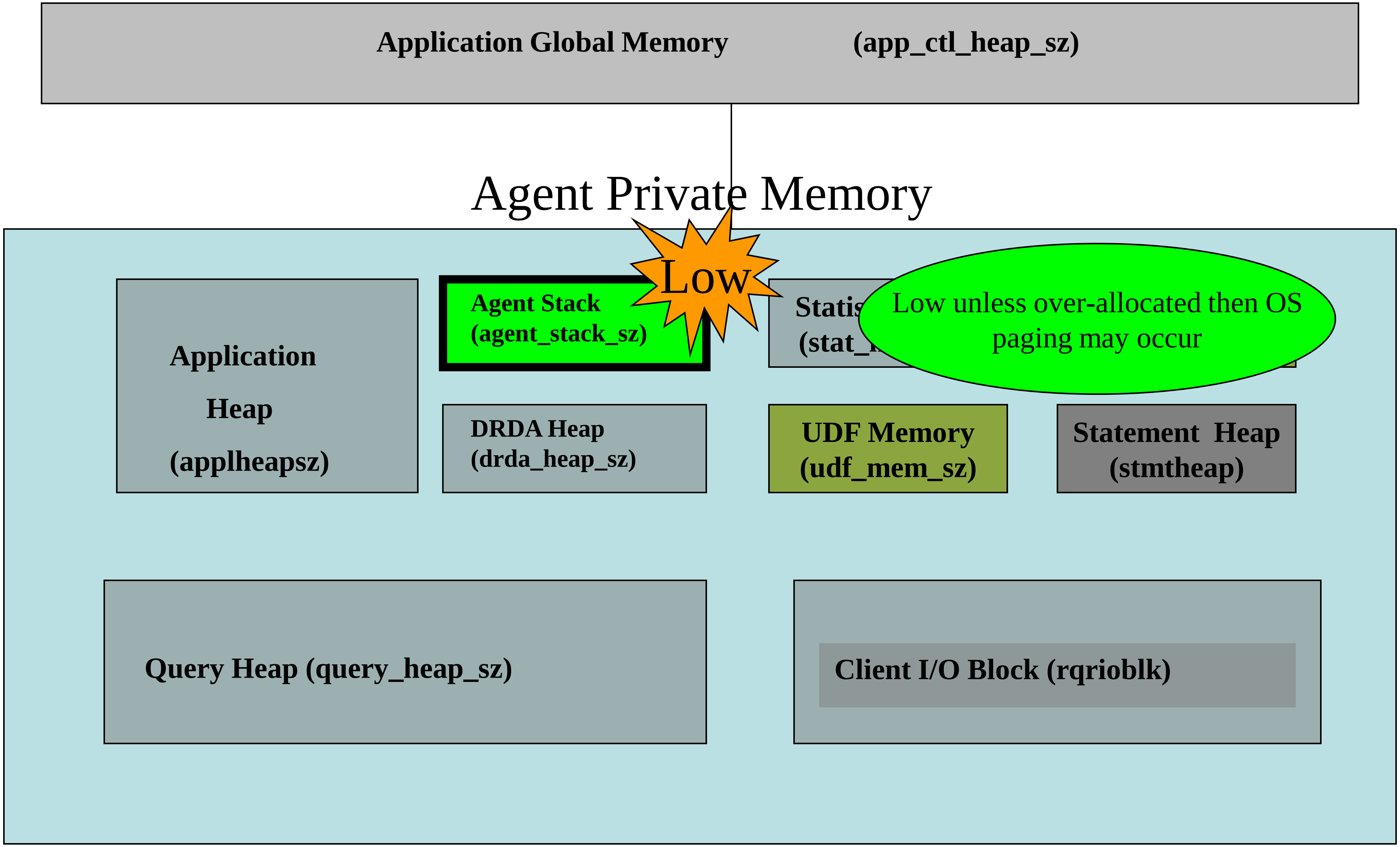

Database Manager Shared Memory
Database Global Memory
Utility Heap
(util_heap_sz)
Buffer Pools
(buffpage)
Monitor using elements:
cat_cache_lookups, cat_cache_inserts,
cat_cache_overflows,
cat_cache_heap_full
Default depends on OS with range of
16 – 60,000 4k pages
(backbufsz)
Cache
(logbufsz)
If not large enough increase a few
pages at a time
Catalog Cache
(catalogcache_s
z)
Med
(restbufsz)
(locklist)
Stores table descriptor info used when
table, view, or alias referenced
during compilation of an SQL
statement


Database Manager Shared Memory
Database Global Memory
Utility Heap
(util_heap_sz)
Buffer Pools
Database Heap
SHEAPTHRES is an instance wide soft limit for private sorts
Backup Buffer
Cache
(logbufsz)
SHEAPTHRES for shared sorts is an instance wide hard limit on the on total amount of memory used by shared sorts at any given time
Catalog Cache
(catalogcache_sz)
Package Cache
(pckcachesz)
Sort Heap – Shared Sort
(sortheap)
High

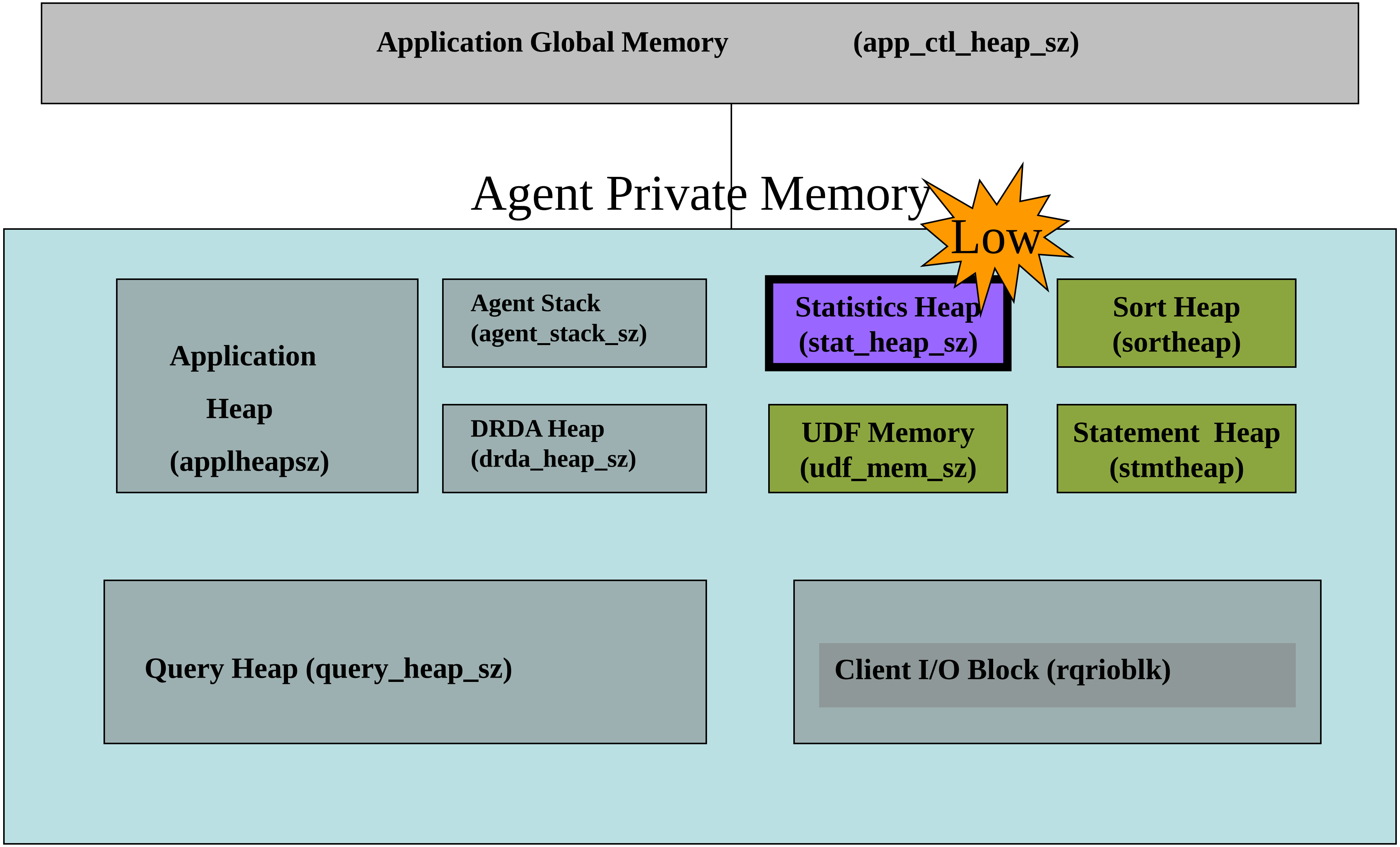


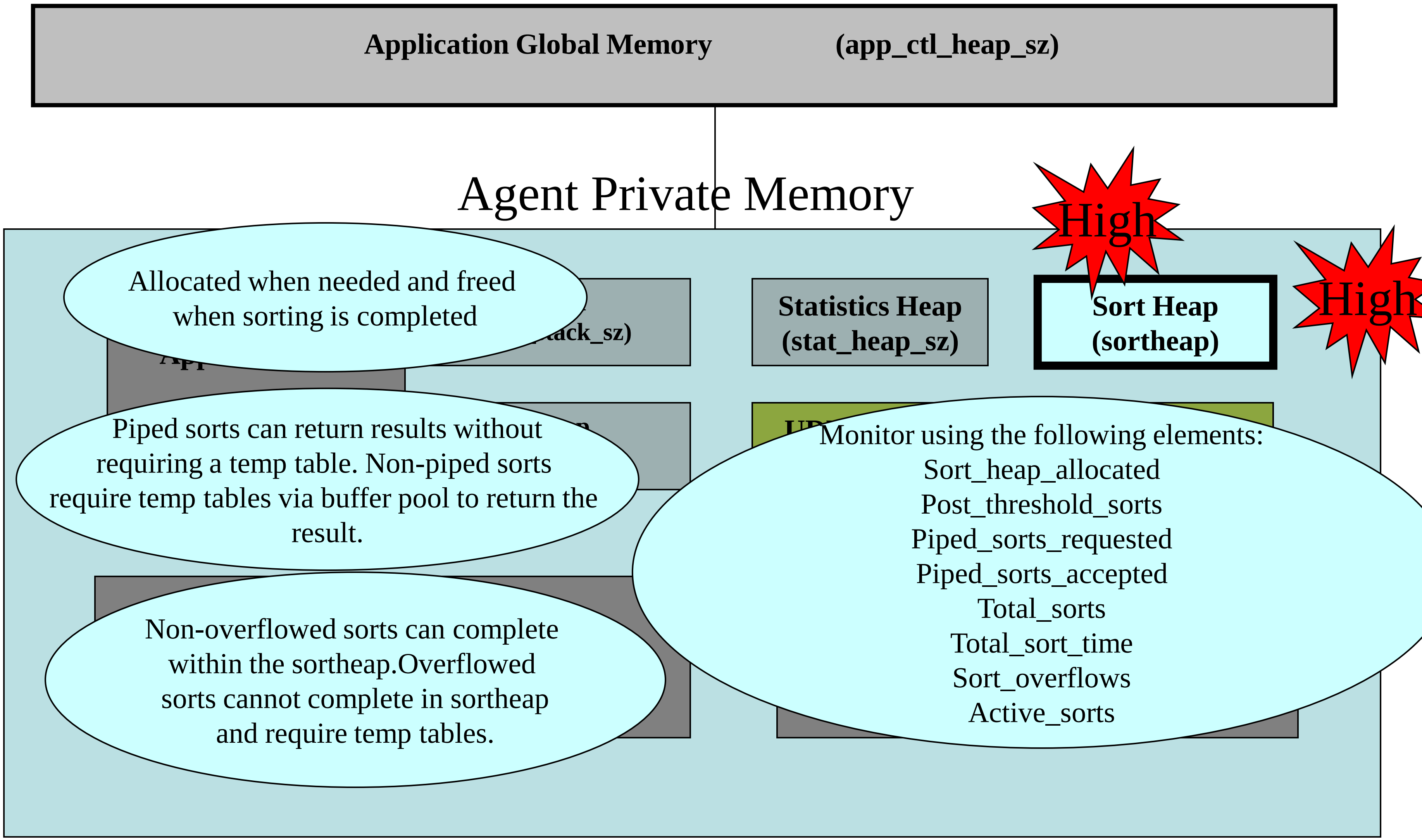
Application Global Memory (app_ctl_heap_sz)
Med
Only allocated if if you are using DPF or ESE with intra_parallel enabled
Used to store Declared Temporary Tables in DPF
Agent Private Memory
Application Heap (applheapsz)
Agent Stack (agent_stack_sz)
Statistics Heap (stat_heap_sz)
Sort Heap (sortheap)
DRDA Heap (obsolee in V8)
UDF Memory (obsolete in V8)
Statement Heap (stmtheap)
Query Heap (query_heap_sz)
Client I/O Block (rqrioblk)


Application Global Memory (app_ctl_heap_sz)
Agent Private Memory
Allocated when agent initialized and freed when agent completes work for an application. Stores copies of executing SQL statements
Med
Application Heap (applheapsz)
Sort Heap (sortheap)
DRDA Heap (drda_heap_sz)
UDF Memory
Statement Heap (stmtheap)
Default of 128 or 64 4k pages depending on EEE or not with a range of 16 – 60,000 4k pages
Query Heap (query_heap_sz)
Client I/O Block (rqrioblk)

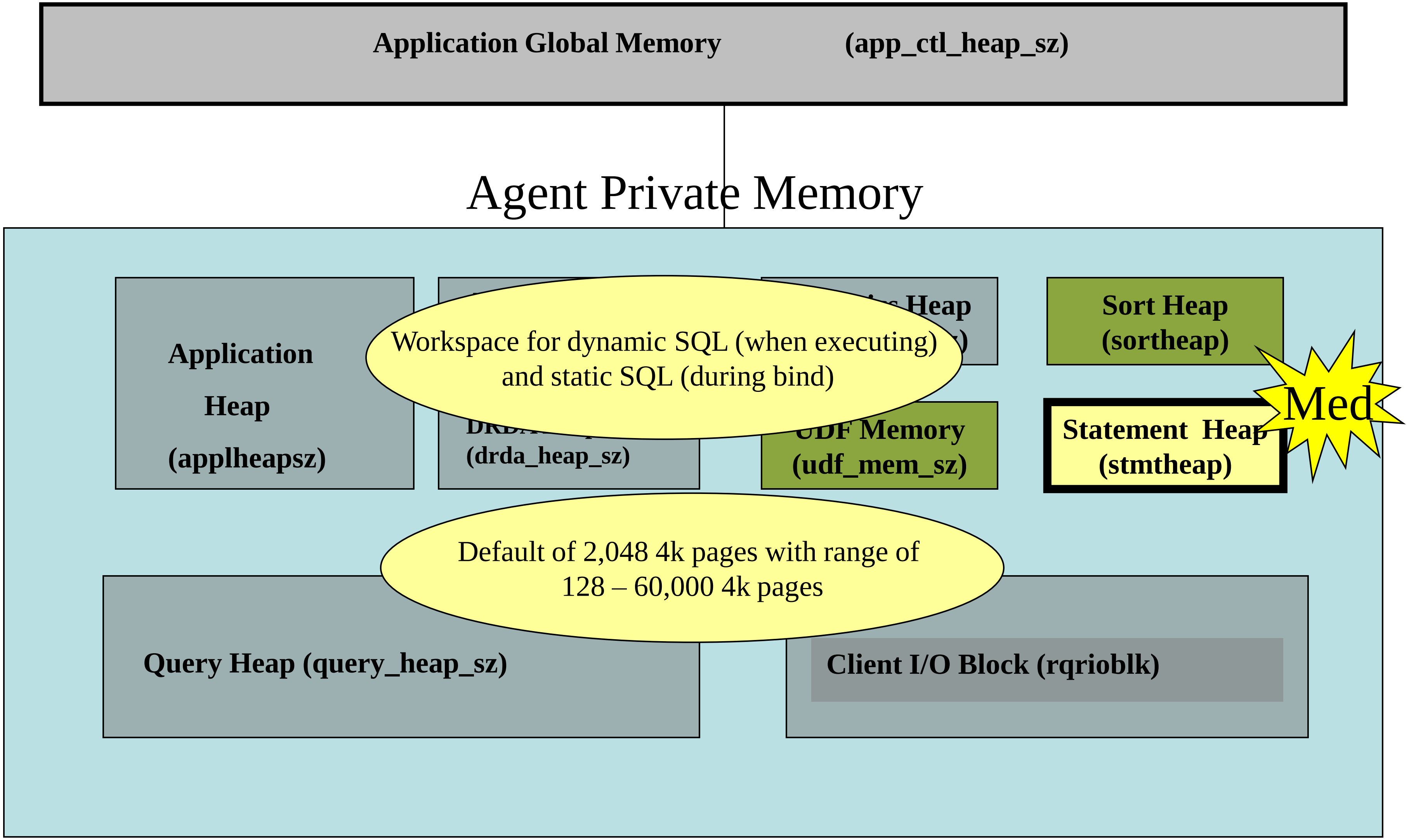


Application Global Memory (app_ctl_heap_sz)
Agent Private Memory
Low
Application Heap (applheapsz)
Agent Stack (agent_stack_sz)
Statis
(stat_
Low unless over-allocated then OS paging may occur
DRDA Heap (drda_heap_sz)
UDF Memory (udf_mem_sz)
Statement Heap (stmtheap)
Query Heap (query_heap_sz)
Client I/O Block (rqrioblk)

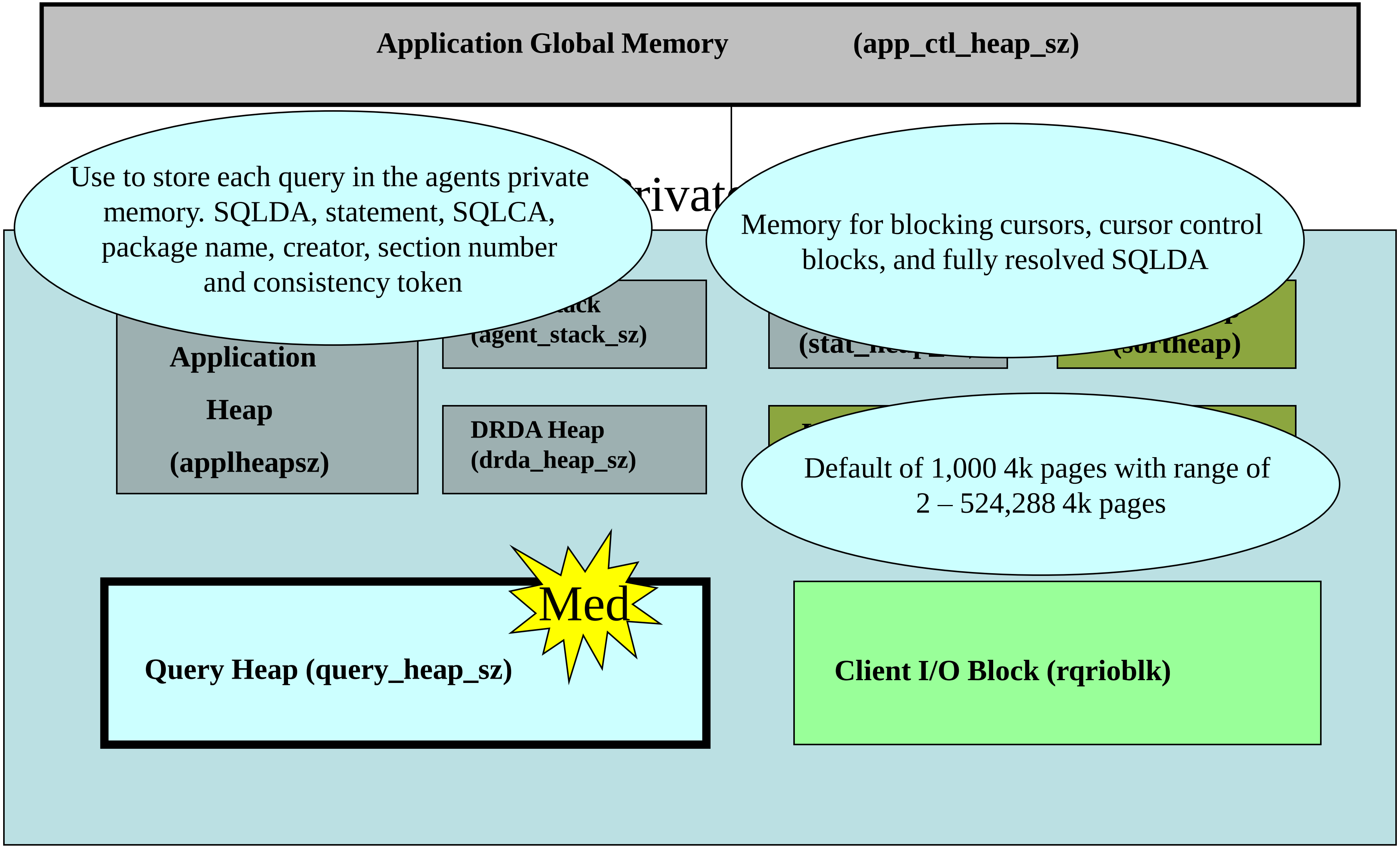

Application Global Memory (app_ctl_heap_sz)
Agent Private Memory
Low
Application Heap (applheapsz)
Agent Stack (agent_stack_sz)
DRDA Heap (drda_heap_sz)
Statistics Heap (stat_heap_sz)
Sort Heap (sortheap)
UDF Memory (udf_mem_sz)
Statement Heap (stmtheap)
Query Heap (query_heap_sz)
Client I/O Block (rqrioblk)



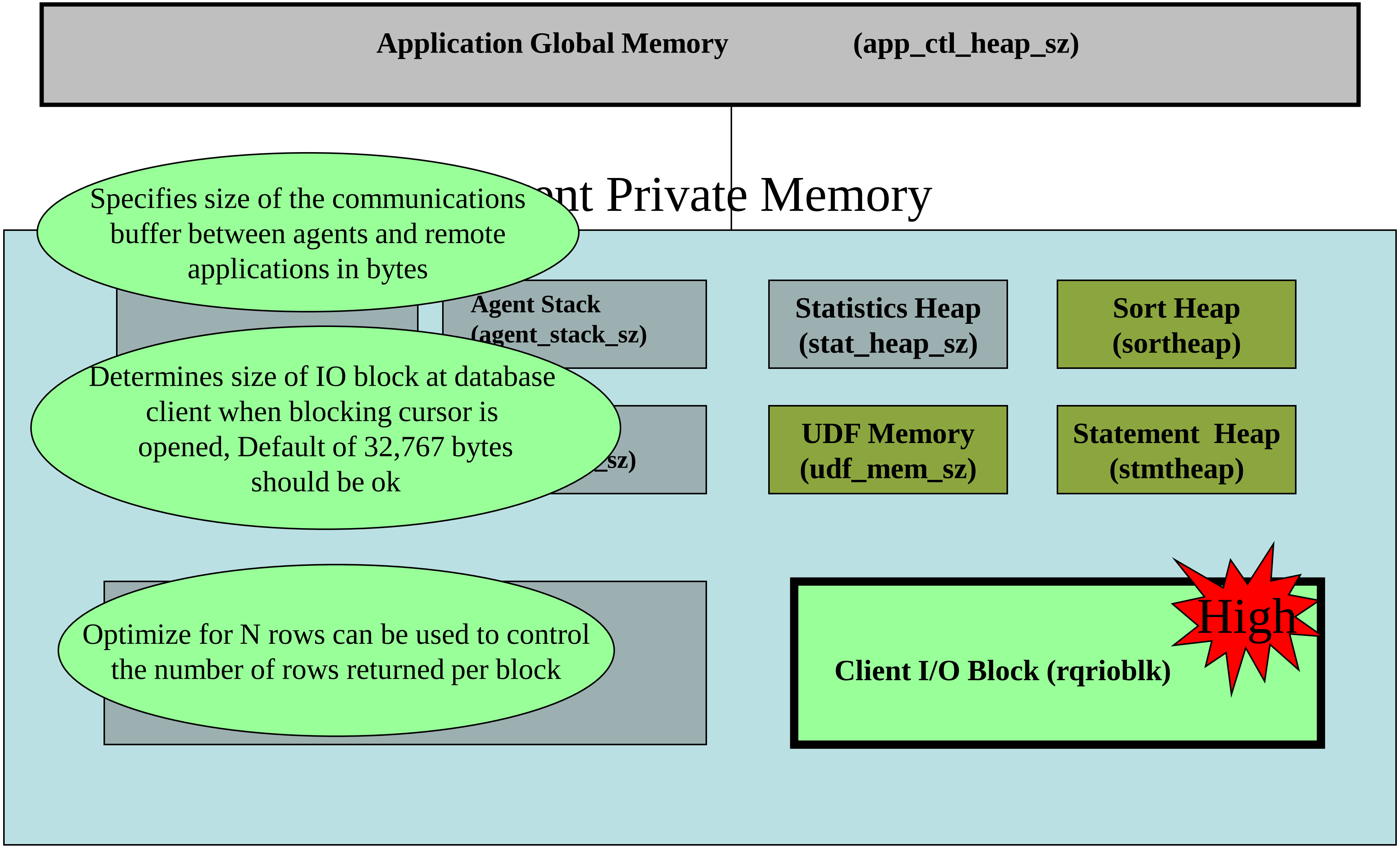
Application Global Memory (app_ctl_heap_sz)
Agent Private Memory
High
High
Allocated when needed and freed when sorting is completed
Statistics Heap (stat_heap_sz)
Sort Heap (sortheap)
Piped sorts can return results without requiring a temp table. Non-piped sorts require temp tables via buffer pool to return the result.
Monitor using the following elements:
Sort_heap_allocated
Post_threshold_sorts
Piped_sorts_requested
Piped_sorts_accepted
Total_sorts
Total_sort_time
Sort_overflows
Active_sorts
Non-overflowed sorts can complete within the sortheap.Overflowed sorts cannot complete in sortheap and require temp tables.



Application Global Memory (app_ctl_heap_sz)
Agent Private Memory
Application Heap (applheapsz)
(drda_heap_sz)
Heap
Sort Heap (sortheap)
UDF Memory (udf_mem_sz)
Statement Heap (stmtheap)
Med
Workspace for dynamic SQL (when executing) and static SQL (during bind)
Default of 2,048 4k pages with range of 128 – 60,000 4k pages
Query Heap (query_heap_sz)
Client I/O Block (rqrioblk)



Application Global Memory (app_ctl_heap_sz)
Use to store each query in the agents private memory. SQLDA, statement, SQLCA, package name, creator, section number and consistency token
Memory for blocking cursors, cursor control blocks, and fully resolved SQLDA
(agent_stack_sz)
Application Heap (applheapsz)
DRDA Heap (drda_heap_sz)
(sortheap)
Default of 1,000 4k pages with range of 2 – 524,288 4k pages
Med
Query Heap (query_heap_sz)
Client I/O Block (rqrioblk)


Application Global Memory (app_ctl_heap_sz)
Agent Private Memory
Specifies size of the communications buffer between agents and remote applications in bytes
Agent Stack (agent_stack_sz)
Statistics Heap (stat_heap_sz)
Sort Heap (sortheap)
Determines size of IO block at database client when blocking cursor is opened, Default of 32,767 bytes should be ok
UDF Memory (udf_mem_sz)
Statement Heap (stmtheap)
High
Optimize for N rows can be used to control the number of rows returned per block
Client I/O Block (rqrioblk)

Monitoring

- IBM Self – Managing and Resource Tuning (SMART) Project
- Will make DBAs more productive and job easier
  - DBA will always be in-the-loop
- Compete better with SQL Server and to a lesser extent Oracle

Automatic Configuration

- Autoconfigure command can be used to set DBM and DB CFG parameters initially until the workload can be better defined

Syntax -- AUTOCONFIGURE [USING config-keyword value [{,config-keyword value}...]]

[APPLY {DB ONLY | DB AND DBM | NONE}]

config-keyword:

MEM_PERCENT, WORKLOAD_TYPE, NUM_STMTS, TPM, ADMIN_PRIORITY, IS_POPULATED

NUM_LOCAL_APPS, NUM_REMOTE_APPS, ISOLATION, BP_RESIZEABLE.

Automatic Configuration

- “db2 autoconfigure apply none” command can be used to see DB2 recommendations but not apply them.
- Refer to autoconfigure example output.

Monitoring

- In order to make sure that resources are being used efficiently and to ensure that business requirements are met, continuous monitoring must be practiced
- Use Snapshot Monitoring and Event Monitoring
- Third Party Vendor Tools

NUM_IOCLEANERS

- DB CFG Default 1 Range(0 –255)
  – Specifies number of asynchronous page cleaners for a database.
  – Write changed pages from Bufferpool to disk
  – Triggered by CHNGPGS_THRESH which specifies a percentage of used pages at which asynchronous page cleaners will start writing out pages
  – Set to number of CPUs

Page Cleaning Enhancements

- New registry variable in DB2 V8.1.4 changes page cleaning algorithm
- DB2_USE_ALTERNATE_PAGE_CLEANING registry variable
  - Page cleaners write pages proactively instead of waiting to be triggered

NUM_IOSERVERS

- DB CFG Default 3 Range(1 –255)
  - Used to specify the number of prefetchers that work on behalf of database agents to perform prefetch IO and asynchronous IO for utilities such as backup and restore.
  - Set to at least the number of physical disks available to increase opportunity for parallel IO

Agent Parameters

- Coordinator Agent – Each application has one which does work on its behalf and in a parallel environment distributes work to subagents
- Upon disconnect or detach from an instance the coordinating agent is freed and marked as idle if max number of pool agents not reached else it is terminated and storage freed if max number of pool agents reached
- DBM CFG parameter max_coordagents

Agent Parameters

- Maximum Number of Agents (maxagents) — specifies the maximum number of database manager agents, whether coordinating agents or subagents, available at any given time to accept application requests
- Can be used in resource constrained systems to limit memory usage

Agent Parameters

- Maximum Number of Active Applications (maxappls)
  - Specifies the maximum number of concurrent applications that can be connected to a database
  - When reached, an error is returned to the application and connection is not allowed
- Can be used to throttle applications in a resource constrained system

Agent Parameters

- Maximum Number of Concurrent Agents (maxcagents)
  - Specifies the max number of database manager coordinating agents that can be concurrently executing a database manager transaction
  - Does not limit the number of applications connected but limits the number of database manager agents that can be processed concurrently
- Can be used to throttle applications if resource constrained

Agent Parameters

- Initial Number of Agents in Pool (Num_initagents)
  - Specifies the initial number of idle agents that are created in the agent pool at DB2START
- By specifying a value, agents are available in the pool for initial requests and the overhead of repeated agent creation is avoided

Agent Parameters

- Agent Pool Size (num_poolagents)
  - Specifies how large the agent pool can get
  - Contains <u>subagents</u> and <u>idle agents</u>
  - Idle agents can be used as coordinating agents or subagents
  - If more agents created than this parameter they will be terminated when the current request is completed rather than returned to the pool

Conclusion

- Successful system tuning requires knowledge of:
  - DB2 processing
  - Available monitoring facilities
  - Instance Configuration Parameters
  - Database Configuration Parameters
  - Cause and Effect of parameters to processing
- *Available References

References

- Database Performance on AIX in DB2 UDB and Oracle Environments, SG24-5511-00
- URL: http://www.software.ibm.com/data/db2/udb

References

- SC09-4821, Administration Guide: Performance
- SC09-4822, Administration Guide: Planning
- SC09-4820, Administration Guide: Implementation
- SC09-4848, What’s New, DB2 UDB V8

REFERENCES

- DB2 UDB v8 Announcement, 202-14, http://www.ibmlink.ibm.com/usalets&parms=H_202-214
- DB2 Connect v8 Announcement, 202-215http://www.ibmlink.ibm.com/usalets&parms=H_202-215
- SC09-4828, Command Reference

DB2 Architecture Workshop

- Refer to your student workbook and examples for the DB2 Architecture Workshop. You have 1 hr to complete the workshop.

Module 7
DB2 Optimizer
Explain & Design Advisor

OBJECTIVES

- Upon completion of this module you will understand:
  - Optimization Classes and selection of classes based on type of environment (OLTP,WEB, Mixed, BI/DW)
  - How to use DB2 Explain to tune SQL
  - What the optimizer uses to determine access paths
  - How to use Design Advisor
- Types of predicates

Basic SQL Coding Rules

- Limit the amount of data returned
- Avoid “SELECT *”
- Use predicates with good selectivity
- Use Range delimiting and Index Sargable predicates wherever possible
- When joining tables, specify predicates that are indexed
- Analyze local, order by, and join predicates for index access (index only access, elimination of sorts, cartesian products.

TYPE-2 INDEXES

- Type-2 indexes improve performance by eliminating most next-key-share locks, as entries are marked deleted instead of physically deleted from the page. Type-2 indexes are required for online load, online reorganization, and MDC. A table cannot have a mix of type-1 and type-2 indexes. Tables can be migrated to type-2 indexes via index reorganization. Type-2 indexes let you create an index on a column that is longer than 255 bytes.

DB2 Optimizer

Characteristic	Range Delimiting	Index SARGable	Data SARGable	Residual
Reduce index I/O	YES	NO	NO	NO
Reduce data page I/O	YES	YES	NO	NO
Reduce number of rows passed internally	YES	YES	YES	NO
Reduce number of qualifying rows	YES	YES	YES	YES

DB2 Optimizer - Predicates

- Range delimiting (Index Manager)
  - Used to bracket an index scan
  - Provide start and stop keys for the index search

DB2 Optimizer

- Index Sargable
  - Not used to bracket a search
  - But are evaluated from the index
    - Columns involved in the predicates are part of the index
  - Evaluate by Index Manager

DB2 Optimizer -Predicates

- Data Sargable
  - Cannot be evaluated by index manager
  - Evaluated by Data Management Services
  - Typically require data access
- DMS will retrieve the columns to evaluate the predicate as well as any other to satisfy columns in the select list that could not be obtained via an index

DB2 Optimizer - Predicates

- Residual predicates require I/O beyond just accessing the base table
- Example: Quantified subqueries
  - ANY, ALL, IN, SOME, Long VARCHAR, LOBS
- Are evaluated by RDS and are the most costly of the 4 categories of predicates

DB2 Optimizer

Predicates	column c1	column c2	column c3	Comments
c1 = 1 and c2 = 2 and c3 = 3	Range delimiting (start-stop)	Range delimiting (start-stop)	Range delimiting (start-stop)	The equality predicates on all the columns of the index can be applied as start-stop keys.
c1 = 1 and c2 = 2 and c3 >= 3	Range delimiting (start-stop)	Range delimiting (start-stop)	Range delimiting (start)	Columns c1 and c2 are bound by equality predicates and the predicate on c3 is only applied as a start key.

DB2 Optimizer

Predicates	column c1	column c2	column c3	Comments
c1 >= 1 and c2 = 2	Range delimiting (start)	Range delimiting (start-stop)	not applicable	The leading column c1 has a ">=" predicate and can be used as a start key. The following column c2 has an equality predicate, and therefore can also be applied as a start-stop key.
c1 = 1 and c3 = 3	Range delimiting (start-stop)	not applicable	Index SARGable	The predicate on c3 can not be used as a start stop key since there is no predicate on c2. It can however be applied as an Index SARGable predicate.
c1 = 1 and c2 > 2 and c3 = 3	Range delimiting (start-stop)	Range delimiting (start)	Index SARGable	The predicate on c3 can not be applied as a start-stop predicate because the previous column has a ">" predicate. Had it been a ">=" instead, we would be able to use it as a start-stop key.
c1 = 1 and c2 <= 2 and c4 = 4	Range delimiting (start-stop)	Range delimiting (stop)	Data SARGable	Here the predicate on c2 is a "<=" predicate. It can be used as a stop key. The predicate on c4 can not be applied on the index and is applied as a Data SARGable predicate during the FETCH.
c2 = 2 and UDF_with_external_action(c4)	not applicable	Index SARGable	Residual	The leading column c1 does not have a predicate, and therefore the predicate on c2 can be applied as an Index SARGable predicate where the whole index is scanned. The predicate involving the user defined function with external action is applied as a Residual predicate.

DB2 Optimizer

Predicates	column c1	column c2	column c3	Comments
c1 = 1 or c2 = 2	Index SARGable	Index SARGable	not applicable	The presence of an OR does not allow us this multi-column index to be used as start-stop keys. This might have been possible had there been two indexes — one with a leading column on c1, and the other with a leading column on c2, and the DB2 optimizer chose an "index-ORing" plan. However, in this case the two predicates are treated as Index SARGable predicates.
c1 < 5 and (c2 = 2 or c3 = 3)	Range delimiting (stop)	Index SARGable	Index SARGable	Here the leading column c1 is exploited to stop the index scan using the predicate with a stop key. The OR predicate on c2 and c3 are applied as Index SARGable predicates.

DB2 Optimizer

Original predicate or query	Optimized predicates	Comments
c1 between 5 and 10	c1 >= 5 and c1 <= 10	The between predicates are rewritten into the equivalent range delimiting predicates so that they can be used internally as though the user specified the range delimiting predicates.
c1 not between 5 and 10	c1 < 5 or c1 > 10	The presence of the OR predicate does not allow the use of a start-stop key unless the DB2 optimizer chooses an index-ORing plan.

DB2 Optimizer

Original predicate or query	Optimized predicates	Comments
select * from t1 where EXISTS (select c1 from t2 where t1.c1 = t2.c1)	select t1.* from t1 EOJOIN t2 where t1.c1= t2.c1	The subquery may be transformed into a join — internally a special join called an "early out join" is used so that we do not multiply the rows from t1 if there are multiple rows having the same value in the join column in t2
select * from t1 where t1.c1 IN (select c1 from t2)	select t1.* from t1 EOJOIN t2 where t1.c1= t2.c1	This is similar to the transformation for the EXISTS predicate example above.
c1 like 'abc%'	c1 >= 'abc X X X ' and c1 <= 'abc Y Y Y'	If we have c1 as the leading column of an index, DB2 generates these predicate so that they can be applied as range delimiting start-stop predicates. Here the characters X and Y are symbolical of the lowest and highest collating character.
c1 like 'abc%def'	c1 >= 'abc X X X ' and c1 <= 'abc Y Y Y' and c1 like 'abc%def'	This is like the previous case except that we have to additionally apply the original predicate as a, index SARGable predicate so as to get the match for the characters 'def' correctly.

DB2 Optimizer - Dynamic SQL

- Flexibility
- Most packaged applications and web based applications use dynamic SQL
- Provides flexibility and uses the most current statistics

DB2 Optimizer - Dynamic SQL

- Caution!
  - Growth in data
  - Number of indexes available
- Can result in changing access paths

DB2 Optimizer - Inputs

Buffer pool size (npages)	To determine how much of the buffer pool may be available for tables/indexes involved.
SORTHEAP DB CFG parameter	To determine if a piped sort can be used.
LOCKLIST	To determine amount of memory available for storing locks for this access plan.
CPU Speed	Speed of CPUs available.
PREFETCHSIZE	To determine I/O costs.
Value of INTRA_PARALLEL DBM CFG Parameter	To determine if parallelism may be used.
Type of table space and number of containers	To determine I/O costs and degree of I/O parallelism.
SHEAPTHRES	Determine maximum amount of shared SORTHEAP available.
DISK Speed	To estimate I/O costs.
Degree of clustering	To determine effectiveness of prefetching and to determine how clustered data is.
Indexes Available	To determine if index access cost.
DFT_DEGREE	Default degree of parallelism.
AVG_APPLS	To determine amount of buffer pool space available for a query.
MAXLOCKS	Percent of LOCKLIST used by a single application before lock escalation occurs.
LOCKLIST	Size of memory area reserved for locks.
DFT_QUERYOPT	The default optimization class to be used.
STMTHEAP	Size can effect amount of optimization conducted.
COMM_BANDWITH	Used for partitioned databases.
MAX_QUERYDEGREE	Maximum number of subagents to be used if intra_parallel enabled.

DB2 Optimizer – Distribution Statistics

RUNSTATS

- See Notes

Optimization Class

- The optimization class (dft_queryopt) Database Configuration (DB CFG) parameter specifies how much optimization the query optimizer should use for queries
  - by default it is set to 5
- There are 7 optimization classes available for use
- On the next slide we'll discuss them in detail

Optimization Class

0 -- provides for minimal query optimization

1 -- small amount of query optimization

2 -- slight amount of query optimization

3 -- moderate amount of query optimization

4 -- reserved for future use

5 -- significant query optimization with heuristics to limit the effort expended on selecting an access plan

6 -- reserved for future use

7 -- significant query optimization

9 -- maximal query optimization

Optimization Class

- This page intentionally left blank

Optimization Class

- Use the SET CURRENT QUERY OPTIMIZATION command in dynamic SQL to set the special register to the desired optimization class
- Use explain to verify if an improved access plan is generated
- Test and verify new plan

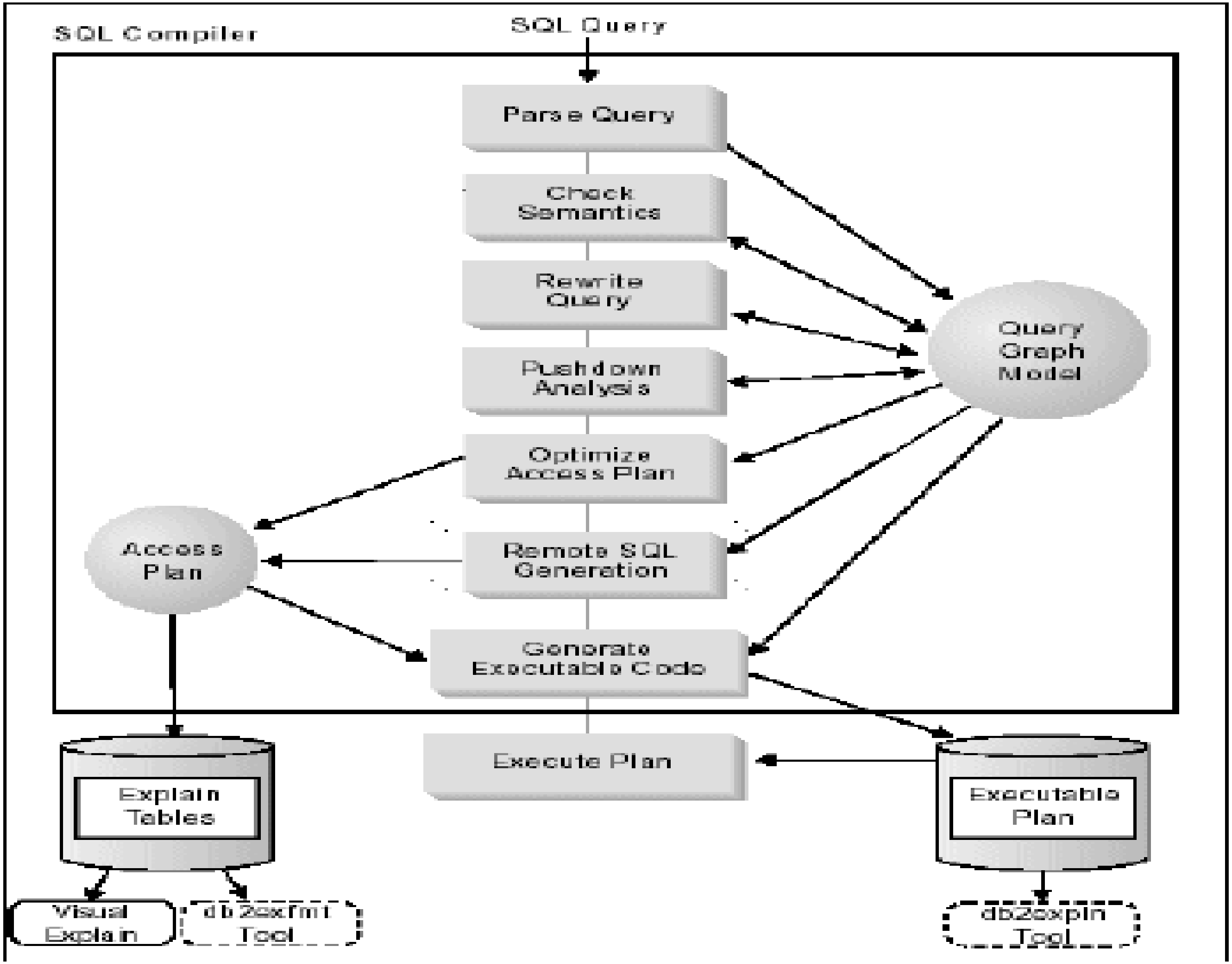
Optimization Class

- Set dft_queryopt through trial and error process with your applications during application development process
- For OLTP and Web-based applications, set dft_queryopt to 1 or 3 (keep restrictions in mind)
  - OLTP/Web applications should be only returning a row or few rows with index only or indexed access
  - Few reads involved
  - Not very complex SQL
- For Mixed environments set dft_queryopt to 5
  - Usually a mixture of simple and complex SQL

Optimization Class

- For BI/DW environments set dft_queryopt to 7 or 9
  - Evaluate differences between optimization class 7 and 9 in your environment, with your applications
  - Can be evaluated easily using Visual Explain

SQL Compiler



Explain Tables

Table Name	Description
EXPLAIN_ARGUMENT	Contains information about the unique characteristics for each individual operator, if any.
EXPLAIN_INSTANCE	The main control table for all Explain information. Each row of data in the Explain tables is explicitly linked to one unique row in this table. Basic information about he source of the SQL statements being explained and environment information is kept in this table.
EXPLAIN_OBJECT	Identifies those data objects required by the access plan generated to satisfy the SQL statement.
EXPLAIN_OPERATOR	Contains all the operators needed to satisfy the SQL statement by the SQL compiler.
EXPLAIN_PREDICATE	Identifies the predicates that are applied by a specific operator.
EXPLAIN_STATEMENT	Contains the text of the SQL statement as it exists for the different levels of explain information. The original SQL statement as entered by the user is stored in this table with the version used by the optimizer to choose an access plan. When an explain snapshot is requested, additional explain information is recorded to describe the access plan selected by the SQL optimizer. This information is stored in the SNAPSHOT column of the EXPLAIN_STATEMENT table in the format required by Visual Explain. This format is not usable by other applications.
EXPLAIN_STREAM	Represents the input and output data streams between individual operators and data objects. The data objects themselves are represented in the EXPLAIN_OBJECT table. The operators involved in a data stream are represented in the EXPLAIN_OPERATOR table.
ADVISE_WORKLOAD	Allows users to describe a workload to the database. Each row in the table represents an SQL statement in the workload and is described by an associated frequency. The db2advis tool uses this table to collect and store work and information.
ADVISE_INDEX	This table stores information about recommended indexes. The table can be populated by the SQL compiler, the db2advis utility or a user. This table is used in two ways: • To get recommended indexes • To evaluate indexes based on input about proposed indexes.

DB2 Explain

- DB2 offers three types of Explains
- dynexpln which is used for explaining dynamic SQL is still available but deprecated
- db2exfmt is a “complete explain” and produces explain information in text format
- DB2 Visual Explain
  - DB2 Visual Explain is a powerful tool which visually displays access paths and statistics associated with the SQL statement being analyzed. It can be launched from the Control Center
  - DB2 Visual Explain has a Tutorial available

Optimization Parameters

Optimization Parameters

P3600 - DB2 - SAMPLE

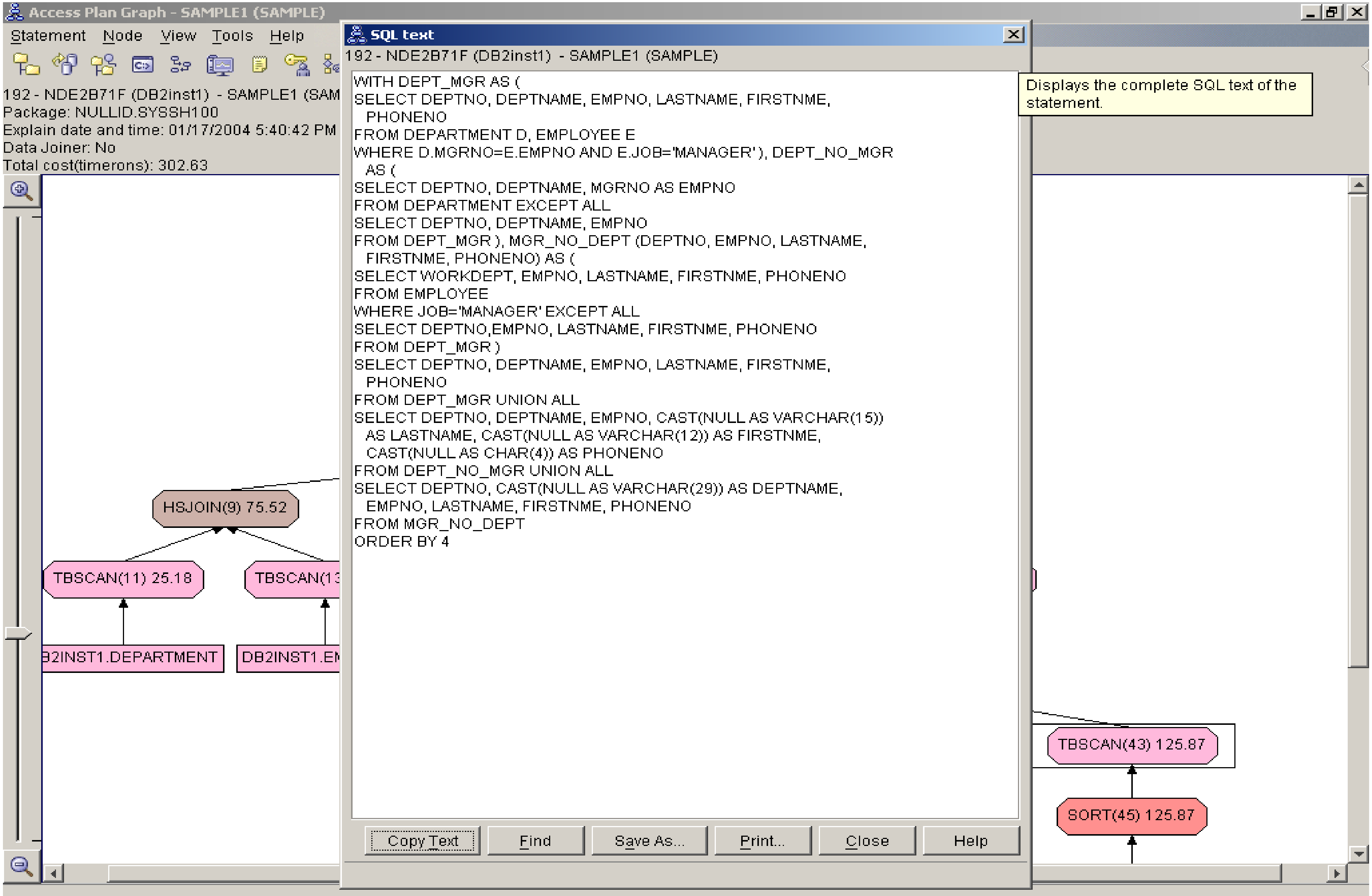
Explain date and time: 07/26/2000 8:06:22 AM
Current date and time: 07/26/2000 8:06:45 AM

Configuration parameters	Explained	Current
AVG_APPLS	1	1
SORTHEAP	256	256
LOCKLIST	50	50
MAXLOCKS	22	22
NUM_FREQVALUES		10
NUM_QUANTILES		20
LOCKS_AVAIL	1243	
DBHEAP	600	600
CPUSPEED	9.84e-007	9.84E-7
BUFFPAGE	256	250

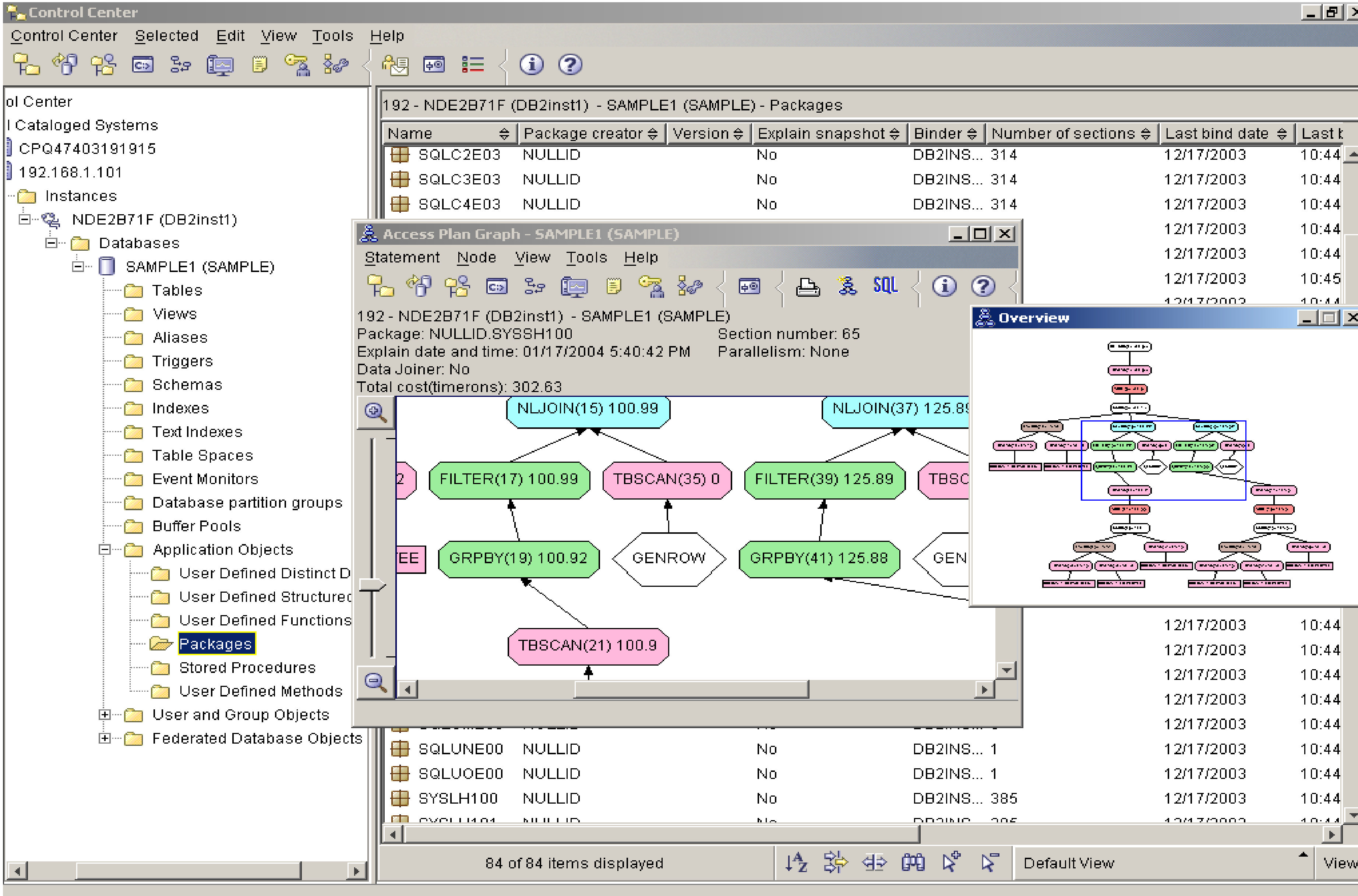
Bind options	Explained	Current
QUERYOPT	5	5
ISOLATION	Cursor stability	
BLOCK	Block all	

Save As... Print... Close Help

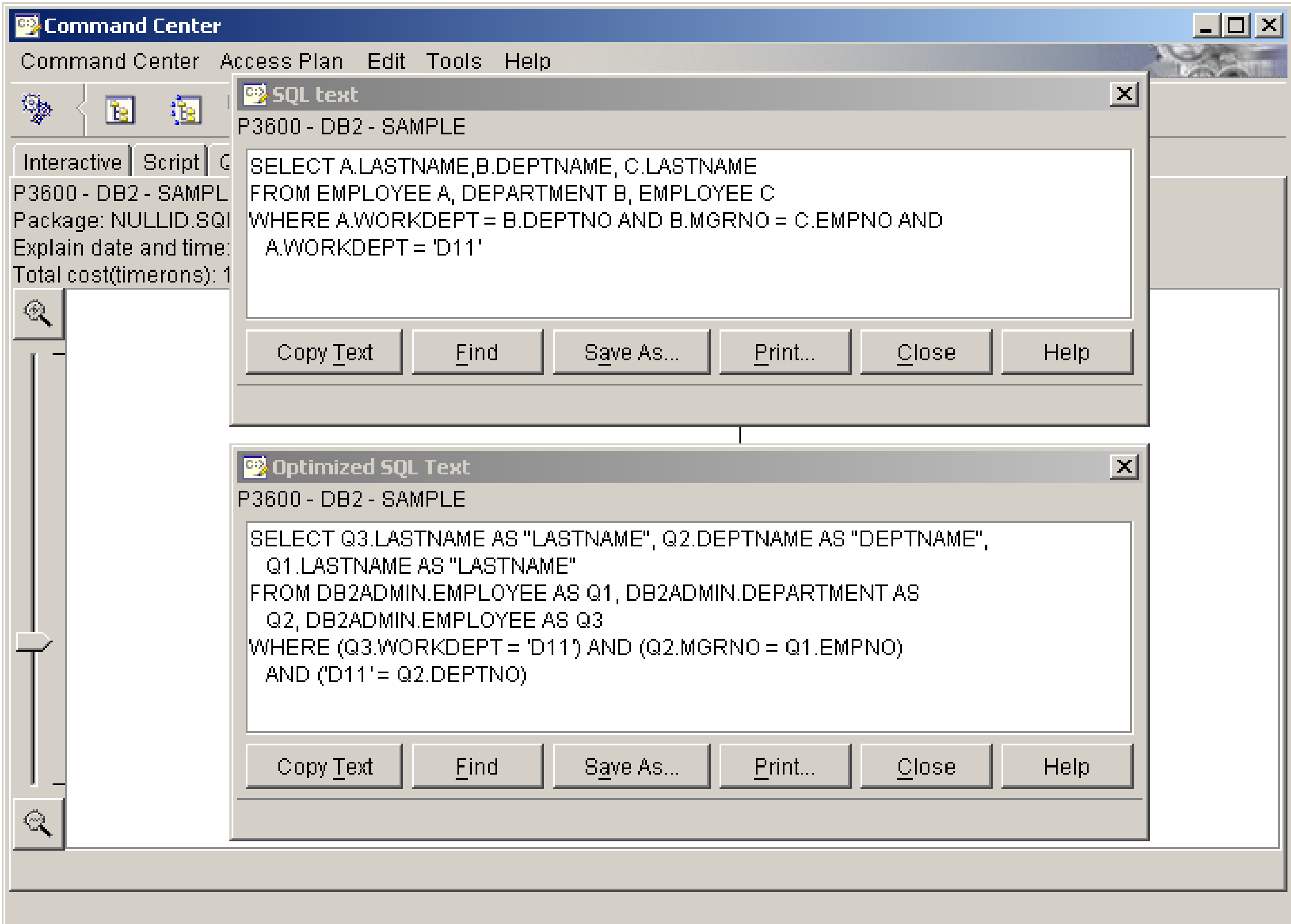
Visual Explain



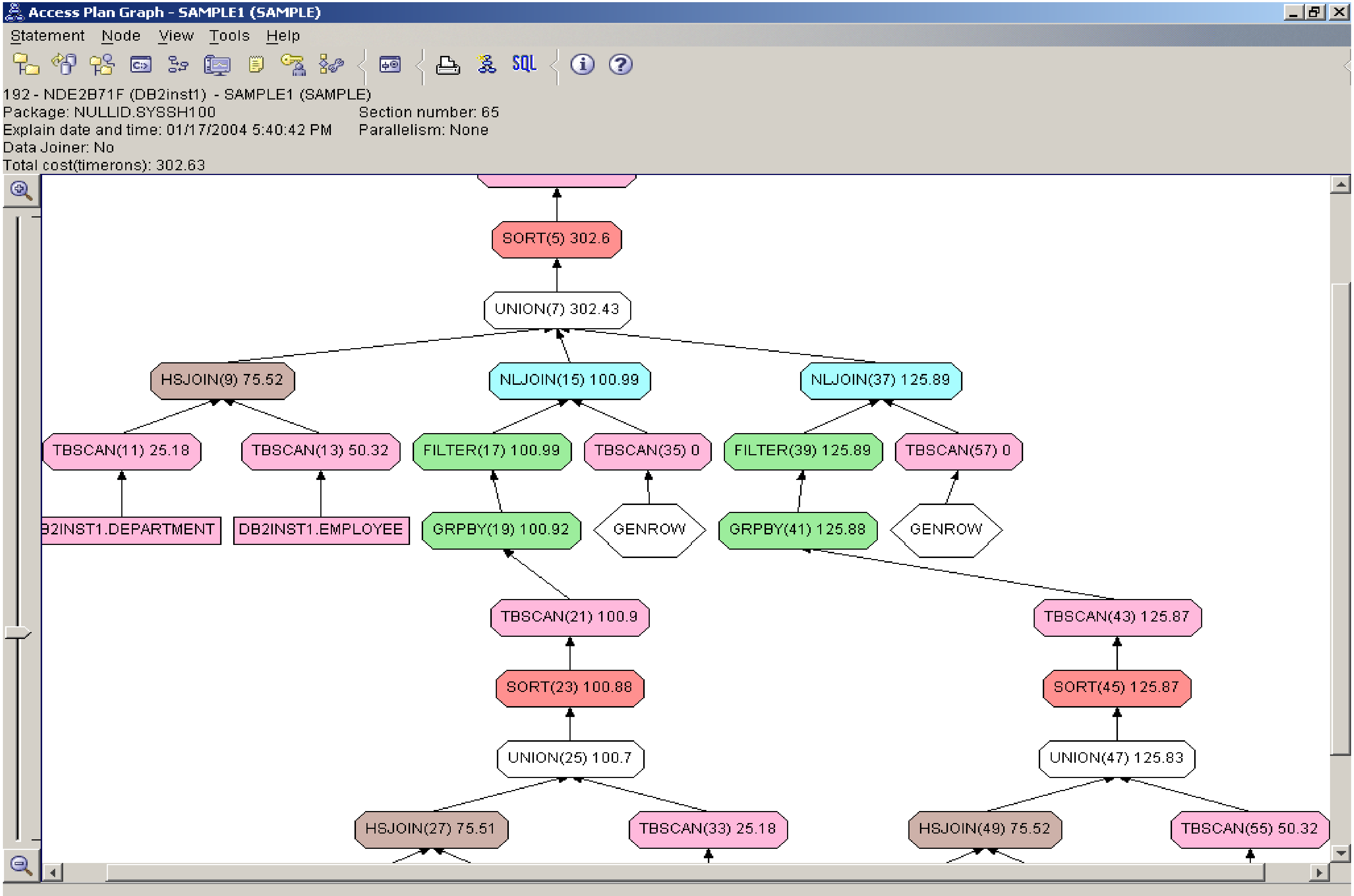
Visual Explain



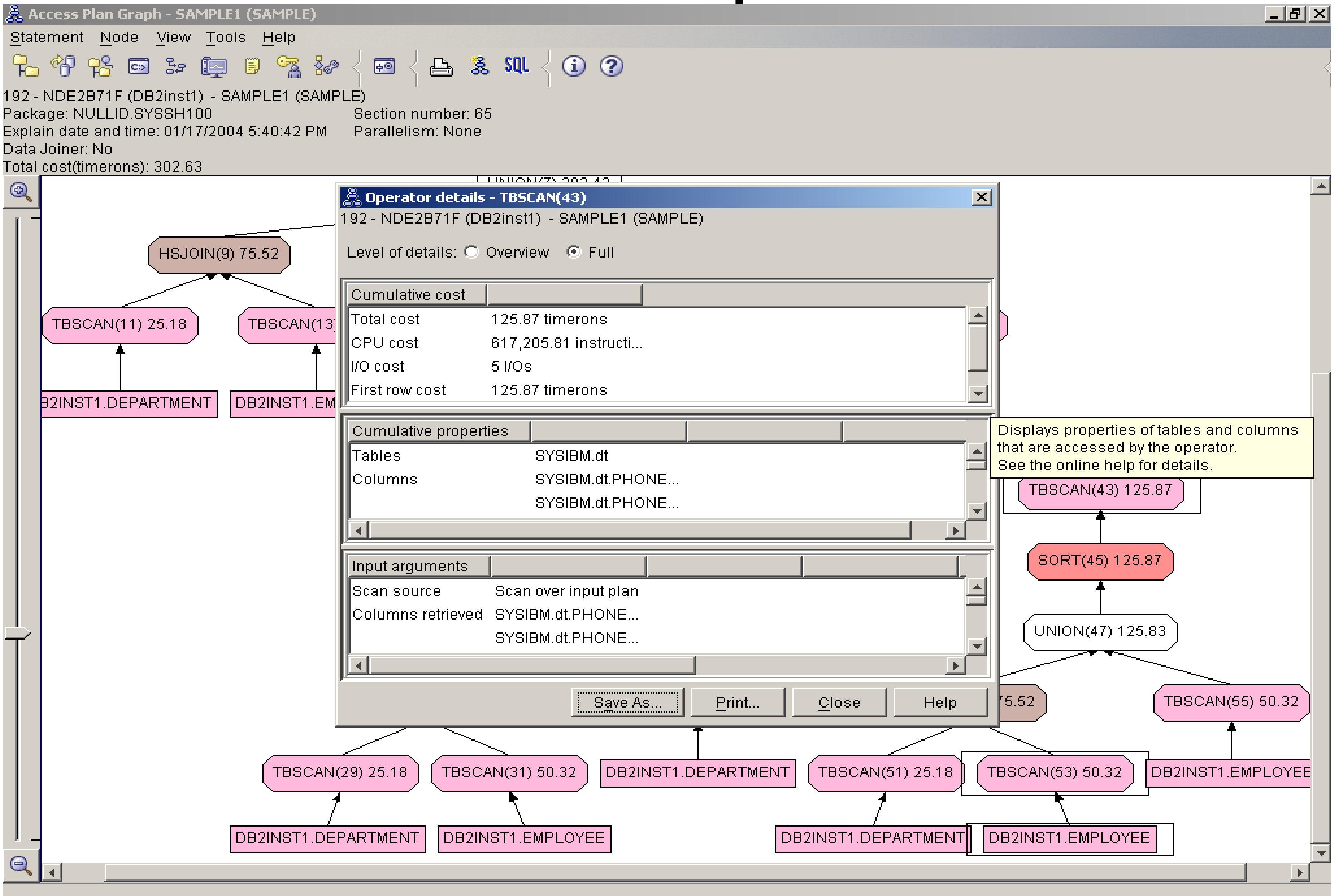
Optimized SQL Text



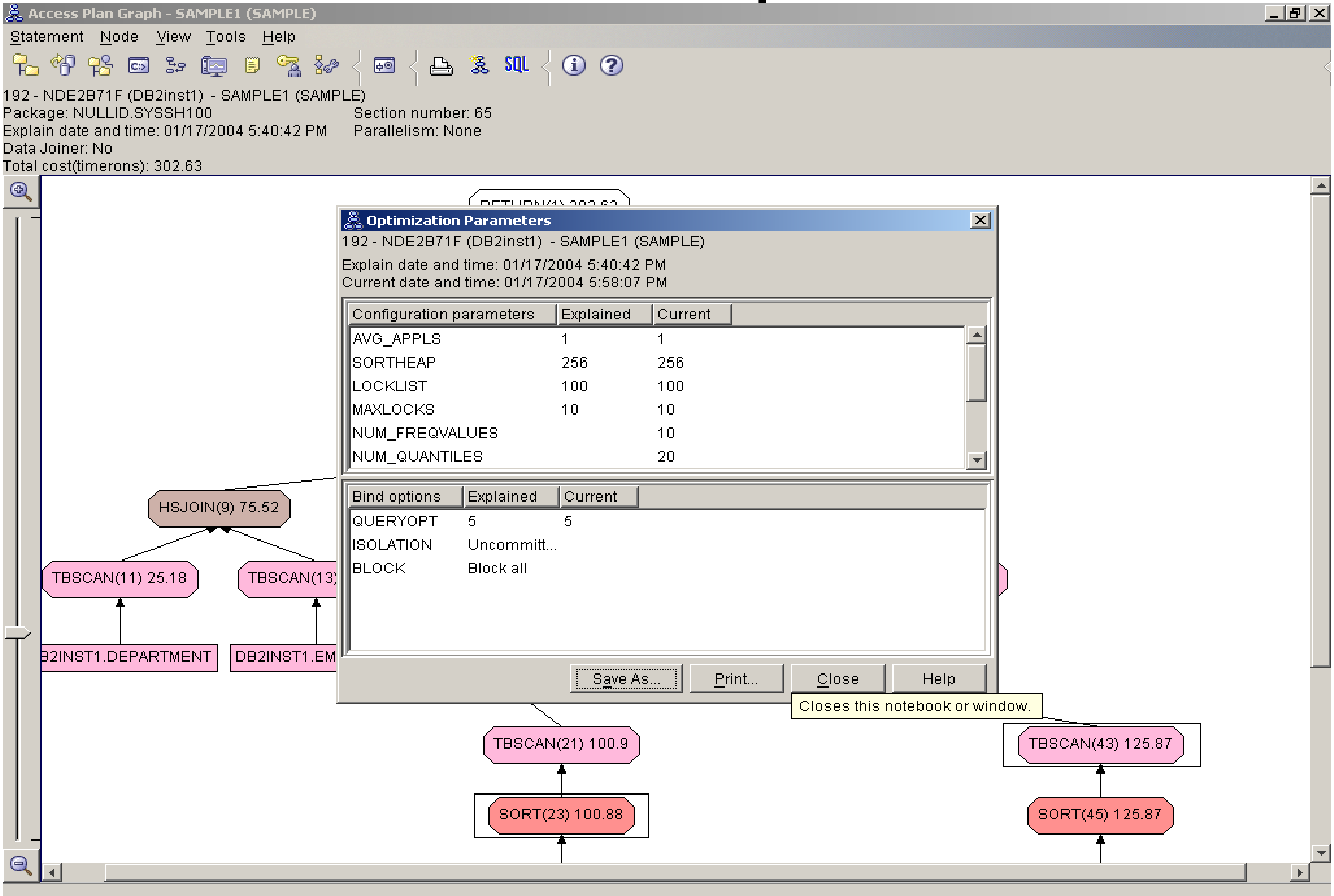
Visual Explain



Visual Explain



Visual Explain



db2exfmt

- Formats rows in explain tables

db2exfmt –d *dbname* -e *schema* -f *O* –n *name* –s *schema* –o *outfile* -# *sectnbr* -h

-d name of database

-e schema of explain table

-s schema of table being explained

-n source name

-o outfile: name of output file

– Wild cards can be used: % and _

db2exfmt

- See below example
- Refer to example 41

db2exfmt

- See below example

Line Command Explains

- db2expln
  - Use for static when not explained
  - Builds output from syssection
- Dynexpln (deprecated in V8)
  - Use for a file of SQL or interactive “quickie”
- db2exfmt
  - To print contents of explain tables

db2expln Command

- DB2 describes the access plan selection for *dynamic and static SQL statements stored in the DB2 catalog
- See below

db2expln

- db2expln

db2expln

- db2expln

db2expln Sample

```
DB2 Universal Database Version 7.1, 5622-044 (c) Copyright IBM Corp.
   1991, 2000
Licensed Material - Program Property of IBM
IBM DATABASE 2 SQL Explain Tool
******************** PACKAGE ***************************************
Package Name = NULLID.SQLUJD03
   Prep Date = 1999/10/19
   Prep Time = 00:00:00
   Bind Timestamp = 2000-05-09-13.41.35.870000

   Isolation Level          = Cursor Stability
   Blocking                 = Block All Cursors
   Query Optimization Class = 5
   Partition Parallel       = No
   Intra-Partition Parallel = No
   Function Path            = "SYSIBM", "SYSFUN", "DB2ADMIN"

-------------------- SECTION ---------------------------------------
```

db2expln Sample

```
-------------------- SECTION ---------------------------------------
Section = 12
SQL Statement:
 SELECT COUNT(NAME) INTO :H00037
 FROM SYSIBM.SYSCOLUMNS
 WHERE (TBNAME = :H00029 AND TBCREATOR = :H00030 ) AND(GENERATED != '')
Estimated Cost        = 149
Estimated Cardinality = 1
 Access Table Name = SYSIBM.SYSCOLUMNS  ID = 0,3
|  #Columns = 3
|  Index Scan:  Name = SYSIBM.IBM01  ID = 1
|  |  Index Columns:
|  |  |  1: TBCREATOR (Ascending)
|  |  |  2: TBNAME (Ascending)
|  |  |  3: NAME (Ascending)
|  |  #Key Columns = 2
|  |  |  Start Key: Inclusive Value
|  |  |  |  1: ?
|  |  |  |  2: ?
```

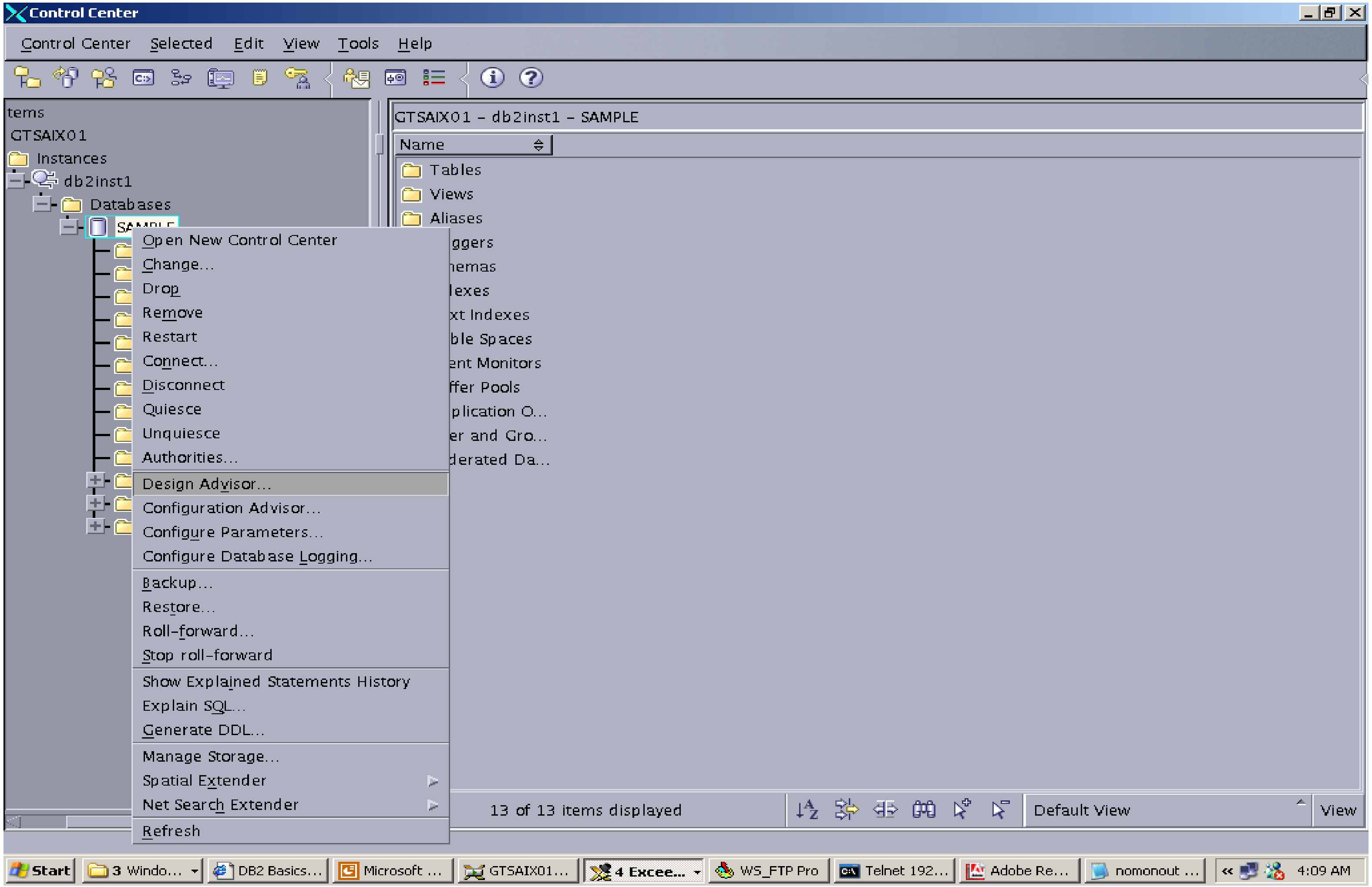
db2expln Sample

```
|  |  |  Stop Key: Inclusive Value
|  |  |  |  1: ?
|  |  |  |  2: ?
|  |  Data Prefetch: None
|  |  Index Prefetch: None
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
|  Sargable Predicate(s)
|  |  #Predicates = 1
|  Predicate Aggregation
|  |  Column Function(s)
Aggregation Completion
|  Column Function(s)
Return Data to Application
|  #Columns = 1
End of section
```

Design Advisor

- Design Advisor is a tool used to evaluate SQL statements and workloads and to recommend indexes to improve performance
  - Can accept single SQL statement or a workload as input
  - Ouputs create index statements that can be used to create recommended indexes
  - Should not blindly apply indexes especially for single SQL statements
  - Take entire workload and business priorities into consideration
- Future enhancements will include recommendations for MQTs and MDC

Design Advisor



Design Advisor

db2advis -d gunprd -I wildsortsql.txt > wildsqlixadvout.txt

execution started at timestamp 2002-08-12-10.25.44.141157

found [1] SQL statements from the input file

Calculating initial cost (without recommmended indexes) [23866.660156] timerons

Initial set of proposed indexes is ready.

Found maximum set of [1] recommended indexes

Cost of workload with all indexes included [75.079346] timerons

total disk space needed for initial set [4.747] MB

total disk space constrained to   [-1.000] MB

1 indexes in current solution

[23866.6602] timerons (without indexes)

[75.0793] timerons (with current solution)

[%99.69] improvement

Trying variations of the solution set.--

-- execution finished at timestamp 2002-08-12-10.25.45.932376--

-- LIST OF RECOMMENDED INDEXES

-- ===========================

-- index[1],   4.747MB

CREATE INDEX WIZ1 ON "PUSER   "."T_FILE" ("FILE_STATUS_NUM" DESC) ;

-- ===========================--

Design Workload tool is finished.

Automatic Summary Tables (ASTs)

- ASTs are a special kind of Materialized Query Table (New in V8.1).
- Powerful way to improve response time for complex queries of the following type:
  - Aggregated data over one or more dimensions
  - Joins and aggregated data over a group of tables
  - Data from a commonly accessed subset of

Automatic Summary Tables (ASTs)

- Repartitioned data from a table, or part of a table, in a partitioned database environment

Automatic Summary Tables (ASTs)

- Can provide <u>drastic</u> performance improvements
  - An example is overnight queries now run in minutes
  - DB2 optimizer may determine that the query may run more efficiently against a summary table than the base table(s)
    - Accomplished via DB2 Optimizer query rewrite option

Automatic Summary Tables (ASTs)

- How are they created?
  - CREATE TABLE with SUMMARY keyword or if SUMMARY keyword omitted and AS followed by fullselect, DB2 recognizes this as a summary table

Automatic Summary Tables (ASTs)

- Example:

```
CREATE SUMMARY TABLE CLASS_SKU
 AS (SELECT STOR_SKU,
            SUM(QTY) AS QUANTITY,
            SUM(AMT) AS AMOUNT
FROM SKU_MASTER GROUP BY
   STOR_SKU )
DATA INITIALLY DEFERRED
REFRESH DEFERRED;
```

Automatic Summary Tables (ASTs)

- The fullselect of a summary table definition cannot contain the following:
  - References to a view
  - Expressions that are a reference type or DATALINK (or a distinct type based on these types)
  - Functions that have an external action
  - Functions that depend on physical characteristics (NODENUMBER, PARTITION)

Automatic Summary Tables (ASTs)

- Table or view references to system objects (explain tables included)
- Cannot use IMPORT or LOAD utility
- Cannot create a unique index
- Limited use of ALTER statement

Automatic Summary Tables (ASTs)

- Tables created with the REFRESH IMMEDIATE option are subject to the following restrictions:
  - The fullselect in the summary table definition must be a sub-select and cannot include:
    - functions that are not deterministic
    - scalar fullselects
    - predicates with fullselects
    - special registers

Automatic Summary Tables (ASTs)

- A Group By clause must be included in the sub-select
- The select list must have a COUNT(*) function (or COUNT_BIG) and no DISTINCT
- Only SUM (if not nullable columns), COUNT, or COUNT_BIG column functions are allowed in the select list (without DISTINCT) and other SELECT list items must be included in the GROUP BY clause.

Automatic Summary Tables (ASTs)

- ALL Group By items must be included in the Select list
- No grouping sets are allowed (including CUBE and ROLLUP) or GROUPING on constants
- A HAVING clause is not allowed

USER-Maintained Materialized Query Tables (MQTs)

- V8.1 features that ALLOWS USERS to migrate from ORACLE, which already has this capability.
- Useful for users already generating summary data through other means, such as a nightly batch job.
- With a user-maintained summary table, users can load this summarized data.
- USER-maintained summary tables are no different from system summary tables with the exception that the creation and loading of the summary table is under user control.

USER-Maintained MQTs

- DB2 does not refresh the data
  - Responsibility of the user to update data as needed
  - DB2 will assume that data in these tables is correct and may produce incorrect results if the data is not valid.
- Created by specifying the MAINTAINED BY USER option of the CREATE SUMMARY TABLE statement.

USER-Maintained MQTs

- Example

```
CREATE SUMMARY TABLE UMST_SALES AS
    (Select
        Region
        COUNT(*) AS SKU_COUNT,
        SUM(RSALES) AS REG_SALES
        SUM(TSALES) AS SORT_TOT
    From SALES GROUP BY REGION
        )
    DATA INTIALLY DEFERRED REFRESH
    DEFERRED MAINTAINED BY USER;
```

Automatic Summary Tables

- Setting the register to NONE will prevent any summary tables from being used during creation.
  - Set back on if you want to take advantage of the newly created summary table
- This special register can contain one of four values
  - ALL-specifies that all possible types of maintained tables controlled by this special register, now and in the future, are to be considered when optimizing the processing of dynamic SQL queries.

Automatic Summary Tables

- LOADING the USER-MAINTAINED SUMMARY TABLE

```
INSERT INTO UMST_SALES
    Select *  From
    (Select REGION, Count(*),
    SUM(RSALES), SUM (CORP_TOT)
    From SALES
    GROUP by REGION
    ) AST;
```

Automatic Summary Tables

- NONE- specifies that none of the object types of maintained tables controlled by this special register, now and in the future, are to be considered when optimizing the processing of dynamic SQL queries.

Automatic Summary Tables

- System- specifies that system maintained refresh-deferred MQTs can be considered to optimize the processing of dynamic SQL queries (Immediate MQTs are always available)
- User- specifies that user-maintained refresh-deferred MQTs can be considered to optimize the processing of dynamic SQL queries.

Automatic Summary Tables

- CURRENT REFRESH AGE special REGISTER
  - Specifies a timestamp duration value with a data type of DECIMAL (20,6)
  - This duration is the maximum duration, since a REFRESH TABLE statement has been processd on a deferred refresh summary table.
  - It determines if a summary table can be used to optimize the processing of a query

Automatic Summary Tables (ASTs)

- ASTs defined as REFRESH IMMEDIATE are immediately updated as inserts, updates, and deletes occur on the base tables
  - Changes are cascaded to AST
- ASTs defined as REFRESH DEFERRED are refreshed as a result of the REFRESH TABLE statement being issued
  - In V8.1 staging tables are used to accomplish a delta refresh of the table

Automatic Summary Tables (ASTs)

- Staging tables are created as part of the CREATE TABLE statement with PROPAGATION IMMEDIATE specified
- The staging tables contain some “control columns” to assist DB2 in managing the deltas

Automatic Summary Tables

- The keyword ANY is shorthand for 99999999999999.
- Initial value of CURRENT REFRESH AGE is 0.
- <u>MUST SET</u> to <u>ANY </u>or 99999999999999 for DB2 to be able for DB2 to consider using <u>deferred refresh summary tables</u>
  - SET CURRENT REFRESH AGE ANY

Automatic Summary Tables

- REFRESHED DEFERRED
  - Data in the table is refreshed when a REFRESH TABLE statement is executed.
  - The data in the summary table reflects the results of the query at the time the REFRESH TABLE statement is executed.

Automatic Summary Tables

- REFRESH IMMEDIATE
  - Means that the changes made to the underlying tables as part of a DELETE, INSERT, or UPDATE statement are <u>cascaded to the summary table immediately.</u>

Automatic Summary Tables

- Additional CREATE TABLE OPTIONS
  - ENABLE QUERY OPTIMIZATION
  - Means table can be used
  - DISABLE QUERY OPTIMIZATION
  - Means the table (MQT) will not be used for query optimization. But, THE TABLE CAN STILL BE QUERIED DIRECTLY.
  - Other Rules

Materialized Query Table (MQT)

- An MQT is a table whose definition is based on the result of a query, and whose data is in the form of precomputed results that are taken from one or more tables on which the MQT definition is based.
  - The definition of an MQT contain joins, functions, and other SQL elements <u>that are not allowed in ASTS!</u>

Materialized Query Table (MQT)

- MQT Example:

CREATE SUMMARY TABLE SMQT_CGS

As (SELECT RAWM,

DIRECT_COSTS, OVHD,

REGION R where C

region= R.region)

Data initially deferred REFRESH DEFERRED;

MQTs

- LIST TABLES command can be used to query the system catalog to produce a list of summary tables. The value S in the type column indicates that this table is a summary table.

AST Summary

- Restrictions
- Not Allowed:
  - Cannot create unique index
  - Limited ALTER CAPABILITY
  - Cannot insert, update or delete unless it is a user-maintained query
- Allowed:
- Create indexes
  - Use RUNSTATS
  - Use Explain information to see if summary table is used

Replicated MQT Tables in DPF

- Replicated MQTs improve performance of frequent executed joins in a partitioned database environment by allowing the database to manage precomputed values if the table data
- Designate Replicated MQT on the CREATE TABLE command using the REPLICATED keyword
- Once created, run RUNSTATS on replicated table
- Replicated MQTs improve performance in DPF environment as the data does not have to be moved across the network to each database partition.
- Replicated tables combined with collocated joins are a powerful combination.

AST and MQT Summary

- ASTs and MQTs are very powerful mechanisms to improve query performance in DW environments.
- Many queries can be satisfied by MQTs resulting in improved performance.
- Summary tables are usually much smaller then the base (fact) tables
- The optimizer can generally elect to use summary tables if the following conditions exist
  - The optimizer chooses to do so via query rewrite
  - Setting of the current REFRESH AGE special register
  - Zero (0) means only summary tables defined with refresh immediate may be used to optimize the processing of a query
  - Any or 9(14) means either summary table with refresh deferred or refresh immediate may be used

User-Maintained MQT Summary

- The Current Maintained Table Types for Optimization Special Register, must be set to USER to enable user-maintained refresh deferred MQTs to be considered for optimizing the processing of dynamic SQL queries.
  - Note Current Refresh AGE must be set to ANY else if it is 0, DB2 will only consider summary tables that are automatically maintained by the system.

Replicated MQTs Summary

- Replicated MQTs combined with collocated joins are a powerful combination to assist you in optimizing the performance of your Data Warehouse.

CLUSTERING

- Every table should have a clustering index specified
  - Clustering keeps index and data pages entries in clustering order.
  - Improves insert performance and prefetching of data as requested data is in order.
  - Over time clustering degrades as a result of inserts
  - Only one clustering index is allowed.
  - REORGCHK showed be run on a regular basis to identify candidates for reorganization.

SQL Tuning Workshop

- Refer to your student workbook and examples for the SQL Tuning Workshop. You have 1 hr to complete the workshop.

Module 8
Health Monitor and Health Center

Objectives

- Upon completion of this module you should understand:
  - Difference between point and exception monitoring
  - Health center and Health monitor operation
  - Health Center components
  - Health Indicators
  - How to setup up alarms and exception actions

Types of Monitoring

- Point Based
  - Visual “Eyes On”
  - Visual Cues and Alerts
  - Dedicated resource required
  - Real time monitoring
  - Drill down capability
- Exception Based
  - Server side lightweight agent
  - Alerts based on predefined or default thresholds
  - Emails and text messages
  - Visual cues/alerts

Health Monitor

- The health monitor is a lightweight server-side agent implemented in the DAS process
- The Health Center interfaces with the Health Monitor and can be used to configure the Health Monitor
- The Health Monitor implements exception based monitoring
  - Can operate 24x7
  - Can generate alerts and actions based on predefined or default threshold breaches
  - Can generate emails and send text messages to pagers

Health Monitor

- Default instance level health_mon switch is off by default
- Must be enabled by issuing “db2 update dbm cfg using health_mon on”

Health Monitor

- Is not displayed by “db2 get dbm monitor switches” or “db2 get monitor switches”
- Issue $ db2 get dbm cfg | grep health*

Monitor health of instance and databases (HEALTH_MON) = ON

to see if switch is enabled or not.

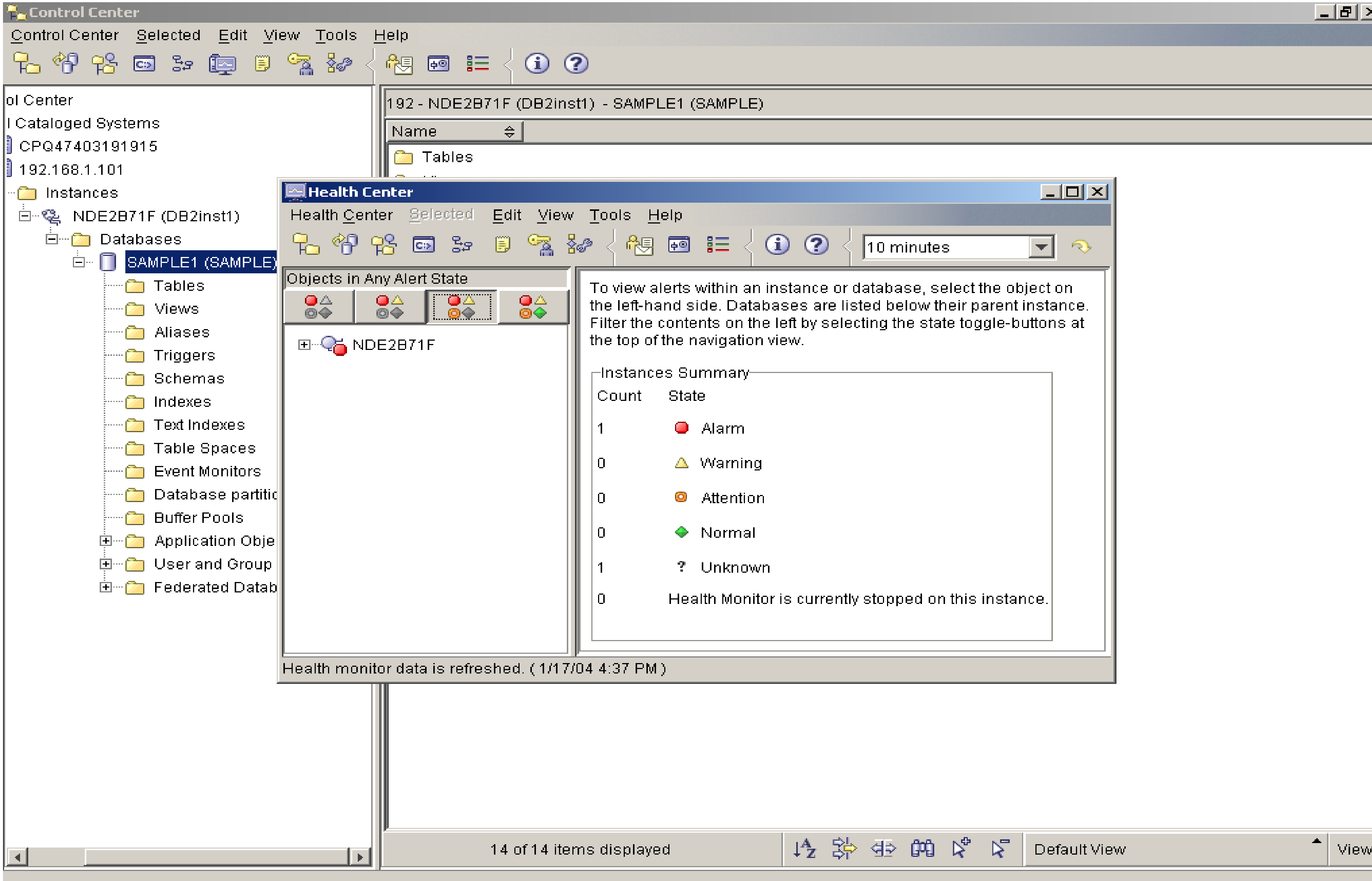
Health Monitor Administration

- Refer to Example 19 for an example of a Health Monitor alert email and associated recommendations
- Can ignore, review recommendations or take action
- “db2 get recommendations for health indicator <health-indicator-short-name>”
- “db2 get description for health indicator <health-indicator-short-name>”

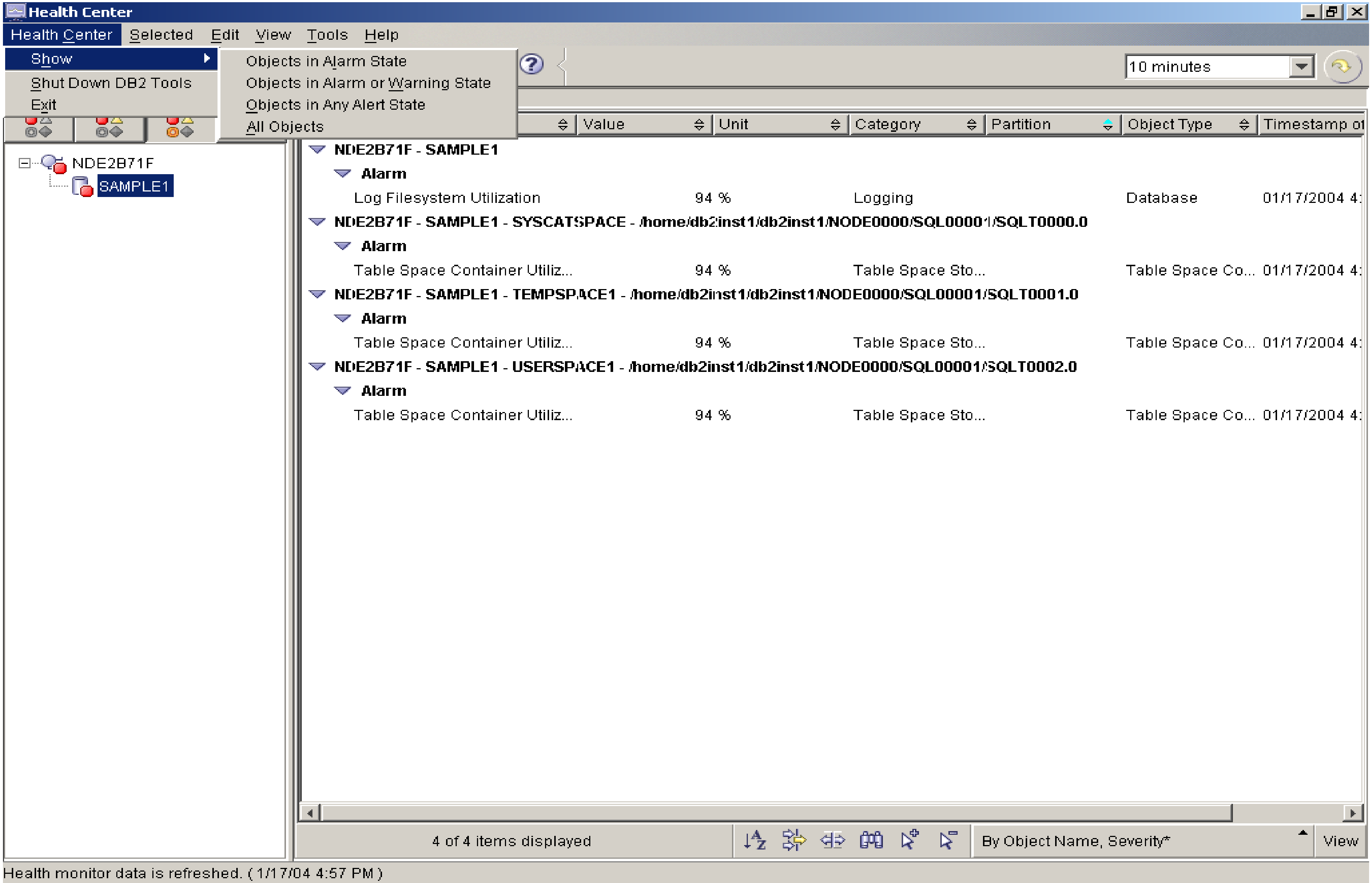
Health Center

- Launched from the Control Center
- Presents instances and databases using in a tree like format
- Database status is reported by Health Monitor and beacons display green, amber or red depending on the severity of the problem
- DBM and DB parameters can be monitored and recommendations provided for parameter setting changes based on recommendations provided

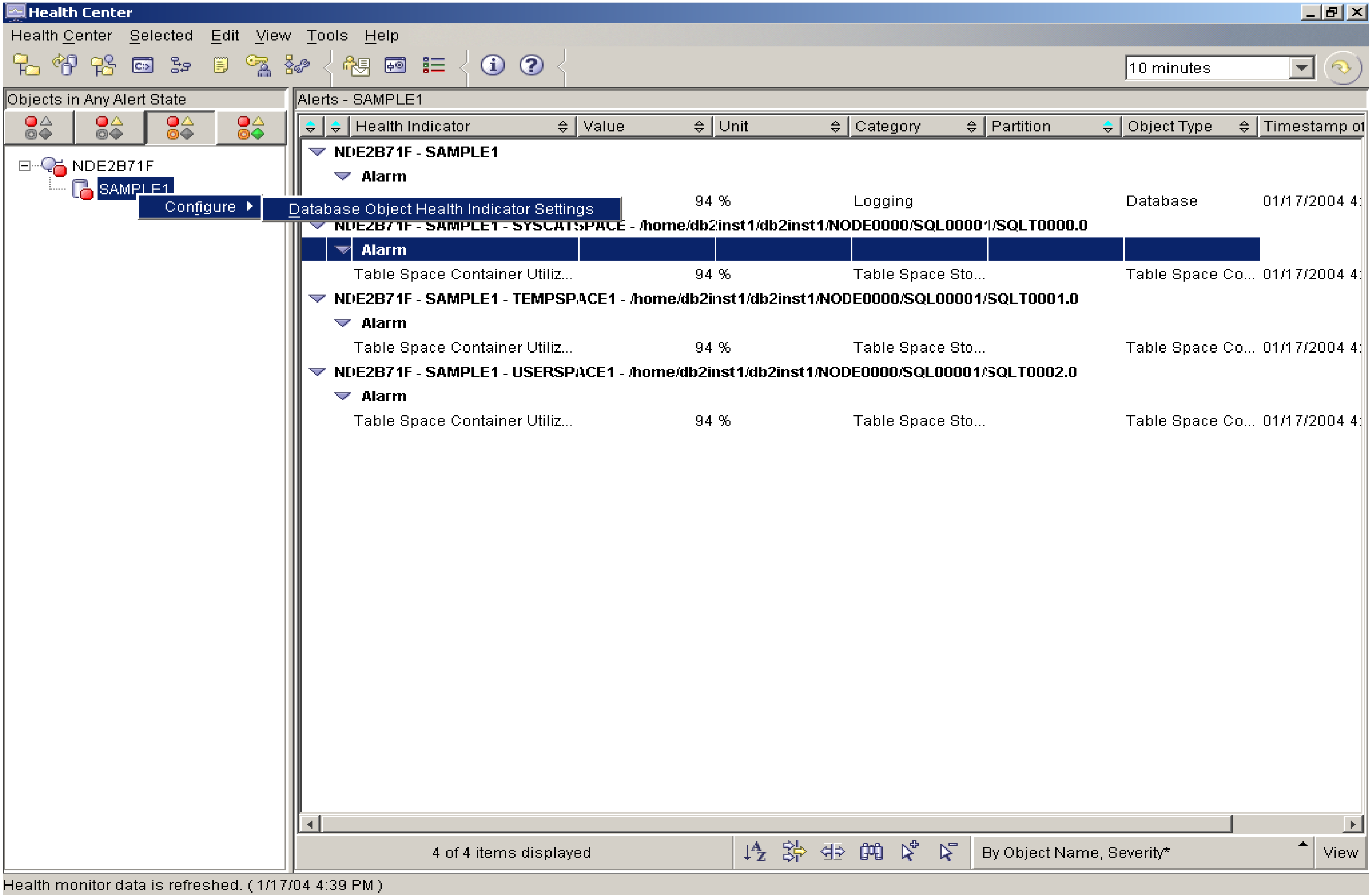
Health Center



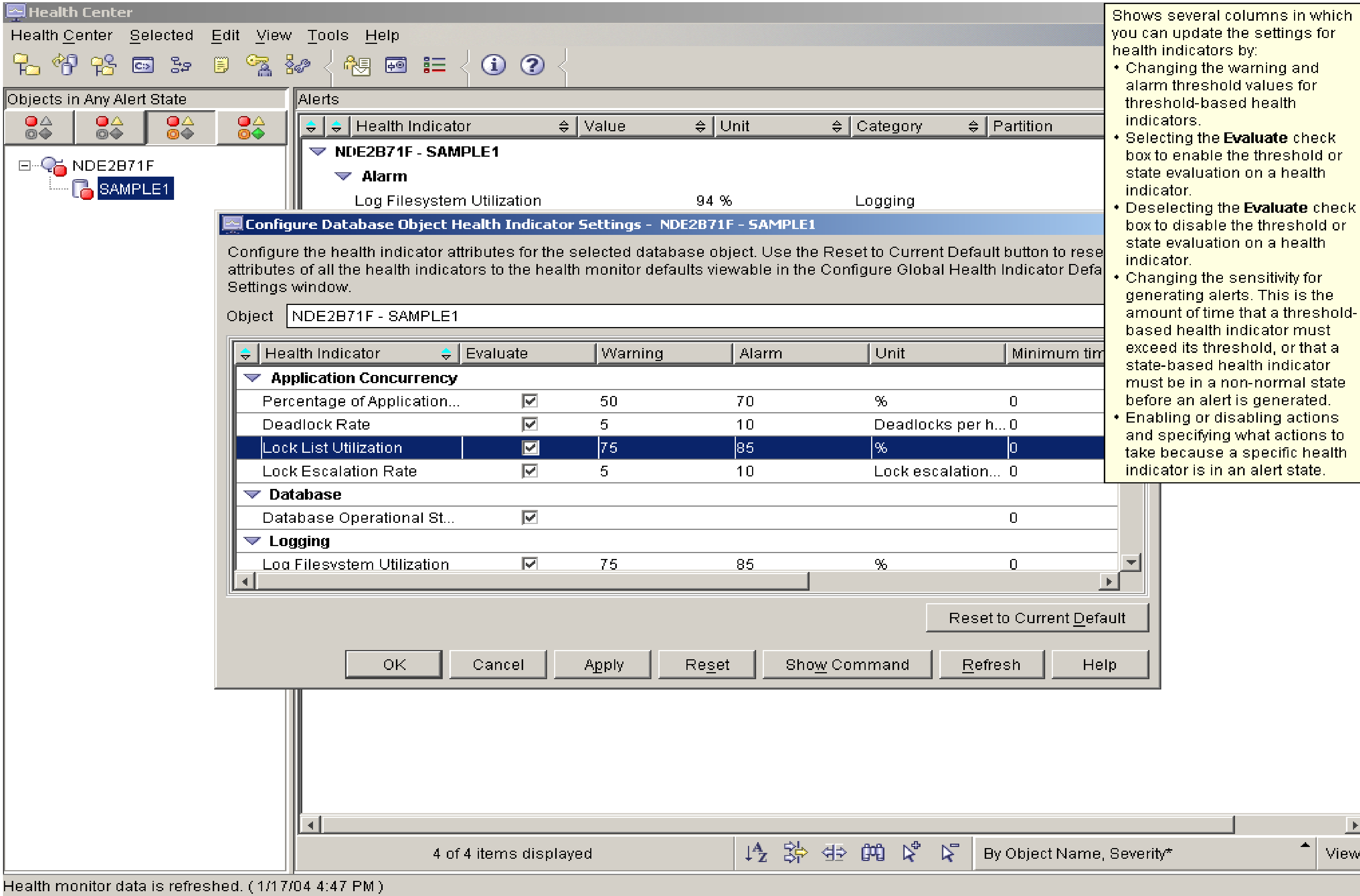
Health Center



Health Center



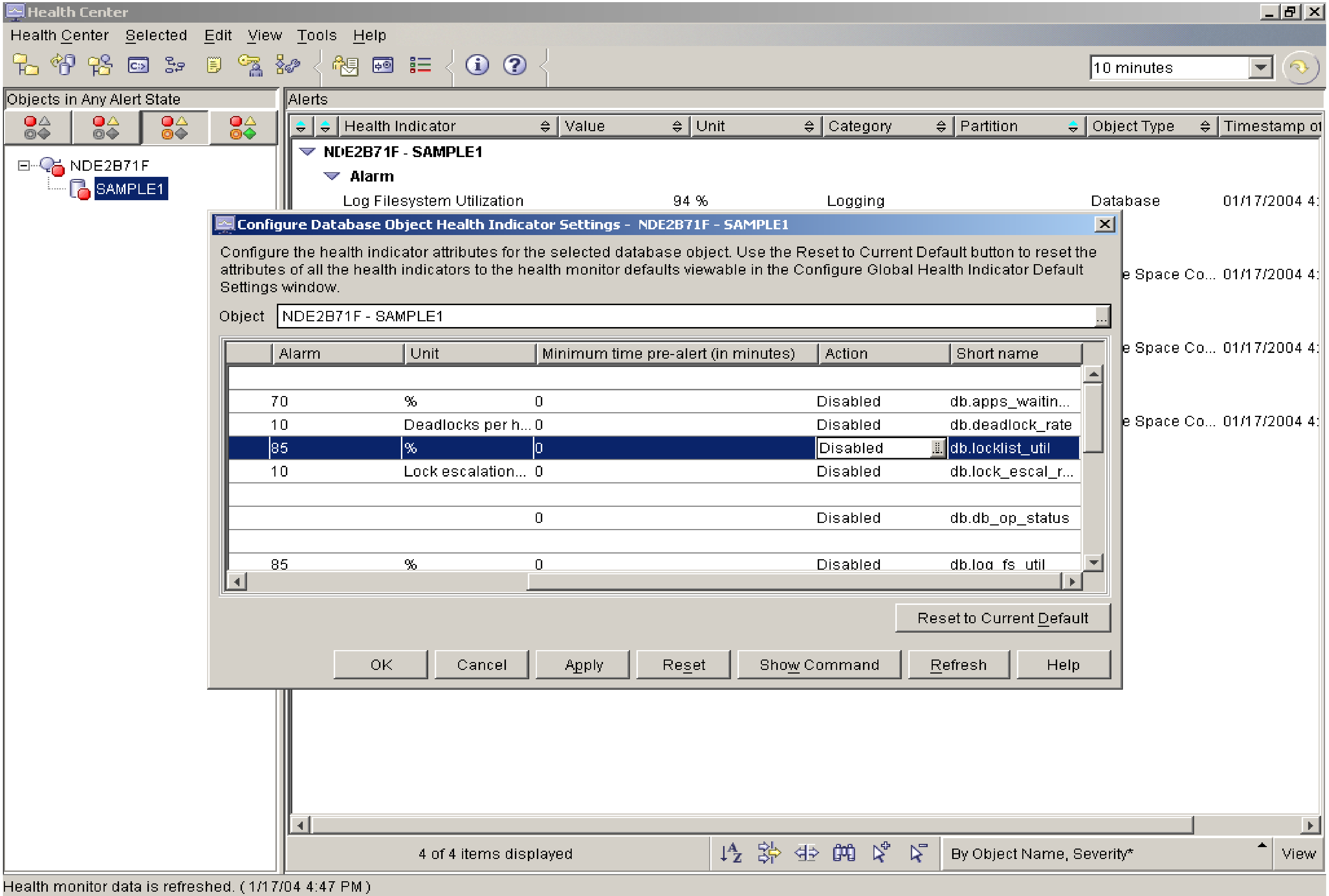
Health Center



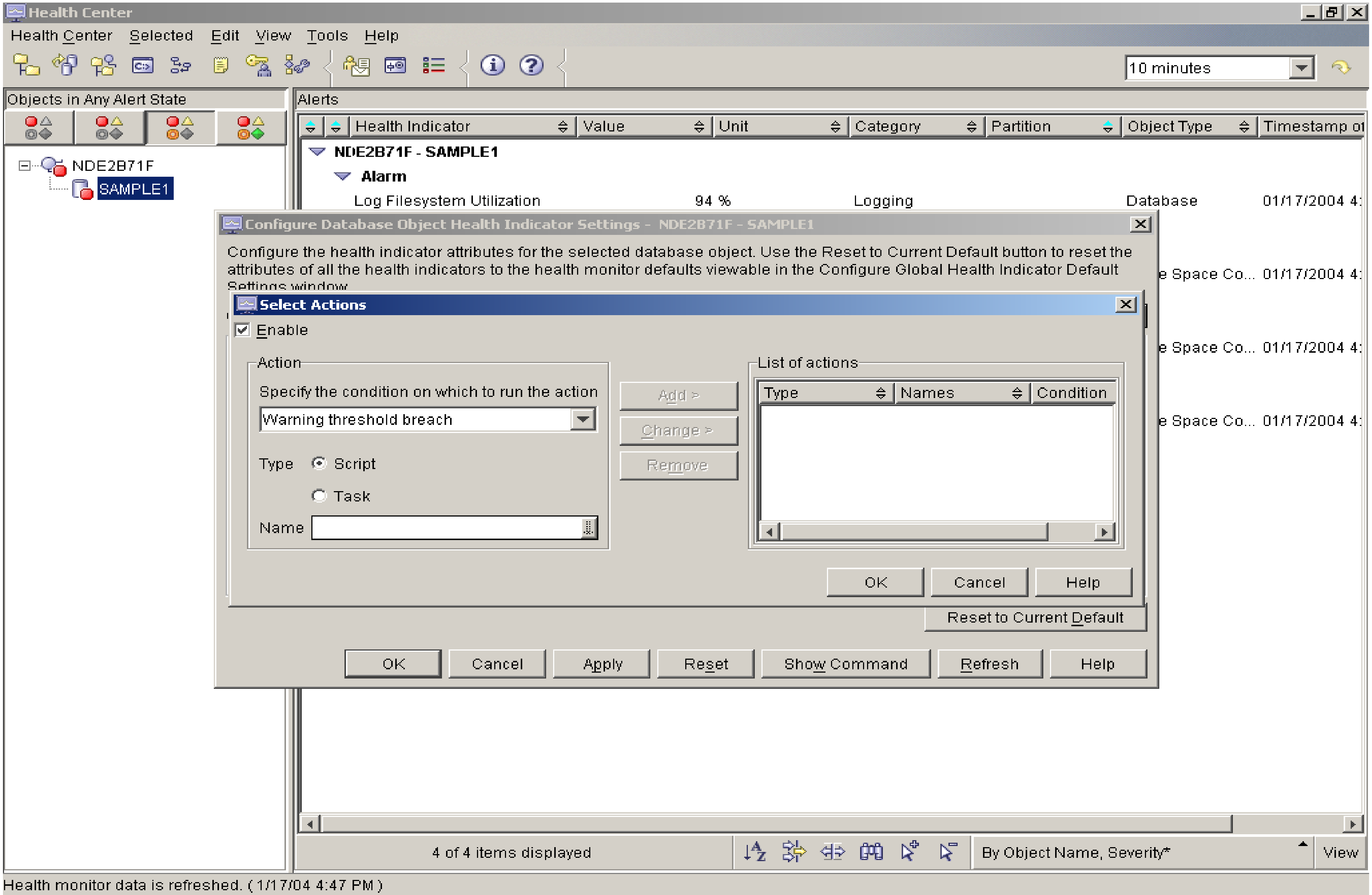
Shows several columns in which you can update the settings for health indicators by:

- Changing the warning and alarm threshold values for threshold-based health indicators.
- Selecting the **Evaluate** check box to enable the threshold or state evaluation on a health indicator.
- Deselecting the **Evaluate** check box to disable the threshold or state evaluation on a health indicator.
- Changing the sensitivity for generating alerts. This is the amount of time that a threshold-based health indicator must exceed its threshold, or that a state-based health indicator must be in a non-normal state before an alert is generated.
- Enabling or disabling actions and specifying what actions to take because a specific health indicator is in an alert state.

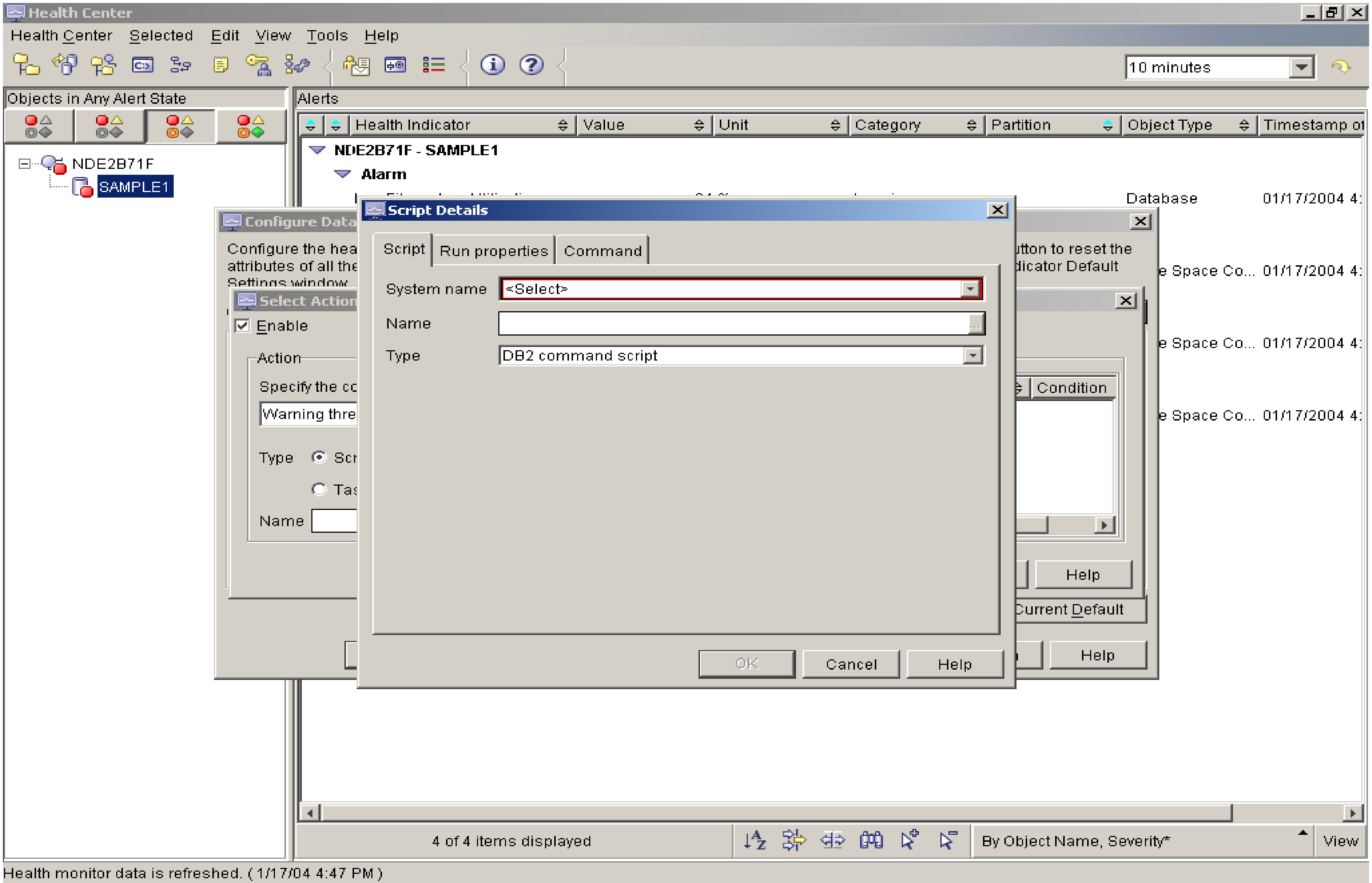
Health Center



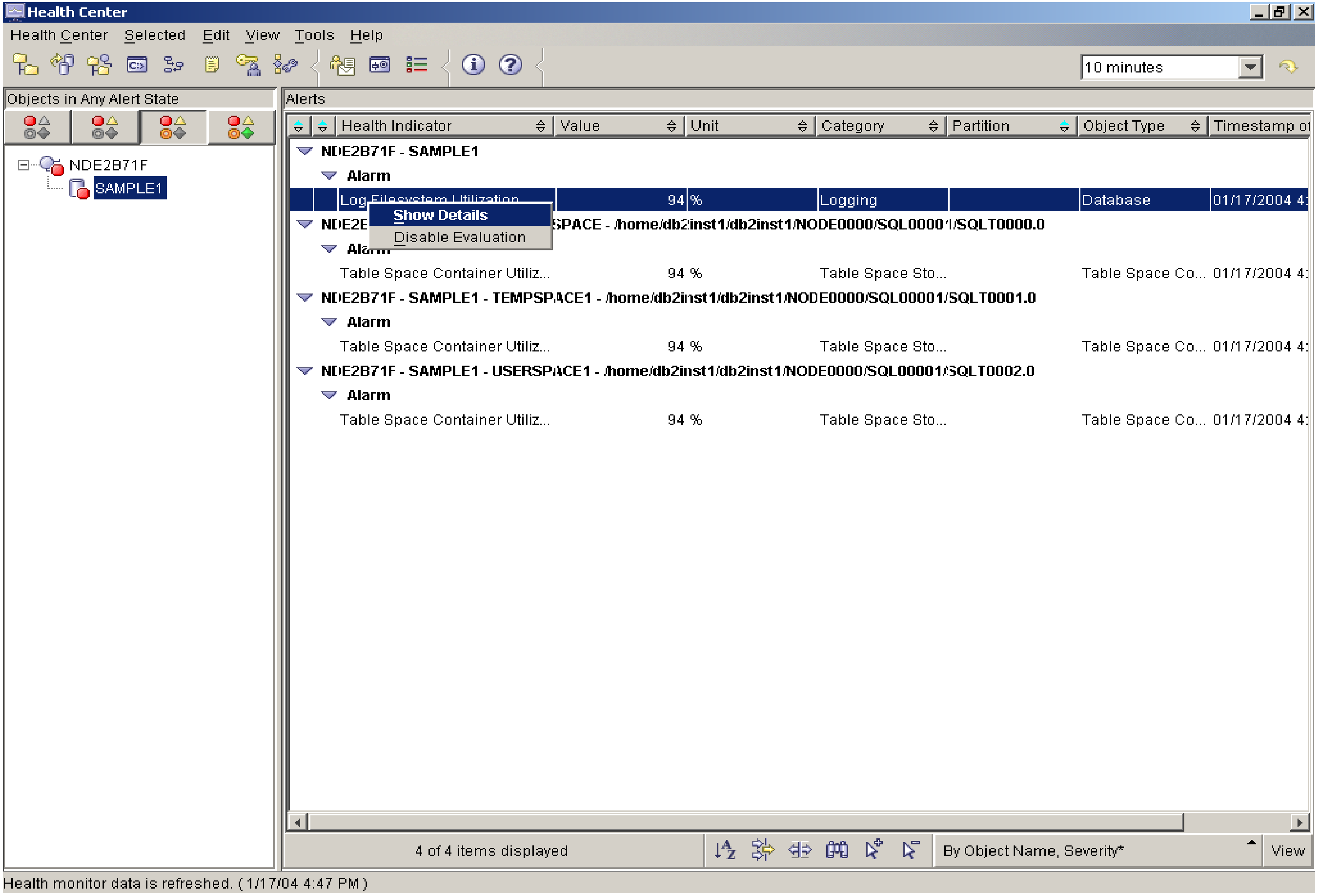
Health Center



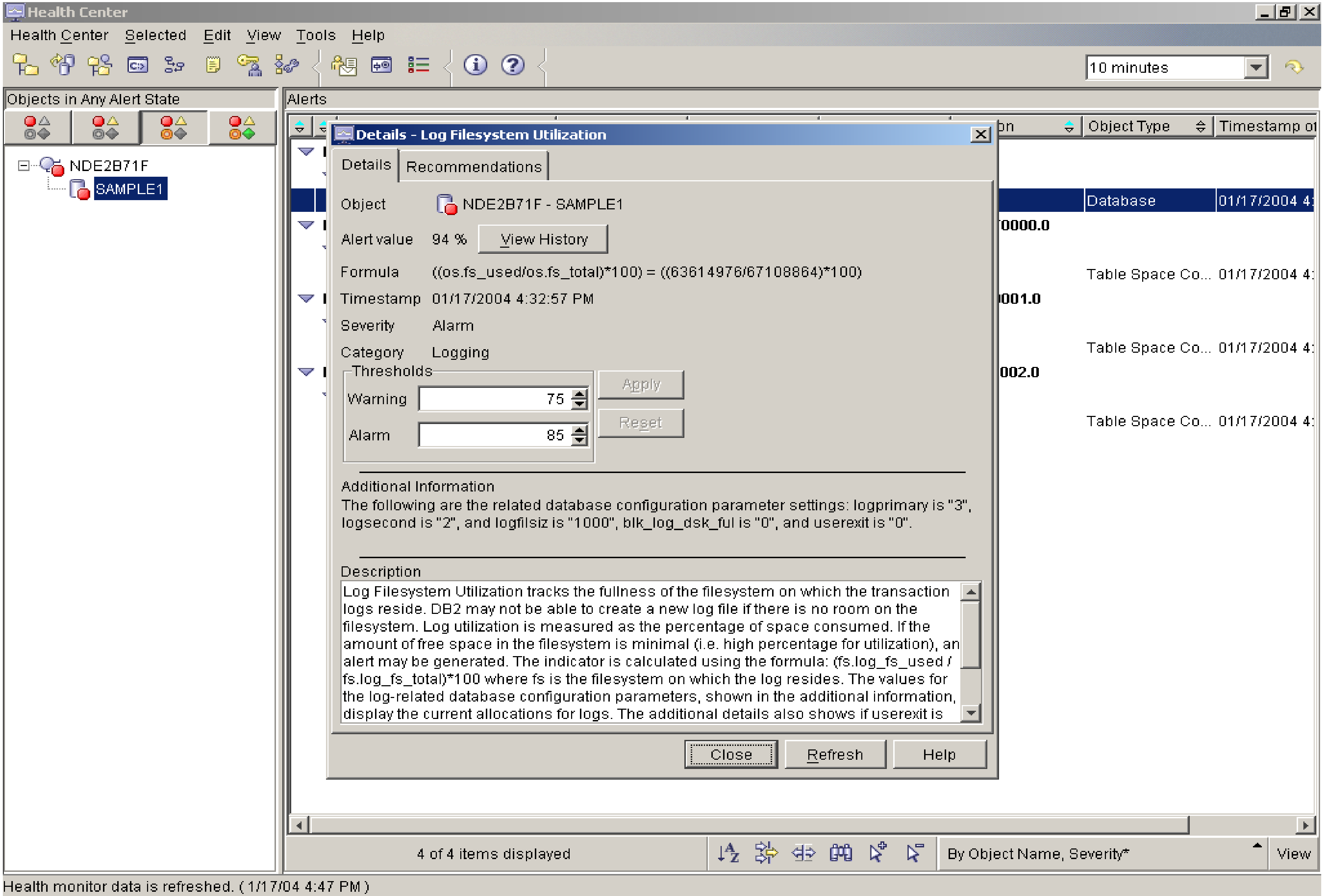
Health Center



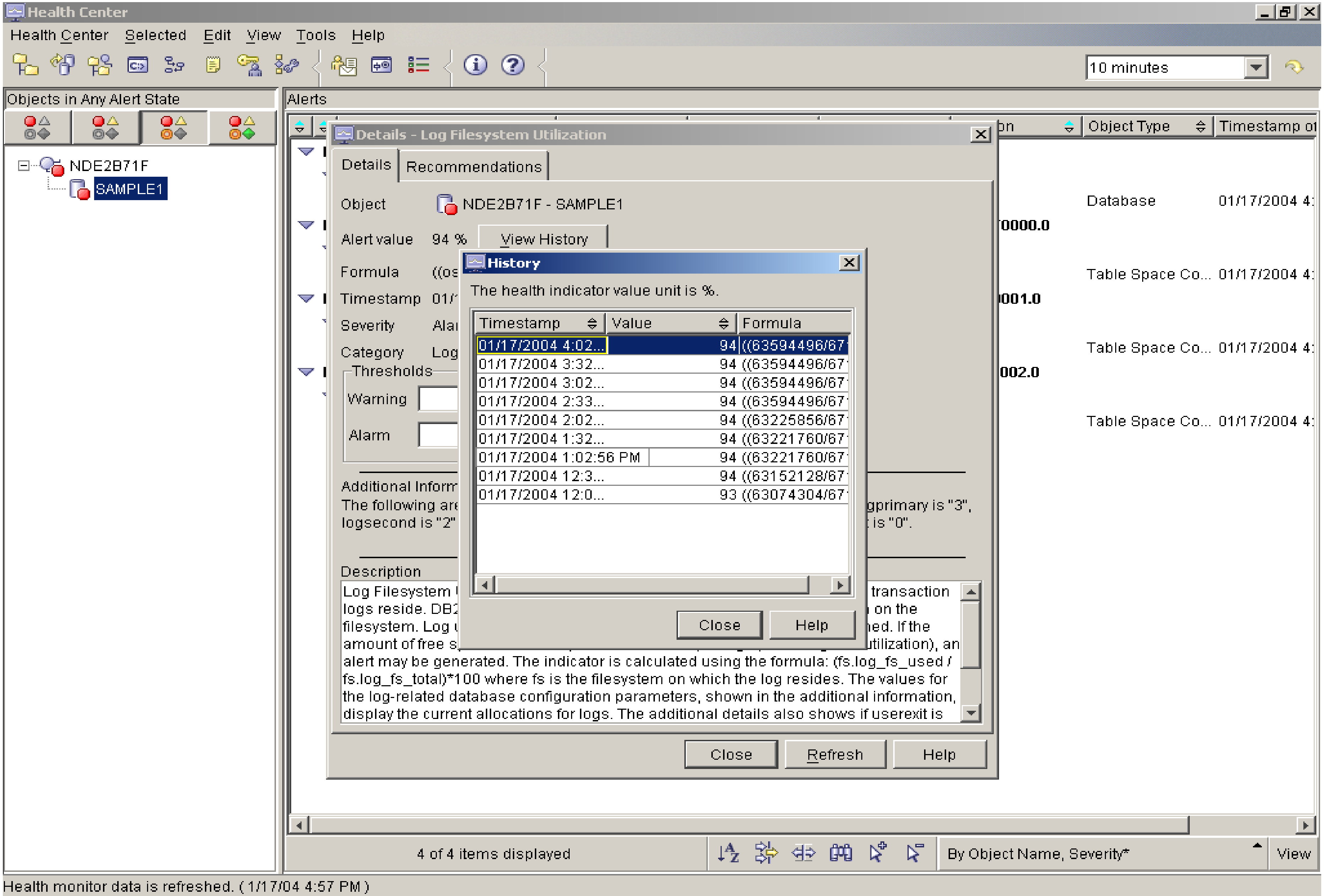
Health Center



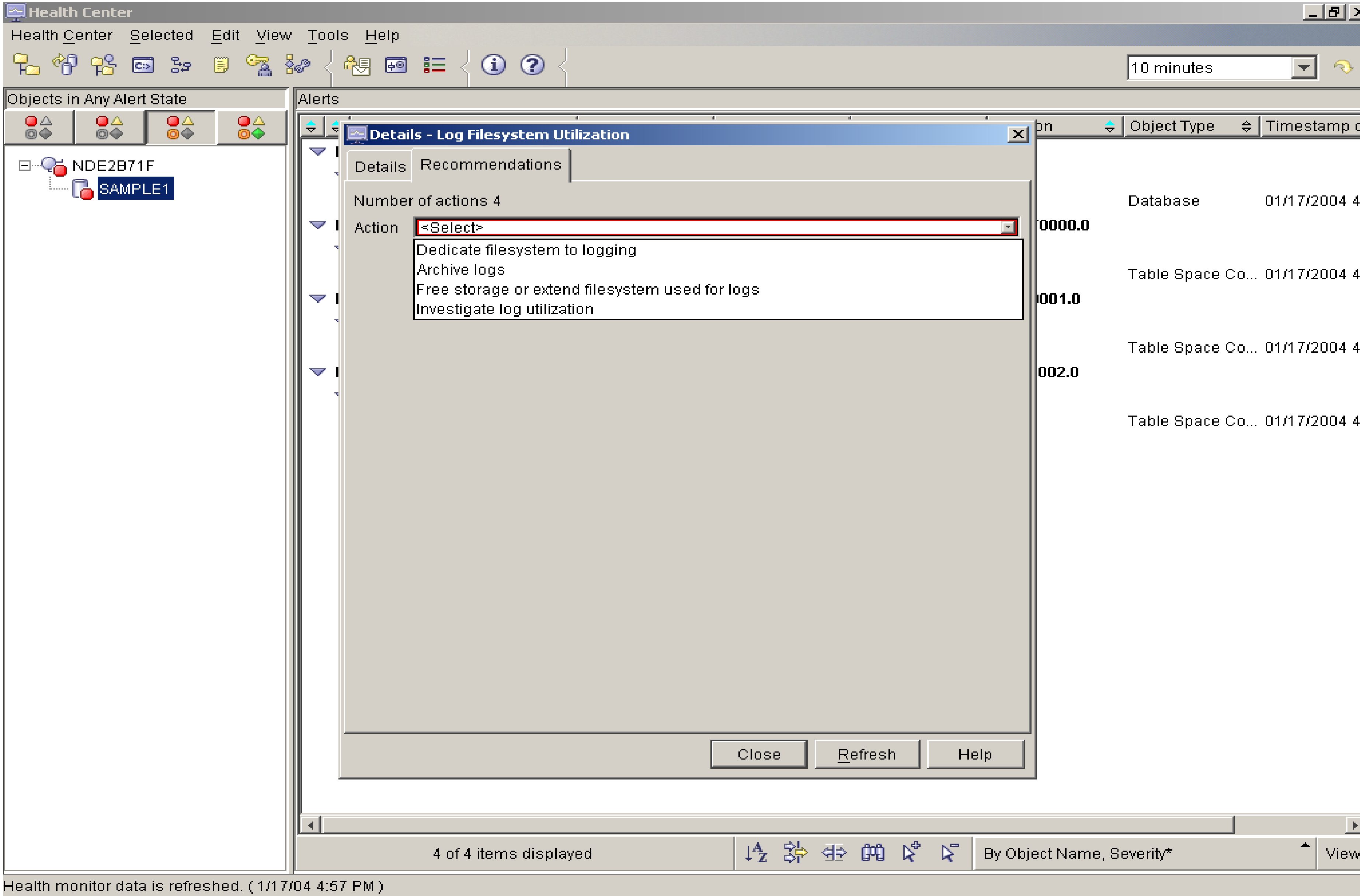
Health Center



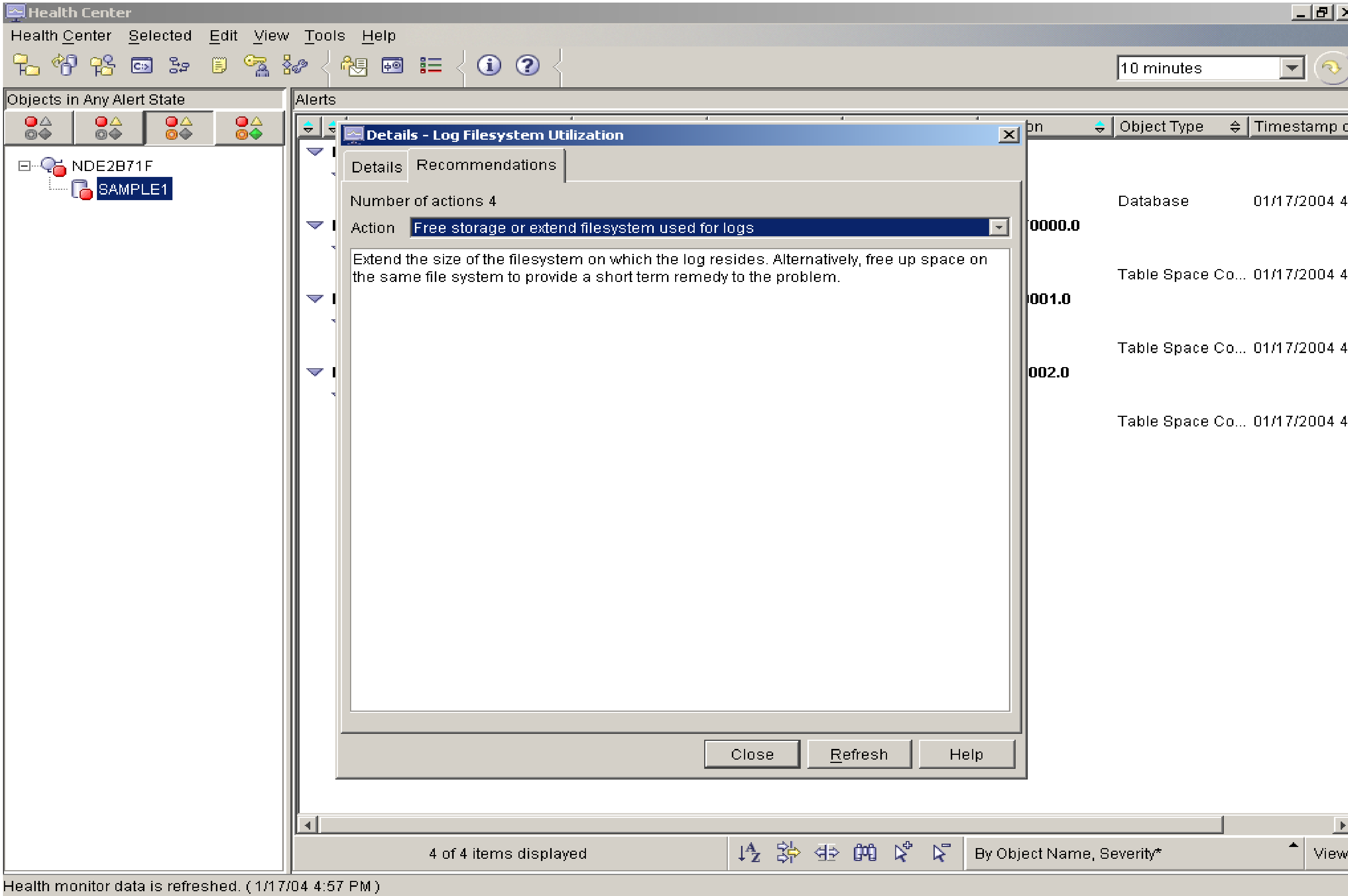
Health Center



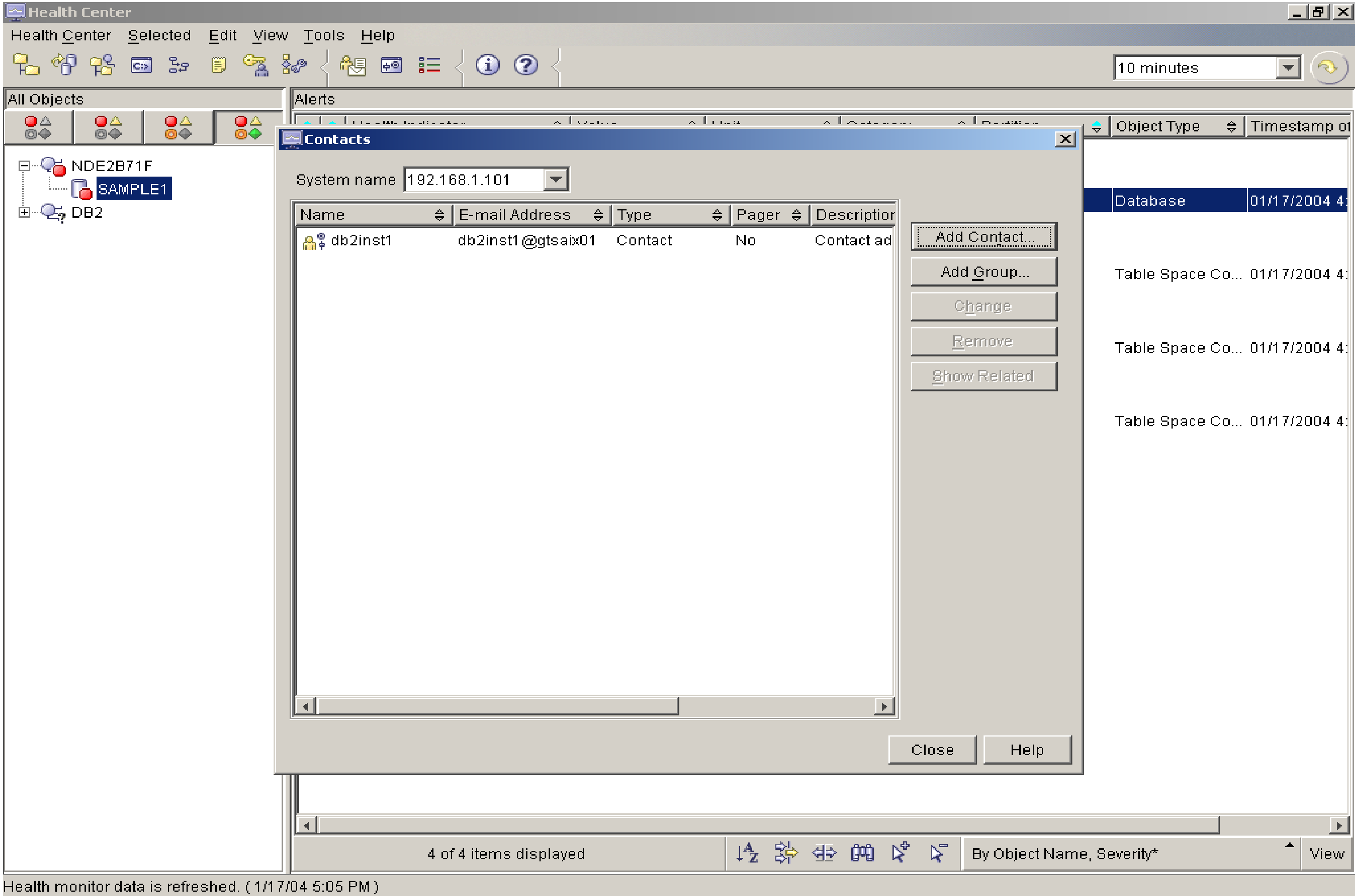
Health Center



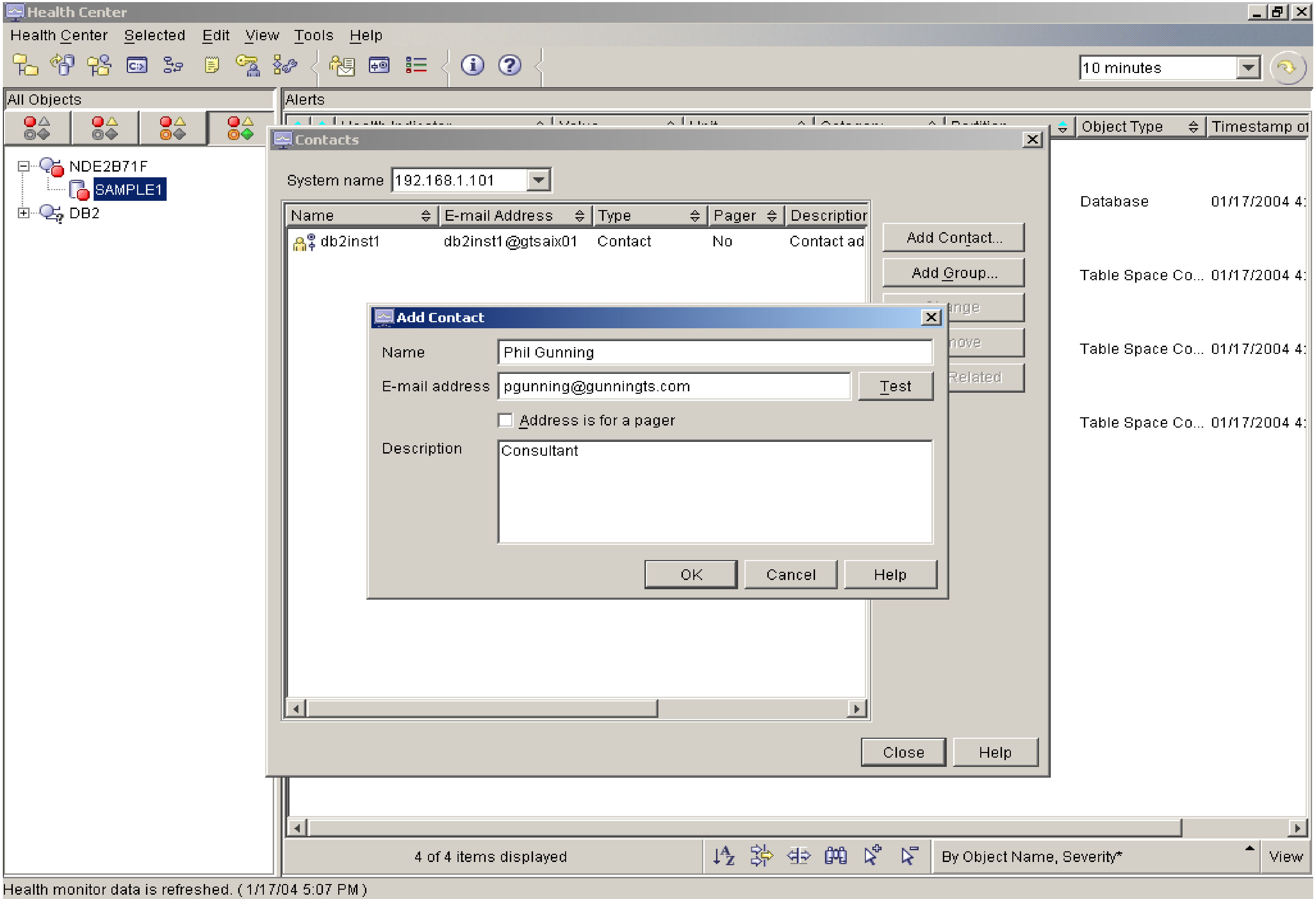
Health Center



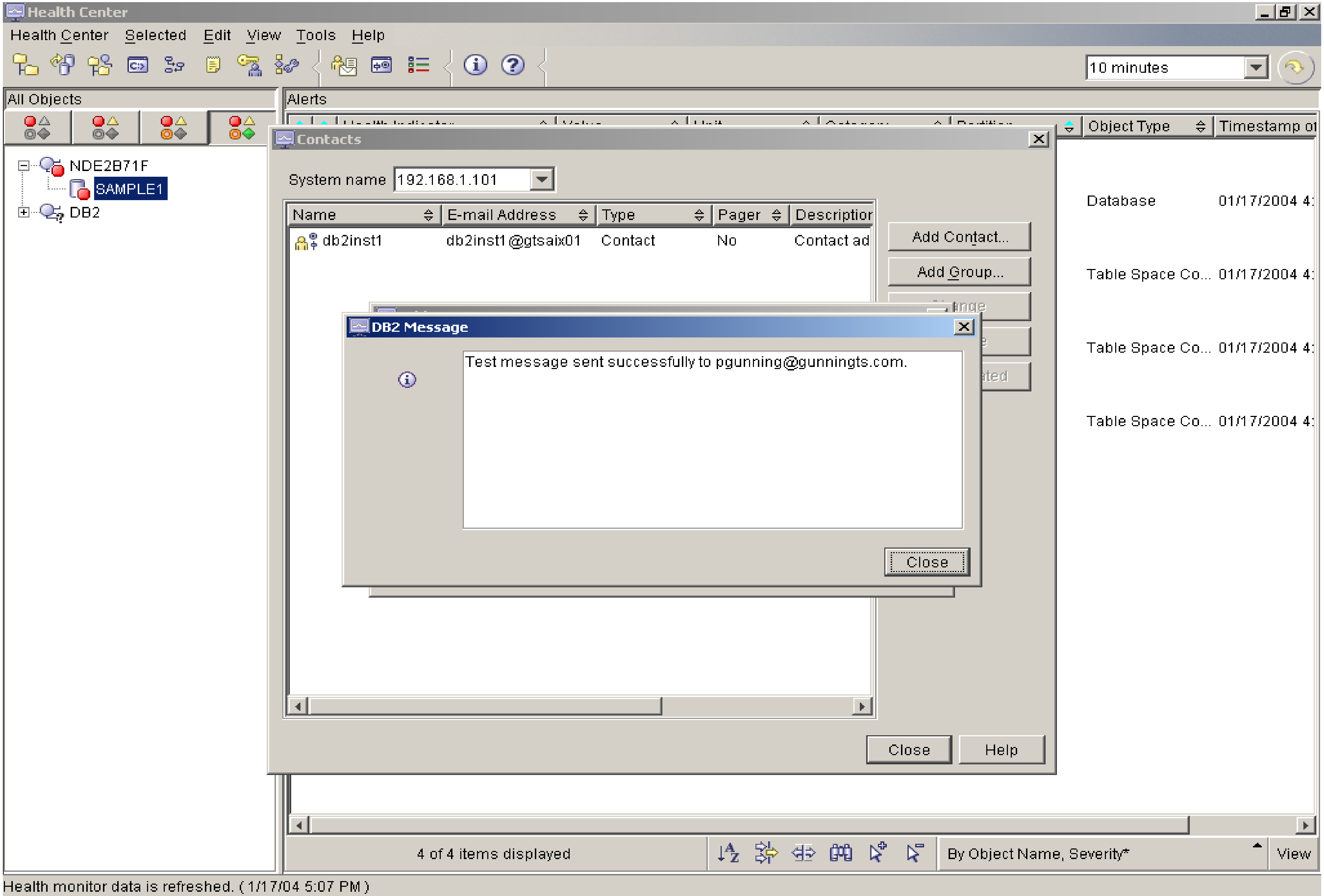
Health Center



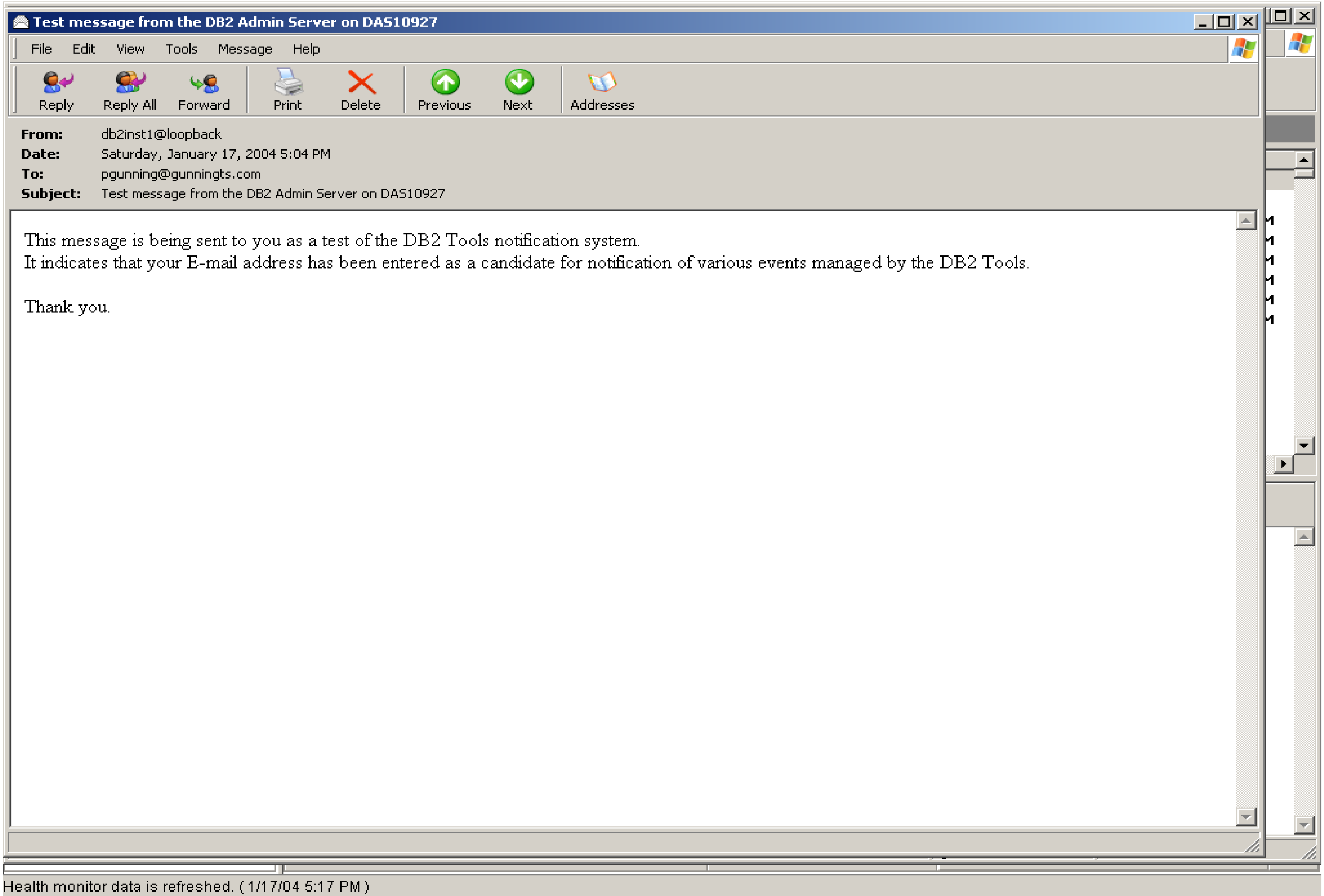
Health Center



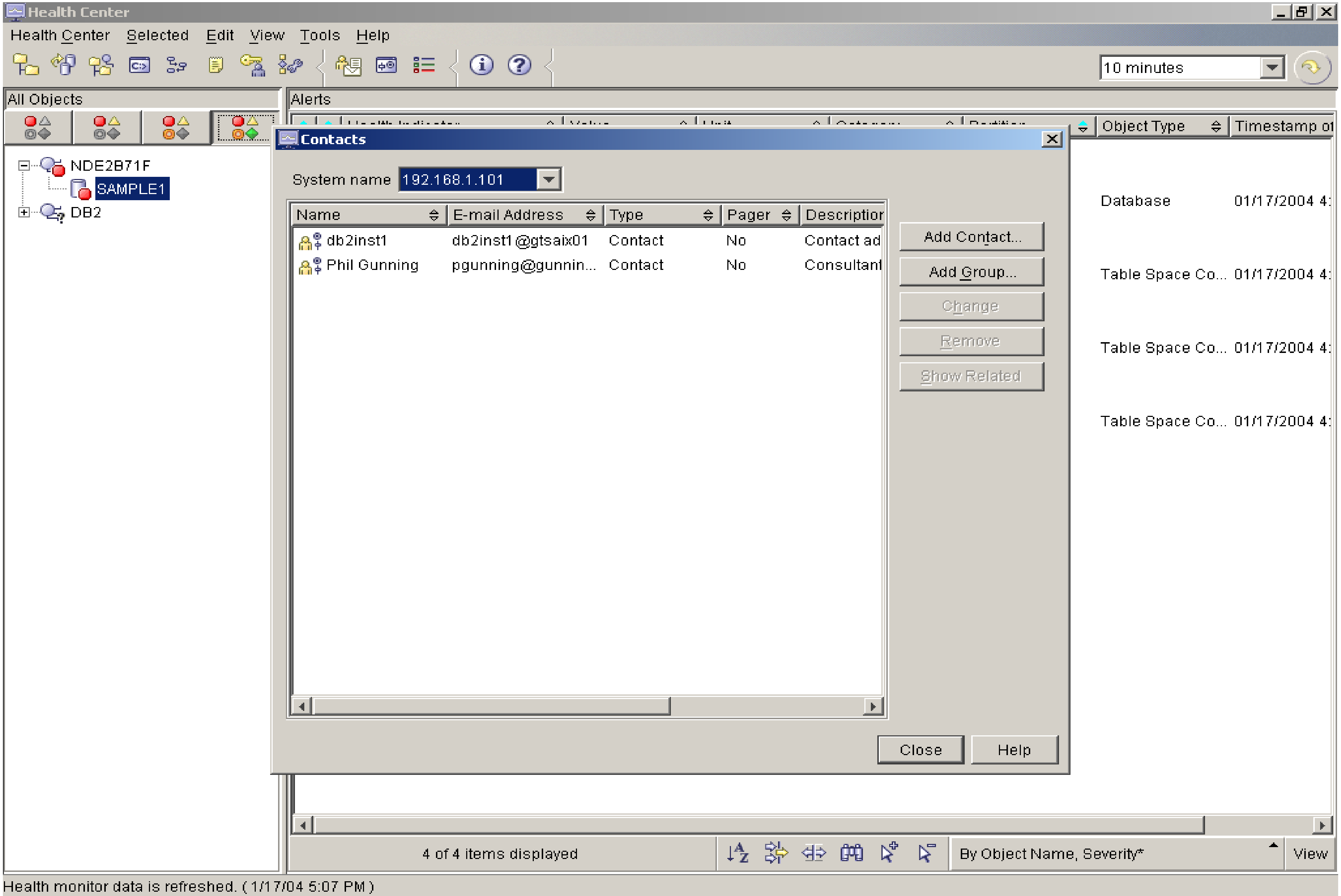
Health Center



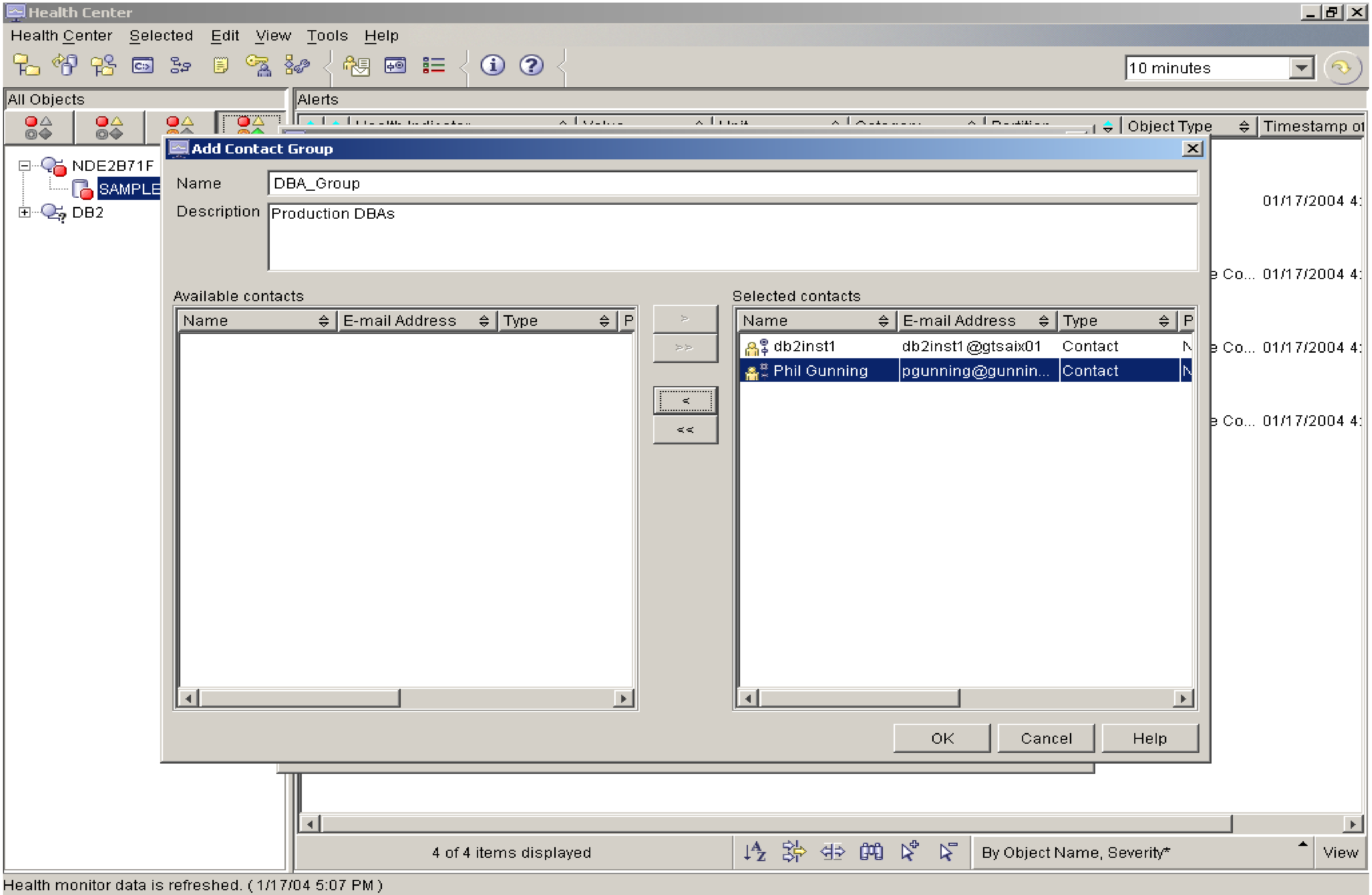
Health Center



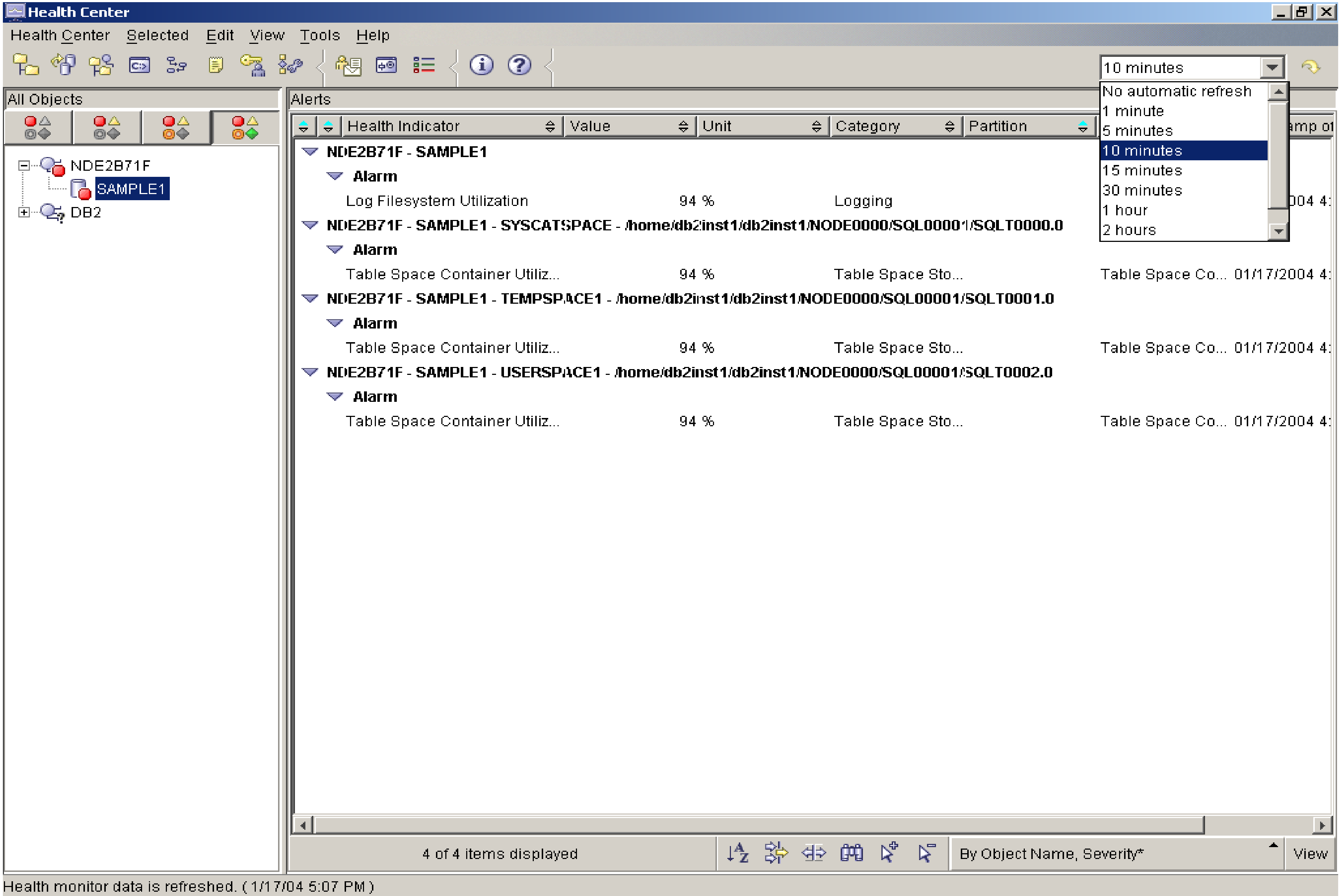
Health Center



Health Center



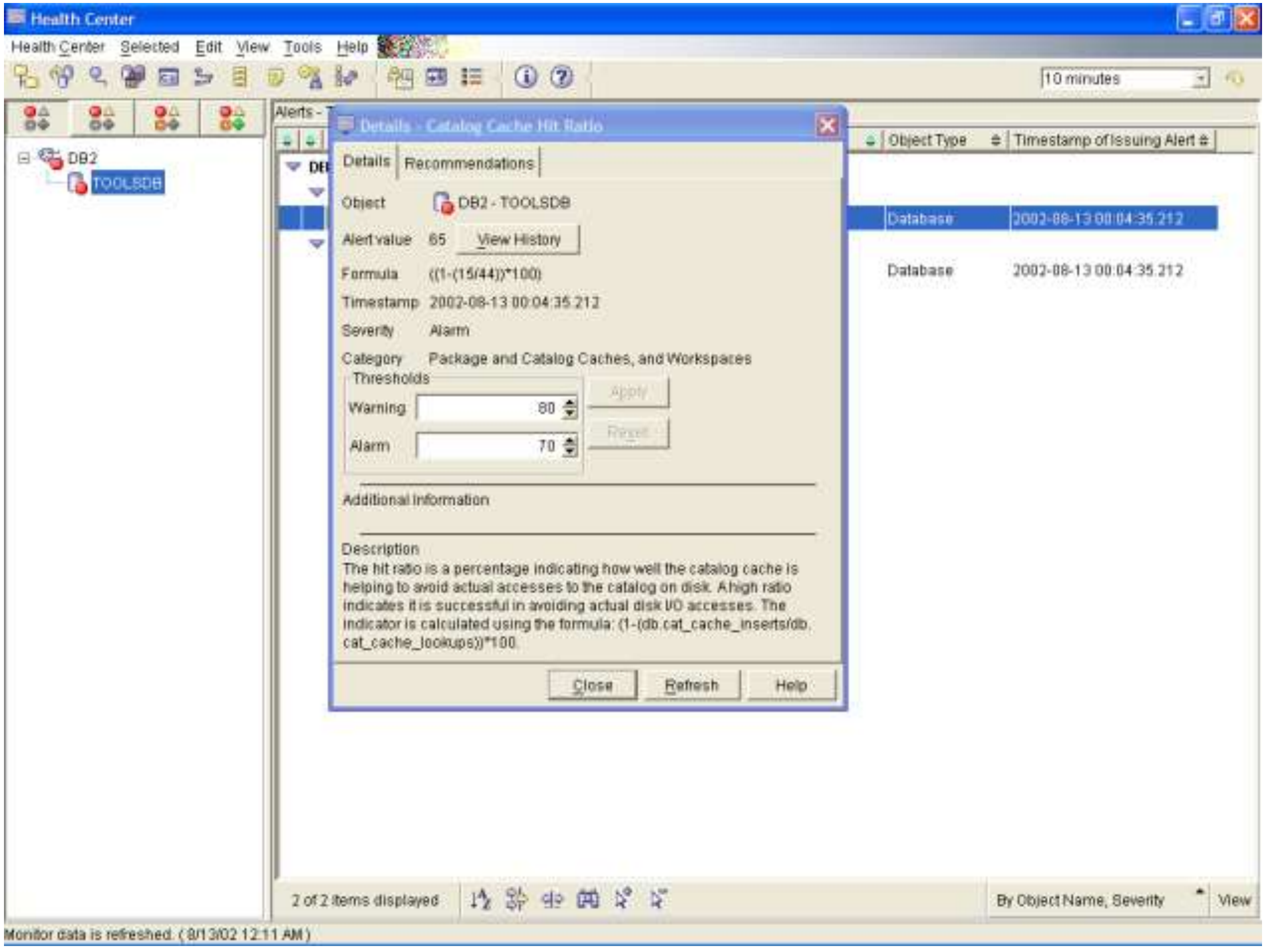
Health Center



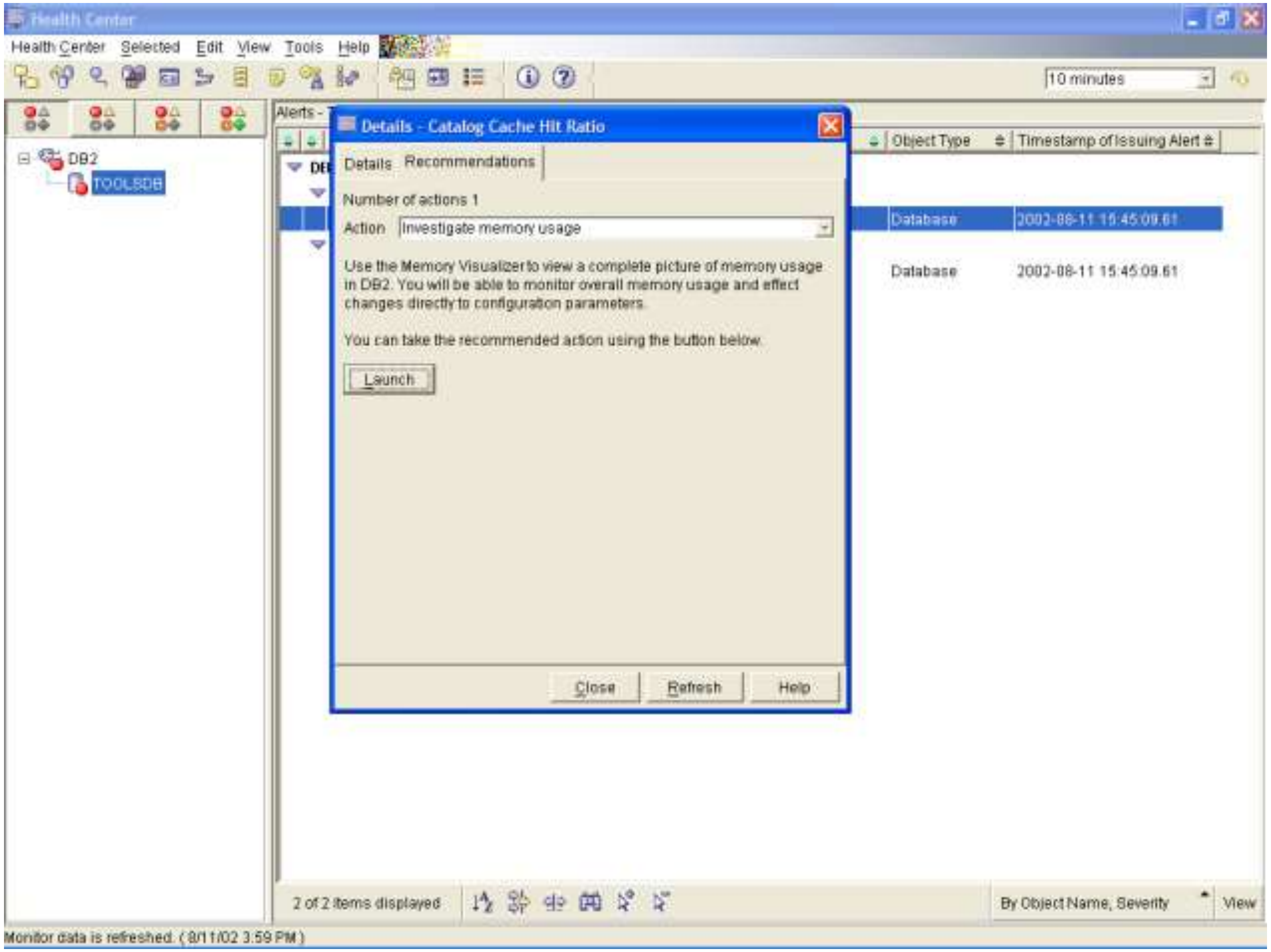
Health Center

- This page intentionally left blank

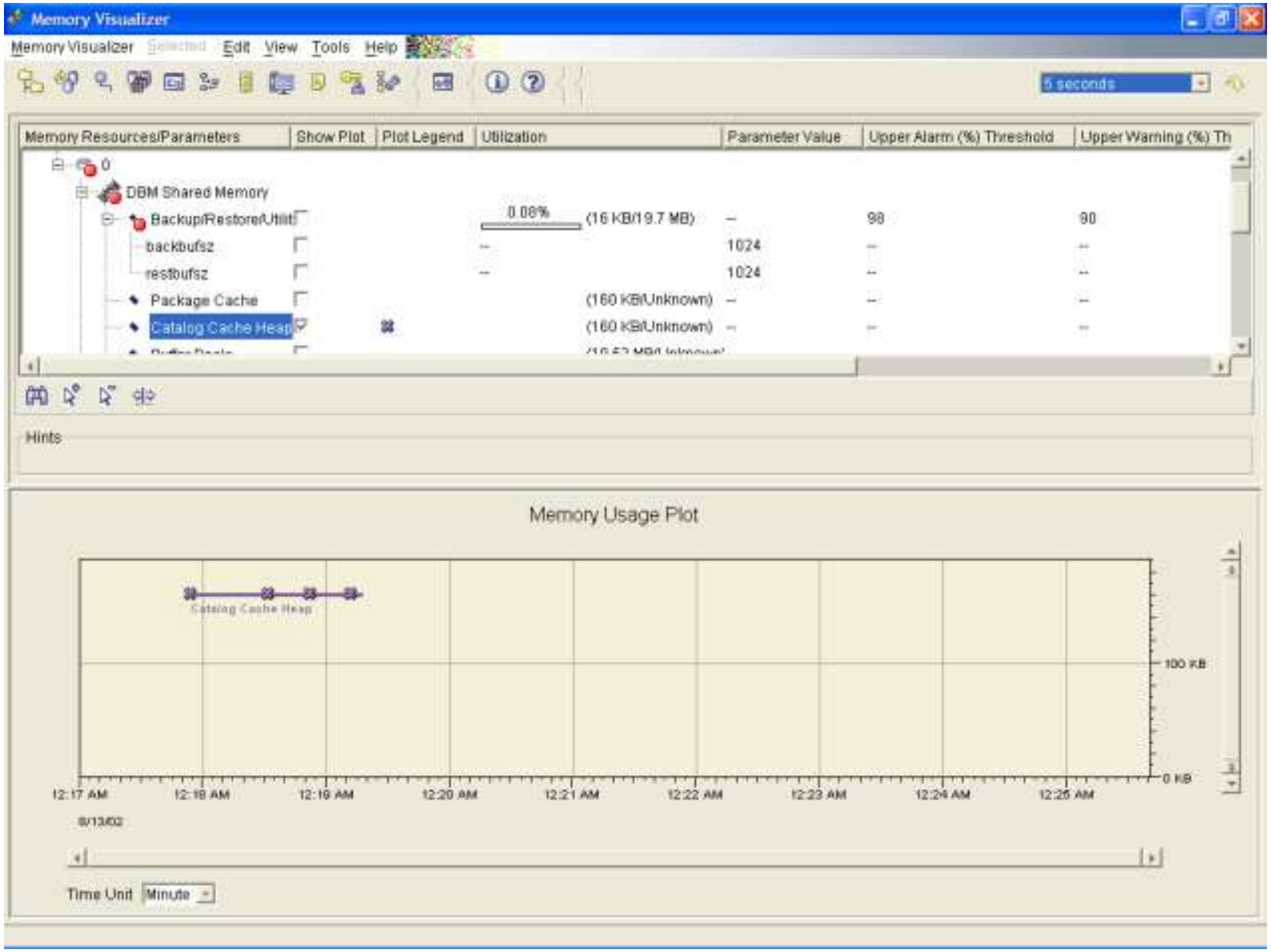
Health Center



Health Center



Memory Visualizer



DBM Health Indicators

- DBM health indicators for an instance can be obtained by issuing the following command:
  - “db2 get alert cfg for dbm”

DBM Health Indicators

Indicator Name = db2.db2_op_status

- Type = State-based
- Sensitivity = 0
- Formula = db2.db2_status;
- Actions = Disabled
- Threshold or State checking = Enabled

DBM Health Indicators

Indicator Name = db2.sort_privmem_util

Type = Threshold-based

Warning = 90

Alarm = 100

Sensitivity = 0

Formula = ((db2.sort_heap_allocated/sheapthres)*100);

Actions = Disabled

Threshold or State checking = Enabled

DBM Health Indicators

Indicator Name = db2.max_sort_privmem_util

Type = Threshold-based

Warning = 60

Alarm = 30

Sensitivity = 0

Formula = ((db2.max_priv_sort_mem/sheapthres)*100);

Actions = Disabled

Threshold or State checking = Enabled

DBM Health Indicators

Indicator Name = db2.mon_heap_util

Type = Threshold-based

Warning = 85

Alarm = 95

Sensitivity = 0

Formula = ((db2.mon_heap_cur_size/db2.mon_heap_max_size)*100);

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

- Database Configuration health indicators for a database can be obtained by issuing the following command:
  - "db2 get alert cfg for databases"

Database Health Indicators

Indicator Name = db.sort_shrmem_util

- Type = Threshold-based
- Warning = 70
- Alarm = 85
- Sensitivity = 0
- Formula = ((db.sort_shrheap_allocated/sheapthres_shr)*100);
- Actions = Disabled
- Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.spilled_sorts

Type = Threshold-based

Warning = 30

Alarm = 50

Sensitivity = 0

Formula = (db.sort_overflows/db.total_sorts)*100;

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.max_sort_shrmem_util

Type = Threshold-based

Warning = 60

Alarm = 30

Sensitivity = 0

Formula = ((db.max_shr_sort_mem/sheapthres_shr)*100);

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.log_util

Type = Threshold-based

Warning = 75

Alarm = 85

Sensitivity = 0

Formula = (db.total_log_used/(db.total_log_used+db.total_log_available))*100;

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.log_fs_util

Type = Threshold-based

Warning = 75

Alarm = 85

Sensitivity = 0

Formula = ((os.fs_used/os.fs_total)*100);

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.deadlock_rate

- Type = Threshold-based
- Warning = 5
- Alarm = 10
- Sensitivity = 0
- Formula = delta(db.deadlocks);
- Actions = Disabled
- Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.locklist_util

Type = Threshold-based

Warning = 75

Alarm = 85

Sensitivity = 0

Formula = (db.lock_list_in_use/(locklist*4096))*100;

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.lock_escal_rate

- Type = Threshold-based
- Warning = 5
- Alarm = 10
- Sensitivity = 0
- Formula = delta(db.lock_escals);
- Actions = Disabled
- Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.apps_waiting_locks

Type = Threshold-based

Warning = 50

Alarm = 70

Sensitivity = 0

Formula = (db.locks_waiting/db.appls_cur_cons)*100;

Actions = Disabled

Threshold or State checking = Enabled

Database Health Indicators

Indicator Name = db.pkgcache_hitratio

Type = Threshold-based

Warning = 80

Alarm = 70

Sensitivity = 0

Formula = (1-(db.pkg_cache_inserts/db.pkg_cache_lookups))*100;

Actions = Disabled

Threshold or State checking = Disabled

Database Health Indicators

Indicator Name = db.catcache_hitratio

Type = Threshold-based

Warning = 80

Alarm = 70

Sensitivity = 0

Formula = (1-(db.cat_cache_inserts/db.cat_cache_lookups))*100;

Actions = Disabled

Threshold or State checking = Disabled

Database Health Indicators

Indicator Name = db.shrworkspace_hitratio

- Type = Threshold-based
- Warning = 80
- Alarm = 70
- Sensitivity = 0
- Formula = ((1-(db.shr_workspace_section_inserts/db.shr_workspace_section_lookups))*100);
- Actions = Disabled
- Threshold or State checking = Disabled

Database Health Indicators

Indicator Name = db.db_heap_util

Type = Threshold-based

Warning = 85

Alarm = 95

Sensitivity = 0

Formula = ((db.db_heap_cur_size/db.db_heap_max_size)*100);

Actions = Disabled

Threshold or State checking = Enabled

Module 9
Memory Tracker and DB2BATCH

Objectives

- Upon completion of this module you should be able to:
  - Monitor a database using memory tracker
  - Understand basic memory tracker usage and capabilities
  - Understand basics of db2batch
  - Be able to use db2batch to perform benchmark tests
  - Improve performance of db2 utilities using paralellism

Memory Tracker

- DB2 Memory Tracker is a new tool available in DB2 V8.1
- Similar usage to UNIX commands such as VMSTAT, IOSTAT
- Run from DB2 command line
- Command and options follow:
- db2mtrk -i | -d | -p [-m | -w] [-v] [-r interval [count]] [-h]
- Handy for tracking memory usage over time interval
- Can be used for point based monitoring and could be scripted and data stored in DB2 tables to show DB2 memory/heap activity over time

Memory Tracker Output

```
Memory for database: SAMPLE

   Backup/Restore/Util Heap has max size of 20660224 bytes
   Package Cache has max size of 536870912 bytes
   Catalog Cache Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Lock Manager Heap has max size of 638976 bytes
   Database Heap has max size of 6127616 bytes
   Other Memory has max size of 12517376 bytes
   Total: 3798040576 bytes

Memory for agent 24254

   Application Heap has max size of 1277952 bytes
   Application Control Heap has max size of 704512 bytes
   Total: 1982464 bytes

Tracking Memory on: 2004/01/12 at 02:18:36

Memory for instance

   Database Monitor Heap has max size of 540672 bytes
   Other Memory has max size of 13467648 bytes
   Total: 14008320 bytes

Memory for database: SAMPLE

   Backup/Restore/Util Heap has max size of 20660224 bytes
   Package Cache has max size of 536870912 bytes
   Catalog Cache Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Buffer Pool Heap has max size of 536870912 bytes
   Lock Manager Heap has max size of 638976 bytes
   Database Heap has max size of 6127616 bytes
   Other Memory has max size of 12517376 bytes
   Total: 3798040576 bytes

Memory for agent 24254

   Application Heap has max size of 1277952 bytes
   Application Control Heap has max size of 704512 bytes
   Total: 1982464 bytes
```

Memory Tracker

- Memory Tracker provides a complete report of memory status for instance, databases, and agents
- The following memory pool allocations are displayed:
  - Current size and maximum size
  - Largest size and type
  - Agent that allocated the pool (if private)

Memory Tracker

- Memory tracker is a good tool that should be integrated into your total monitoring strategy

Health Monitoring Workshop

- Refer to your student workbook and examples for the Health Monitoring Workshop. You have 1 hr to complete the workshop.

Db2batch and Utilities

DB2BATCH

- db2batch is a benchmarking tool provided with DB2
  - Reads SQL statements from either a flat file or standard input and dynamically prepares and describes statements and returns an answer set

DB2BATCH

- The number of rows to be fetched, sent to the output file or standard out can be controlled and the level of performance returned
- SQL statements can be included in a block and information is collected for all of the statements at once, instead of one at a time

DB2BATCH

- Command parameters
  - ROWS_FETCH – number of rows to be fetched from the answer set
  - ROWS_OUT – Number of rows to be sent to output
  - PERF_DETAIL – Level of performance info to be returned
    - 0 – no timing done
    - 1 – return elapsed time only
    - 2 – return elapsed time and cpu time
    - 3 – return a summary of monitoring information
    - 4 – return a snapshot for DBM, DB, application, statement (single statement)
    - 5 – similar to level 4 but return bufferpool, tablespace, and FCM

DB2BATCH

- Command parameters continued:
  - DELIMITER
  - SLEEP – Number of seconds to sleep
  - PAUSE – Prompts the user to continue
  - TIMESTAMP – Generates a timestamp
- Similar parameters for output and the following:
  - o query_optimization_class
  - e explain_mode
  - v Verbose
  - s summary table
  - p ESE DPF only
  - Cli Run db2batch in CLI mode
  - cache_size – size of the statement memory, expressed as number of statements

Utilities

- Throttling of utilities became available in DB2 V8.1.2
- With the introduction of utility throttling, you can regulate the performance impact of maintenance utilities, so that they can be run concurrently with production periods. You can develop a throttling policy that will run the utilities aggressively when the production workload is light, but will run them more conservatively as production demands increase.

Utilities

- Use parallelism to speed up utilities
- Either CPU or I/O parallelism
  - Backup multiple tablespaces in parallel

Utilities

- Online Load
  - Other tables in same talbespace no longer locked out
  - Provides additional flexibility
  - Improves availability

DB2 UDB V8.1 References

References:

IBM DB2 Universal Database Command Reference	SC09-4828
IBM DB2 Universal Database What's New	SC09-4848
IBM DB2 Universal Database Administration Guide: Planning	SC09-4822
IBM DB2 Universal Database Administration Guide: Implementation	SC09-4820
IBM DB2 Universal Database Administration Guide: Performance	SC09-4821
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847

DB2 UDB for Linux, UNIX, and Windows Performance and Tuning