Buffer Pool Statistics

The **Buffer Pool** function invokes the **Buffer Pool Statistics** menu, which is used to obtain buffer-pool-related statistics (including hash table statistics) that are independent of Natural objects.

To invoke Buffer Pools Statistics

- In the SYSBPM **Main Menu**, enter the following function code:

```
A
```

  Or:
  Enter the following SYSBPM direct command:

```
DISPLAY STATISTICS
```

  The **Buffer Pool Statistics** menu appears.

The functions available in the **Buffer Pool Statistics** menu and the commands provided on the screens that are invoked by these functions are described in this section.

- General Buffer Pool Statistics
- Buffer Pool Load/Locate Statistics
- Buffer Pool Fragmentation
- Internal Function Usage
- Buffer Pool Hash Table Statistics
- Performance Hints
- PF Keys and Direct Commands

General Buffer Pool Statistics

This function is used to monitor the performance of the buffer pool, and displays statistics regarding the activity of the buffer pool.

To invoke General Buffer Pool Statistics

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
G
```

Or:
Enter the following SYSBPM direct command:

```
DISPLAY GENERAL
```

The **General Buffer Pool Statistics** screen appears.

The statistics displayed on the **General Buffer Pool Statistics** screen are snapshots of the buffer pool which are refreshed each time you press ENTER. The following information is displayed:

Field	Explanation
Buffer Pool Address	The address of the buffer pool.
Directory Section	The address of the buffer pool directory section relative to the beginning of the buffer pool. Each object loaded in the buffer pool requires a directory entry that contains information on this object. The space for these directory entries is allocated in the buffer pool.
Text Record Section	The address of the text record section relative to the beginning of the buffer pool. After the space used by the directory entries has been allocated, the remaining space is divided into blocks called text records (whose size, by default, is 4 KB). An object can occupy one or more text records, depending on its size.
Dataspace attached	The name of the dataspace (BP cache) attached to the buffer pool.
Buffer Pool Size (MB)	The size of the whole buffer pool in MB. The buffer pool size can be specified with the NTBPI macro in the parameter module or with the `BPI` profile parameter described in the *Parameter Reference* documentation.
Directory Entry Size	The size of a directory entry in bytes.
Text Record Size (KB)	The size of a text record in KB. The text record size can be specified with the NTBPI macro in the parameter module or with the `BPI` profile parameter described in the *Parameter Reference* documentation. You can change the text record size of an existing buffer pool if you reinitialize the buffer pool by using the `INITIALIZE` command. The default text record size is set to 4 KB. However, if you use applications that consist of many rather small objects, we recommend that you reduce it to 2 KB. This reduces the percentage of unused space in the buffer pool, although it can lead to Algorithm 2 (see *METHOD=S* in the *Operations* documentation) being invoked more frequently.
Buffer Pool Start	The date and time when the buffer pool was originally started.

Field	Explanation
Last Initialization	The date and time when the buffer pool was most recently initialized, and the ID of the user who performed the initialization. The buffer pool is initialized when: • originally starting the buffer pool, • executing the `INITIALIZE` SYSBPM direct command, or • executing the `REFRESH` command of the GBP operating program; see also *Global Buffer Pool Operating Functions* in the *Operations* documentation.
Text Records - Total	The total number of text records.
Text Records - Used	The number of text records currently used.
Text Records - Used in %	The percentage of text records currently used.
Text Records - Max Used	The maximum number of text records used.
Text Records - Total Size	The total space used by all text records used, which is **Text Records - Used** multiplied by the size of a single text record. The difference between the total text record size and the total object size shows the amount of unused size in the text record section and can also be an indicator for the system administrator of whether to modify the text record size.
Text Records - Avg Usage %	The average usage in percent of all text records used, which is **Objects - Total Size** divided by **Text Records - Total Size**. This value should not be significantly less than 75%. If the buffer pool is almost full, any value above 75% indicates good usage of the buffer pool. If the usage is significantly less than 75%, the text record size should be reduced.
Space Used %	The actual usage in percent of the text record section, which is **Objects - Total Size** divided by the total size of the **Text Records** section. **Tip:** If the buffer pool is almost full (that is, the value in the field **Text Records - Used** is almost 100%), any value above 75% indicates good usage of the buffer pool. If the usage is significantly less than 75%, the text record size should be reduced.
Objects - Loaded	The number of objects currently loaded in the buffer pool.
Objects - Max Loaded	The maximum number of objects ever loaded simultaneously in the buffer pool since the buffer pool was started.
Objects - Total Size	The total size in bytes of the objects currently loaded.
Objects - Avg TR Used	The average number of text records used by one object.

Field	Explanation
Objects - SumOfUseCounts	Totals the use counts of all objects currently loaded in the buffer pool. The use count counts all applications currently executing an object. If an object is currently not in use, its use count changes to `0` (zero).
Objects - AvgLifetimeUsed(min)	The average life time (in minutes) of objects currently loaded in the buffer pool.
Objects - AvgLifetimeReplace(min)	The average life time (in minutes) of objects, which have already been replaced in the buffer pool.

Buffer Pool Load/Locate Statistics

This function provides statistical information on the loading of objects into the buffer pool and the locating of objects in the buffer pool. This information also serves as an indicator of buffer pool performance.

To invoke Buffer Pool Load/Locate Statistics

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
L
```

  Or:
  Enter the following SYSBPM direct command:

```
DISPLAY LOAD
```

  The **Buffer Pool Load/Locate Statistics** screen appears.

The statistics displayed on the **Buffer Pool Load/Locate Statistics** screen are snapshots of the buffer pool which are refreshed every time you press ENTER.

The following information is displayed on the screen:

Field	Explanation
Total Locate Calls	The total number of object location calls; that is, the total number of times the Natural buffer pool manager was requested to search the buffer pool for an object. If the location is successful, the object has been found in the buffer pool or the BP cache and need not be loaded from a Natural system file thereby saving calls and I/Os.
Total Locate Calls - successful	The total number of successful Locate calls as an absolute number.

Field	Explanation
Total Locate Calls - failed	The total number of Locate calls that failed.
Quick Locate Calls	The total number of Quick Locate calls. A Quick Locate call is the attempt to locate an object in the buffer pool by using the internal fast locate table. For further information, see *Internal Fast Locate Table*. In this case, the library (name, database ID and file number) and the position of the object in the buffer pool of the last successful locate call for this object is used again to locate the object. This is the most efficient way to locate an object.
Quick Locate Calls - successful	The number of Quick Locate calls that have been successfully performed.
Quick Locate Calls - failed	The number of Quick Locate calls that failed. A failed Quick Locate call indicates that an object could not be located at its previous position in the buffer pool by using the internal fast locate table (see the relevant section in *Performance Considerations*). This occurs if the object has either been deleted from the buffer pool or was moved to another position. A Quick Locate call that failed results in a Normal Locate call but without a steplib search since the Natural runtime remembers the library that contains the object.
Normal after Quick	The number of Normal Locate calls that result from **Quick Locate Calls - failed**. The value of **Normal after Quick** is always identical to the value of **Quick Locate Calls - failed**.
Normal after Quick - successful	The number of successful Normal Locate calls that result from **Quick Locate Calls - failed**. A Normal after Quick call is successful if the requested object is still in the buffer pool but at another position.
Normal after Quick - failed	The number of failed Normal Locate calls that result from **Quick Locate Calls - failed**. A failed Normal after Quick call indicates that the requested object is no longer available in the buffer pool. As a result, the object is reloaded from the system file by using the library entry in the internal fast locate table (see the relevant section in *Performance Considerations*).

Field	Explanation
Normal Locate Calls	The total number of Normal Locate calls. A Normal Locate call is the attempt to locate an object in the buffer pool without using the internal fast locate table (see the relevant section in *Performance Considerations*). A Normal Locate call always occurs if an object is referenced for the first time within a Natural session after a `LOGON` system command has been performed. The Natural runtime does not yet know the library where the object resides, compared with a Quick Locate call.
Normal Locate Calls - successful	The number of Normal Locate calls that were successful in locating the required object in the buffer pool.
Normal Locate Calls - failed	The number of Normal Locate calls that failed. A failed Normal Locate call indicates that an object (identified by its name and the library where it resides) could not be located in the buffer pool. A failed Normal Locate will either result in a load from the BP cache or a system file, or a Normal Locate call for the next library in the steplib chain.
STEPLIB Searches	The number of **Normal Locate Calls** that occurred from failed attempts to find an object in a steplib library. Normal Locate calls that perform unsuccessful steplib searches do not result in the load of an object from the BP cache or the system file. **STEPLIB Searches** does *not* count Locate calls for objects that are neither contained in the current library nor any steplib or system file (error message `Invalid command, or program does not exist in library`). Locate calls that fail due to incorrect programming add to the counter of **Normal Locate Calls - failed**. The number of **STEPLIB Searches** is calculated by using the following formula: **Normal Locate Calls - failed** - (**Number Loads into BP - Normal after Quick - failed**) The fewer the number of **STEPLIB Searches**, the better the buffer pool is performing. See also *Searching in Steplibs*.

Field	Explanation
Number Loads into BP	The number of times a load into the buffer pool was performed successfully. The load into the buffer pool (storage allocation request) can be triggered either by a load from the database or by a load from the BP cache.
Loads from Cache	The total number of successful Locate calls of objects that resided in the BP cache. This information is counted only if the previous Locate call (**Normal after Quick - failed** or **Normal Locate Calls - failed**) failed. It indicates the number of database loads saved. This means, that, without the BP cache, the object would have to be loaded from the database.
Loads from DB	The number of times an object was loaded from a Natural system file into the buffer pool. As several load calls may be necessary to load a single object, this value provides the actual number of object loads made since the most recent buffer pool refresh. When loading an object, the buffer pool manager uses different search algorithms: see *METHOD=S* and *METHOD=N* in the *Operations* documentation.
Loads from DB - finished	The number of object loads that finished successfully. An object load cannot finish if the load operation is canceled due to any of the following reasons: ● A concurrent object load occurred: see **Loads from DB - concurrent**. ● During the object load, an Adabas response code occurs. ● During the object load, a SYSBPM delete operation is executed for this object.

Field	Explanation
Loads from DB - concurrent	The number of object loads that have been performed simultaneously for the same object: Concurrent object loads occur if two or more Natural sessions that run simultaneously request the same object. While an object is being loaded by one session, other sessions request the same object and start loading it before a session has finished loading. In this case, the same object is loaded more than once. The first session that finishes loading the object will mark the object of the other sessions to be deleted from the buffer pool. The other sessions will then stop loading the object, remove the object marked for deletion from the buffer pool and use the object loaded successfully by the first session. The numbers of objects calculated by the counters **Loads from DB - finished** and **Loads from DB - concurrent** are usually identical. The numbers only differ if the concurrent load is only detected after both sessions have finished the load.
Load Calls	The total number of load calls made since the buffer pool has been refreshed. The load calls are correlated with the access to the system file from which the objects are read. The number of system file accesses is calculated as follows: ● Adabas system file: The number of **Load Calls** plus the number of **Loads from DB** (see above). The total number does not include Adabas RC calls. ● VSAM system file: The number of **Load Calls**.
Number Loads BP 2nd	This field is displayed if `METHOD=S` (selection process) is used as the search method for allocating storage. This field shows the number of times a storage allocation request satisfied the search criteria of Algorithm 2 as described in *METHOD=S* in the *Operations* documentation.
Number Load Cycles	This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation. This field indicates the number of times a search has been performed starting from the top of the buffer pool. This number gives an estimate of the frequency of cycling through the buffer pool in a wrap-around fashion.

Field	Explanation
Last Cycle Start	This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation. The time and date when **Number Load Cycles** was last increased.
Number Lock Retries	This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation. This field indicates the number of times a chain of locked buffer pool entries had to be unlocked, because they could not satisfy the allocation request.
Largest Alloc (TR)	The largest single allocation size so far requested, specified in number of text records.
Number Load Failure	The total number of times an object load failed. The reason for a failure is that either all directory entries are in use at the time of the load request or not enough storage is available in the text record section to perform the load.
Number Load Failure - Sizes failing last	The number of text records that would have been required by the three most recent storage allocation requests that failed.

For detailed information on the search methods used for allocating space in the buffer pool, see *Buffer Pool Search Methods* in the *Operations* documentation.

Buffer Pool Fragmentation

This function provides an overview of the buffer pool fragmentation; that is, an overview of how many different Natural objects occupy how many text records, and how the object locations are spread over the buffer pool.

To invoke Buffer Pool Fragmentation

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
F
```

Or:
Enter the following SYSBPM direct command:

```
DISPLAY FRAGMENTATION
```

The **Buffer Pool Fragmentation** screen appears.

Some of the fields provided on the **Buffer Pool Fragmentation** screen are identical to the items explained in *General Buffer Pool Statistics*:

Buffer Pool Size

Buffer Pool Address

Text Record Section

Text Record Size

Number of Text Records (corresponds to **Text Records - Total**)

In addition, the screen displays a diagram which shows how many different individual objects occupy how much text record size.

For example:

```
                1---+----10---+----20---+----30---+----40---+----50
  005F0480  .._.+***______++  ....**_.+**_.++_ *..+**+__++++XX
```

Each symbol in the diagram represents one text record, and each sequence of equal symbols represents a different individual object occupying one or more text records. The symbols have the following meaning:

_ and .	Objects with a **Current Use Count** (see *Directory Information*) of 0 (zero).
+ and *	Objects with a **Current Use Count** greater than 0 (zero).
blank character	An unused text record.
XX	The end of the buffer pool, which means that no further text records are available.

In the example above, the buffer pool contains 48 text records. Three of them are not in use; the rest is occupied by 24 different objects, 12 of them with a **Current Use Count** of 0 (zero), and 12 with a **Current Use Count** greater than 0.

Internal Function Usage

This function provides statistical information on the calls made to the Natural buffer pool manager.

To invoke Internal Function Usage

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
I
```

Or:
Enter the following SYSBPM direct command:

```
DISPLAY FUNCTION
```

The **Internal Function Usage** screen appears.

The statistics displayed on the **Internal Function Usage** screen are snapshots of the buffer pool which are refreshed every time you press ENTER.

The field **Total Calls** shows the overall number of all internal calls that have been made to the buffer pool manager.

Internally, the buffer pool manager can be invoked for various different functions. For each function, the number of times it has been invoked is displayed, both as an absolute number and as percentage. In addition, these numbers are represented in a horizontal bar chart.

Buffer Pool Hash Table Statistics

This function only applies to buffer pools of the type Natural.

Buffer Pool Hash Table Statistics displays statistics about hash table slots and collisions per slot. The statistics determine the efficiency of the hash algorithm used.

For further information on hash tables, refer to *Buffer Pool Hash Table* in the *Operations* documentation.

The statistics are primarily intended for internal use by Software AG personnel only.

To invoke Buffer Pool Hash Table Statistics

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
H
```

Or:
Enter the following SYSBPM direct command:

```
DISPLAY HASH
```

The **Hash Table Collisions** screen appears.

The statistics displayed on the **Hash Table Collisions** screen are snapshots of the hash table which are taken every time you press ENTER. The following information is displayed:

Field	Explanation
Total Number of Slots	The total number of hash table slots; that is, the total possible entries in the hash table that link the object names with the location of the objects. The number of slots, that is, the size of the hash table will be calculated internally depending on the number of text records.
Number of Slots used	The number of slots in the hash table that have at least one object name mapped to them.
Number of Slots free	The number of slots in the hash table that have no object name mapped to them.
Max. Collisions per Slot	The maximum number of collisions of any slot. The maximum number of collisions is the longest possible search path for an object. A collision is caused if the name of two different objects is mapped to the same slot by the hash algorithm. In this case, a collision resolution is used in order to find another slot.
Collisions	The number of current collisions. Depending on the collisions that occur, the table contains up to 10 rows: `0` No collision. `1` 1 collision. `2` 2 collisions. `3` 3 collisions. `4` 4 collisions. `5` 5 collisions. `6 - 10` Between 6 and 10 collisions. `11 - 15` Between 11 and 15 collisions. `16 - 20` Between 16 and 20 collisions. `21` More than 21 collisions. No collision means that only one object name is mapped per slot. To locate this object, you need to access the hash table once only. If the number of collisions is greater than `0` (zero), for example, *x*, *x*+1 object names are mapped to the same slot. To locate one of these objects, you need to access the hash table up to *x*+1.

Field	Explanation
Number of Slots	The number of slots related to the number of collisions. In addition, the percentage of these slots related to all slots used is displayed.
Number of Slots Totaled	The same values as **Number of Slots**, but the values are totaled.

Example of Hash Table Statistics

```
14:36:26             ***** NATURAL SYSBPM UTILITY *****             2003-08-13
BPNAME NATGBP       - Buffer Pool Hash Table Statistics -      Type Global Nat
BPPROP OFF                                                       Loc DAEF QA41

Total Number of Slots ..           523
 Number of Slots used ..           475 (  90.8 %)     Max. Collisions
 Number of Slots free ..            48 (   9.1 %)       per Slot ..... 7

     Collisions           Number of Slots   Number of Slots Totaled
              0              0 (   0.0 %)              0 (   0.0 %)
              1            164 (  34.5 %)            164 (  34.5 %)
              2            194 (  40.8 %)            358 (  75.3 %)
              3             96 (  20.2 %)            454 (  95.5 %)
              4             16 (   3.3 %)            470 (  98.9 %)
              5              4 (   0.8 %)            474 (  99.7 %)
          6 - 10             1 (   0.2 %)            475 ( 100.0 %)




Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Last        Flip                              Canc
```

Performance Hints

This function only applies to a buffer pool and a BP cache of the type Nat (Natural).

The **Performance Hints** function provides statistical information about workloads on a Natural buffer pool and a BP cache including ratings of the overall performance.

Related Topic:

- *Performance Considerations*

This section covers the following topics:

- Invoking Performance Hints
- Evaluating Performance Hints

Invoking Performance Hints

This section describes how to invoke the **Performance Hints** function and the statistics field displayed on the **Performance Hints** screen.

To invoke Performance Hints

- In the **Buffer Pool Statistics** menu, enter the following function code:

```
P
```

Or:
Enter the following SYSBPM direct command:

```
DISPLAY PERFORMANCE
```

A **Performance Hints** screen similar to the example below appears:

```
13:27:16             ***** NATURAL SYSBPM UTILITY *****             2005-05-19
BPNAME QA41GBP             - Performance Hints -               Type Global Nat
BPPROP OFF                                                       Loc DAEF QA41
                                                               Preload QA41GBPL

                                                        Rating
                                                  (1=best - 6=worst)
Buffer Pool
             Locates / Loads Ratio ......       162.85       3
             Wrap Time Last (hh:mm:ss) ..     00:06:29       4
             Wrap Time Avg (hh:mm:ss) ...     00:01:22       5

BP Cache
             Object Reuse Factor ........         3.11       4
             Wrap Time Last (hh:mm:ss) ..     02:02:29       1
             Wrap Time Avg (hh:mm:ss) ...     00:32:06       3
             Get / Search % .............        74.48 %



Command ===>
Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
      Help        Exit  Last        Flip                                Canc
```

The fields on the **Performance Hints** screen provide the following information:

Field	Explanation
Buffer Pool - Locates / Loads Ratio	The ratio of **Total Locate Calls - successful** to **Loads from DB**. A value greater than 1 indicates that Natural located more objects in the buffer pool than it loaded from the system file. This ratio serves as a buffer pool efficiency indicator. The larger the number, the better the buffer pool is performing. This is the primary indicator of performance from one buffer pool session to the next.
Buffer Pool - Wrap Time Last	This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation. The time in hours, minutes and seconds (*hh***:***mm***:***ss*) since the buffer pool was last completely reused (wrapped around). Objects are loaded into the buffer pool one after the other (in sequential order) filling the top of the buffer pool first and the bottom last. When the bottom of the buffer pool has been reached, the buffer end is wrapped around and the next object is loaded again at the top of the buffer pool. When the buffer pool has been filled completely for the first time, a new object loaded into the buffer overwrites an object that was loaded in the previous wrap-around cycle and that is not currently locked (marked as resident or in use). Every object loaded into the buffer pool is assigned a directory entry that contains information such as the name of the object, the library where it is stored and the **BP Load Time** time stamp of the loading into the buffer pool. **Wrap Time Last** is evaluated each time an object is loaded into the buffer pool. **Wrap Time Last** indicates the time period between the last loading of an object (**BP Load Time**) and the last overwriting of an object. The longer the period of a wrap-around cycle, the better the buffer pool is performing. This period can vary considerably at night or over a weekend when user traffic is low compared with normal working hours.

Field	Explanation
Buffer Pool - Wrap Time Avg	This field is displayed if `METHOD=N` (next available) is used as the search method for allocating storage as described in the *Operations* documentation. The average time in hours, minutes and seconds (*hh***:***mm***:***ss*) of one wrap-around cycle since the buffer pool was initialized or refreshed. **Wrap Time Avg** is calculated by dividing the life time of the buffer pool by the number of wrap-around cycles. **Wrap Time Last** compared with **Wrap Time Avg** shows whether the buffer pool is currently being used more frequently than on average.
BP Cache - Object Reuse Factor	The ratio of objects swapped out of the BP cache (Get calls) to objects swapped into the BP cache (Put calls). The value is calculated by dividing successful Get calls by successful Put calls. It shows the overall reuse factor; that is, how often an object loaded once into the BP cache could be successfully reloaded into the buffer pool. The higher the value, the better the BP cache efficiency. Example of an **Object Reuse Factor**: A ratio of `5.70` indicates that an object loaded in the BP cache was swapped into the buffer pool 5.7 times on average.

Field	Explanation
BP Cache - Wrap Time Last	The time in hours, minutes and seconds (*hh***:***mm***:***ss*) since the BP cache was last completely reused (wrapped around). Objects are loaded into the BP cache one after the other (in sequential order) filling the top of the BP cache first and the bottom last. When the bottom of the BP cache has been reached, the end of the BP cache is wrapped around and the next object is loaded again at the top of the BP cache. When the BP cache has been filled completely for the first time, a new object loaded into the buffer overwrites the object that was loaded in the previous wrap-around cycle. Each object loaded into the BP cache is assigned a directory entry that contains information such as the name of the object, the library where it is stored and the **BPC Load Time** time stamp when it was loaded into the BP cache. **Wrap Time Last** is evaluated every time an object is loaded into the BP cache. **Wrap Time Last** indicates the time period between the last loading of an object (**BPC Load Time**) and the last overwriting of an object. The longer the period of a wrap-around cycle, the better the BP cache is performing. This period can vary considerably at night or over a weekend when user traffic is low compared with normal working hours.
BP Cache - Wrap Time Avg	The average time in hours, minutes and seconds (*hh***:***mm***:***ss*) of one wrap-around cycle since the BP cache was started. **Wrap Time Avg** is calculated by dividing the life time of the BP cache by the number of wrap-around cycles. **Wrap Time Last** compared with **Wrap Time Avg** shows whether the BP cache is currently being used more frequently than on average.

Field	Explanation
BP Cache - Get / Search %	The percentage of objects returned from the BP cache successfully (Get calls) compared with the total number of search requests (Search calls) the buffer pool sent to the BP cache. This value indicates the percentage of objects the buffer pool could load from the BP cache, instead of the Natural system file FNAT or FUSER. The higher the value, the better the cache efficiency. Keep in mind that a search for an object in a chain of steplibs may increase the number of Search calls the buffer pool sends to the BP cache. However, these calls will not result in a successful Get call or a load from the Natural system file, because an object may not be found in a steplib. **Get / Search %** does not consider the search through a long chain of steplibs that is frequently used. We recommend that you use as few steplibs as possible to increase overall performance. Example of a **Get / Search %** value: A value of `70%` indicates that 70 % of all objects loaded into the buffer pool were retrieved from the BP cache and 30% were loaded from the system file.

Evaluating Performance Hints

The statistical values provided on the **Performance Hints** screen are the basis for a performance rating system where 1 denotes best (highest) and 6 denotes worst (lowest) performance.

The ratings shall give you an idea of how a buffer pool or a BP cache performs with the sized specified for them. The rating values should suit the requirements of most system environments.

There are environments where it has proven to be worth having a BP cache that is five times as big as the buffer pool. If ratings tend to be bad, the size of the buffer pool or BP cache should be increased. However, the size of a buffer pool or a BP cache can guarantee good performance even though ratings are bad. If the ratings of a BP cache are good, bad ratings for the buffer pool do not carry so much weight. For environments with extreme workloads, the ratings can, however, be useful indicators for when the size of the buffer pool or the BP cache should be changed.

The statistical values displayed on the **Performance Hints** screen are snapshots of the Natural buffer pool and BP cache which are refreshed each time a wrap-around occurs. Buffer pool and the BP cache should run for some time and the statistics values reach a certain extent to obtain meaningful results from these values. For example, if the size of a buffer pool is so large that only very few BP cache calls are required, the statistics regarding the BP cache will not be meaningful.

When evaluating the statistics you should also consider the type of system environment (for example, production versus test), the type of applications using the buffer pool (batch, online, user-defined or system), user traffic (peak hours or normal hours) and extraordinary operational factors.

PF Keys and Direct Commands

On the buffer pool statistics screens, you can use the PF keys or SYSBPM direct commands listed in the table below. An underlined portion of a command represents its minimum abbreviation. For further commands, see *SYSBPM Direct Commands*.

PF Key	Command	Function
PF1		Provides SYSBPM help information: see also *Online Help*.
PF3	EXIT	Leaves the current function/screen and displays the previous screen.
PF4	LAST	Displays the SYSBPM direct command entered most recently.
PF6	FLIP	Switches the PF-key line: toggles between the display of PF1 to PF12 and PF13 to PF24.
PF8 (Load)	DISPLAY LOAD	Only applies to the screen **General Buffer Pool Statistics**. Displays the **Buffer Pool Load/Locate Statistics** screen.
PF8 (Gen)	DISPLAY GENERAL	Only applies to the **Buffer Pool Load/Locate Statistics** screen. Displays the **General Buffer Pool Statistics** screen.
PF12	CANCEL	Same as EXIT.
PF15	MENU	Invokes the SYSBPM **Main Menu**.