OpenEdge Revealed: Mastering the OpenEdge Database with Fathom™ Management

Adam Backman

Expert Series

August 2004

Product Code: 4009

Item Number: 101211

Release: V3.0A

Contents

Figures

Tables

Examples

Preface

This Preface contains the following sections:

Overview

This manual documents some of the best practices for maintaining your Progress® OpenEdge™-based application. An application is composed of many distinct pieces that interact with each other. In very general terms, an application contains the following elements:

- Hardware:
  - Disks — The physical storage of the system.
  - Memory — The link between the disks and the CPU.
  - CPU — The computing engine of the system.
- Operating system
- Software

Purpose

This manual begins by discussing the internals of the system, such as disks and memory, and builds on this information to explain the OpenEdge database and how it uses these resources. This manual then focuses on to how to best manage your system as a whole. First is the heart of the system, with a discussion of hardware best practices. Then, the internal database resources are detailed, emphasizing the administration of the system and the application as a whole, including the applicability of using Fathom™ Management as a solution to these issues. Finally, installation options and best practices for Fathom Management, as well as some basic troubleshooting hints, are discussed.

Fathom Management with OpenEdge or Progress

Fathom Management Version 3.0A runs against the following:

- OpenEdge 10.0B.
- Progress Version 9.1D and the 9.1D09 service pack.

For the sake of simplicity, the procedures and screen shots provided in this manual refer to running Fathom against OpenEdge 10.0B. However, be assured that unless indicated otherwise, the procedures are the same for both Progress Version 9.1D with the 9.1D09 service pack and OpenEdge 10.0B. For example, if a procedure refers to an OpenEdge database, the procedure applies to a Progress database as well.

Audience

This manual is a guide for both database and system administrators.

How to use this manual

The elements of your system need to be treated as a whole rather than a sum of the individual parts; a modification in one area often affects other areas. This manual discusses each component individually, and then discusses each component's interaction with the system as a whole. You should read this manual from cover to cover **before** making any changes to your system.The techniques discussed in this manual could cause problems instead of fixing them if the tools are used in isolation from the bigger picture. Some details have been included to provide a greater understanding of the inner workings of the OpenEdge database; however, the goal is not to document OpenEdge internals, but rather to provide information that can aid in making design decisions and avoiding pitfalls.

Organization

This manual contains the following chapters:

Chapter 1, “Managing System Resources”

Details the pertinent issues regarding the hardware resources of a typical system. It provides configuration recommendations and information concerning what to look for to anticipate problems before they occur.

Chapter 2, “Managing OpenEdge Database Resources”

Discusses in detail database internals and configuration considerations.

Chapter 3, “Performing System Administration”

Ties together concepts learned in the previous chapters. It details how to provide database availability, resiliency, and performance, and consideration for periodic maintenance tasks such as annual backups, data archiving, and schema changes.

Chapter 4, “Guidelines for Applying Fathom”

Discusses various install and configuration issues and provides best practices for Fathom Management.

“Glossary”

Contains definitions of database and Fathom Management terms.

Typographical conventions

This manual uses the following typographical conventions:

Convention	Description
Bold	Bold typeface indicates commands or characters the user types, or the names of user interface elements.
Italic	Italic typeface indicates the title of a document, provides emphasis, or signifies new terms.
SMALL, BOLD CAPITAL LETTERS	Small, bold capital letters indicate OpenEdge™ key functions and generic keyboard keys; for example, **GET** and **CTRL**.
KEY1-KEY2	A hyphen between key names indicates a *simultaneous* key sequence: you press and hold down the first key while pressing the second key. For example, **CTRL-X**.
KEY1 KEY2	A space between key names indicates a *sequential* key sequence: you press and release the first key, then press another key. For example, **ESCAPE H**.
Syntax:	
`Fixed width`	A fixed-width font is used in syntax statements, code examples, and for system output and filenames.
`Fixed-width italics`	Fixed-width italics indicate variables in syntax statements.
`Fixed-width bold`	Fixed-width bold indicates variables with special emphasis.
`UPPERCASE fixed width`	Uppercase words are Progress® 4GL language keywords. Although these always are shown in uppercase, you can type them in either uppercase or lowercase in a procedure.
	This icon (three arrows) introduces a multi-step procedure.
	This icon (one arrow) introduces a single-step procedure.

1

Managing System Resources

This chapter describes the various resources used by the Progress® OpenEdge™ database, as well as other applications on your system, and it gives you a greater understanding of each resource's importance in meeting your needs. Each resource is described individually, and various methods for monitoring are discussed. You will also learn about the importance of resource trend analysis. Tracking resource use and availability over time is the only way to obtain a long-term picture of the health of the system and allows the system administrator to plan accordingly.

This chapter discusses the resources in reverse performance order, from slowest (disk) to fastest (CPU):

- Managing disk capacity
- Managing memory usage
- Managing CPU activity

Managing disk capacity

The disk system is the most important resource for a database. Since it is the only moving part in a computer, it is also the most prone to failure. Reliability aside, a disk is the slowest resource on a host-based system, and it is the point where all data resides.

There are three overall goals for a database administrator in terms of disk resources. They are:

- **Quantity** — Have enough disk space to store what you need.
- **Reliability** — Have reliable disks so your data will remain available to the users.
- **Performance** — Have the correct number of disks running at the maximum speed to meet your throughput needs.

These goals sound simple, but it is not always easy to plan for growth or to know what hardware is both reliable and appropriate to meet your needs. With these goals in mind, this section examines the problems that might present roadblocks to fulfilling each of these goals. (For example, it is essential to have enough disk space to meet your storage needs.) The individual portions of the database are described in detail later in this manual.

Ensuring adequate disk storage

The following sections describe how to determine if you have adequate disk storage:

- Understanding data storage.
- Determining data storage requirements.

Understanding data storage

The following is a minimum list of the critical data stored on your system. Used in this context, the term “data” is a more inclusive term than one that defines simple application data. Data can include:

- Databases.
- Before-image files.
- After-image files.
- Application files (OpenEdge, 4GL or SQL code, third-party applications).
- Temporary files.
- Operating systems.
- Swap or paging files.
- Client files.

The term “data” also references other possible data storage requirements, which include:

- A backup copy of the database.
- Input or output files.
- A development copy of the database.
- A test copy of the database.

If this information is already stored on your system, you know (or can determine) the amount of data you are storing.

If you are putting together a new system, planning for these data storage elements can be a daunting task. You will need to have a deep understanding of the application and its potential hardware requirements with little or no data points from which to work. One of the first things you need to know is how many data records you will be storing in each table of your database. Database storage calculations are discussed in Chapter 2, “Managing OpenEdge Database Resources.” However, it is essential to review with the users (or inspect existing databases) to estimate the initial number of records in each table of the database.

Determining data storage requirements

In existing database environments, the first step in determining data storage requirements is to to take an inventory of your storage needs. The types of storage are:

- **Performance oriented** — Database or portions of databases, before-image (BI), after-image (AI).
- **Archival** — Historical data, backups.
- **Sequential or random** — Various RAID types as discussed in the "Increasing disk reliability with RAID" section on page 1–9.

You perform a detailed analysis of your current needs and projected growth to estimate storage requirements. The following sections describe this analysis process.

Determining current storage using operating system commands

To use the operating system to determine current storage usage, use one of the following options:

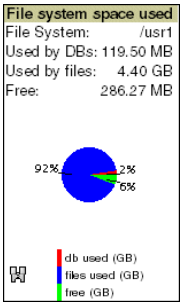
- `df` to determine the amount of free space available on most UNIX systems. There are switches (`-k` to give the result in kilobytes and `-s` to give summary information) that will make the information easier to read.
- `bdf` on HPUX to report the amount of free space available.
- The disk properties option on Windows to provide a graphical display of disk storage information.

Determining current storage using Fathom

Fathom™ Management provides a quick method for determining current storage usage.

To determine current storage usage with Fathom:

1. From the Fathom Management console, select **Resources** from the menu bar.
2. Expand the **File Systems** category in the *detail frame* (the right-hand frame).
3. Select the file system you want more information about. The **File System** page appears. The **File system space used** section provides a detailed, graphical representation of your current file system usage:



Examining your growth pattern

Many companies experience exponential growth in their data storage needs. The business might grow to meet demand or might absorb additional data due to acquisition. The database administrator needs to be “in the loop” when business decisions are being made to be able to plan for growth. Some growth is warranted, but in most cases, a great deal of data is stored that should be archived out of the database and put on backup media. For information that is not directly related to current production requirements, one option is to use a secondary machine with available disk space. This machine can serve as both secondary storage for archival data and as a development or replication server. By moving data that is not mission-critical off the production machine to another location, you can employ less expensive disks on the archival side and more expensive disks for production use.

Moving archival data off the production machine

It is important to fully understand how archived data will be used, if at all, before making any plans to move it off the production system. Some users might want to purge the data, but you should first archive the data so it can be easily recovered, if necessary. Your archive method can be as simple as a tape; however, it is important to remember that you might need to read this tape in the future. It will do you no good if it contains data in a format you can no longer use. You should archive the information to ACSII or some other format that will not depend on third-party software. In particular, if it is OpenEdge data, it is always advisable to archive the data in a standard OpenEdge format.

Examining system storage information trends

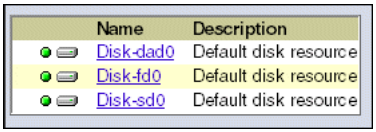
Viewing system storage information and keeping track of it over a period of time can help to determine growth patterns. You can write your own application to track storage information by taking the output from the operating system and placing it in a file or in a database. Or, instead of writing and maintaining your own application, you can use Fathom Management to gather this information. Fathom also allows you to track and report on your collected data.

Before you can begin to collect storage information within Fathom, you need to create a monitoring plan with trending options setup for system disks. See the *Resource Monitoring Guide* for detailed instructions on creating a resource monitoring plan.

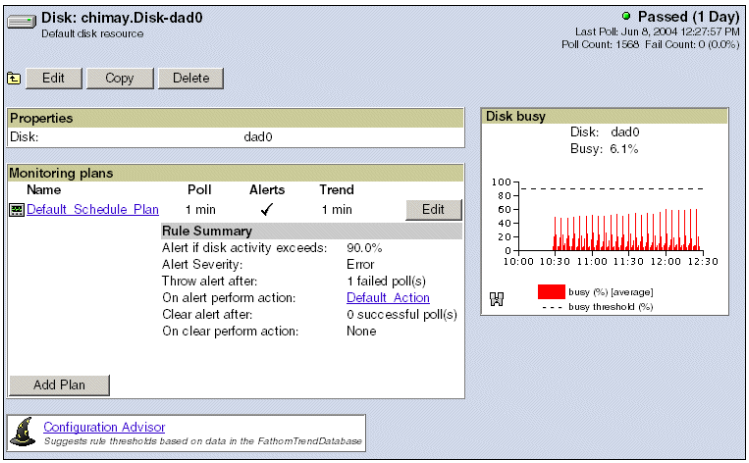
Once you have created a monitoring plan, you can view activity in Fathom.

To view activity in Fathom:

1. From the Fathom Management console, select **Resources** from menu bar.
2. Select **Disks** from the detail frame to display all of the available disks on your system:



3. Select the disk that you want to monitor. The monitoring plan that you have defined for the selected disk appears, as shown in this example:



Note: If the resource is not defined you can define the resource properties and associated monitoring plan now.

4. Repeat Step 1 through Step 3 for each disk you want to monitor.

Comparing expensive and inexpensive disks

Disks are disks are disks, aren't they? No, of course not. When you buy a disk, you are really buying two things: storage capacity and throughput capacity. For example, if you want to buy 72 gigabytes (GB) of capacity, you can purchase either a single 72GB unit or four 18GB units. The storage capacities of these disks are exactly the same, but the four-drive configuration has potentially four times greater throughput capacity. Each drive is capable of doing approximately 100 input/output (I/O) operations per second regardless of its size. The four-drive system has the potential to perform 400 I/O operations per second.

It is generally less expensive to purchase fewer larger disks to get to your desired capacity for a number of reasons. First, the 72GB drive is only marginally more expensive than a single 18GB drive, but buying four 18GB drives to obtain the same capacity as a 72GB drive will substantially increase your cost. Second, you need to have the rack space to hold these drives—more rack space, more cost. Third, you might need more controllers to efficiently run additional disks—adding to the cost.

However, if performance is important, you should have a greater number of physical disk drives to give you the greatest throughput potential. The additional cost of the multi-disk solution is offset by increases in performance (user efficiency, programmer time, customer loyalty, and so on) over time. If you are only using the database as archival storage and you do not care about the performance of the database, fewer large disks are recommended to decrease cost. This approach allows you to store the greatest amount of data on the fewest number of disks.

Determining the location of data on disks

The outer 25 percent of the disk is approximately 10 percent to 15 percent faster than the inner 25 percent. Use this point to your advantage when configuring your disks. Some operating system and disk manufacturers allow you to select which portion of the disk to use for each allocation; others place the first disk partition on the outer portion of the disk and the last on the inner portion. In either case, using the outer portion of the disk provides the greatest performance potential. In large, high-volume environments, production data should only fill the outer 75 percent to 80 percent of each disk. This leaves the innermost portion for static storage, such as backups or maintenance areas.

Understanding cache utilization

Disk vendors provide for caching on their disk arrays to increase performance. However, there is a limit to how effective this cache can be. If your system is doing hundreds of thousands of reads per hour on your system, the cache becomes saturated after a short period of time. Under these conditions, the system quickly degrades to conventional disk speeds. It is important to consider all disks, regardless of manufacturer's published guidelines, as performing at conventional speeds. Otherwise, you will experience a larger problem than you would expect when the cache is saturated.

For example, an index rebuild is a process that saturates the disk cache quickly. This process performs many modifications in a short time frame. The ability of OpenEdge to saturate the disk cache has been demonstrated on disk arrays costing millions of dollars. The cache is nice to have and will provide a benefit, but don't let the benefit sway you from regarding the layout of these disks as identical to any other disk.

Increasing disk reliability with RAID

It is important to understand the terms reliability and performance as they pertain to disks. *Reliability* is the ability of the disk system to accommodate a single- or multi-disk failure and still remain available to the users. *Performance* is the ability of the disks to present information to the users in an efficient manner.

Adding redundancy almost always increases the reliability of the disk system. The most common way to add redundancy is to implement a redundant array of inexpensive disks (RAID).

There are two types of RAID:

- Software

  Software RAID can be less expensive. However, it is almost always much slower than hardware RAID, since it places a burden on the main system CPU to manage the extra disk I/O.

- Hardware

  The most commonly used hardware RAID levels are: RAID 0, RAID 1, RAID 5, and RAID 10. The main differences between these RAID levels focus on reliability and performance as defined earlier in this section.

The following sections discuss the different RAID types:

- RAID 0: Striping
- RAID 1: Mirroring
- RAID 10 Or 1 + 0
- RAID 5

It is possible to have both high reliability and high performance. However, the cost of a system that delivers both of these characteristics will be higher than a system that is only reliable or only efficient.

RAID 0: Striping

RAID 0: Striping has the following characteristics:

- **High performance** — Performance benefit for randomized reads and writes.
- **Low reliability** — No failure protection.
- **Increased risk** — If one disk fails, the entire set fails.

Figure 1–1 illustrates how striping works.

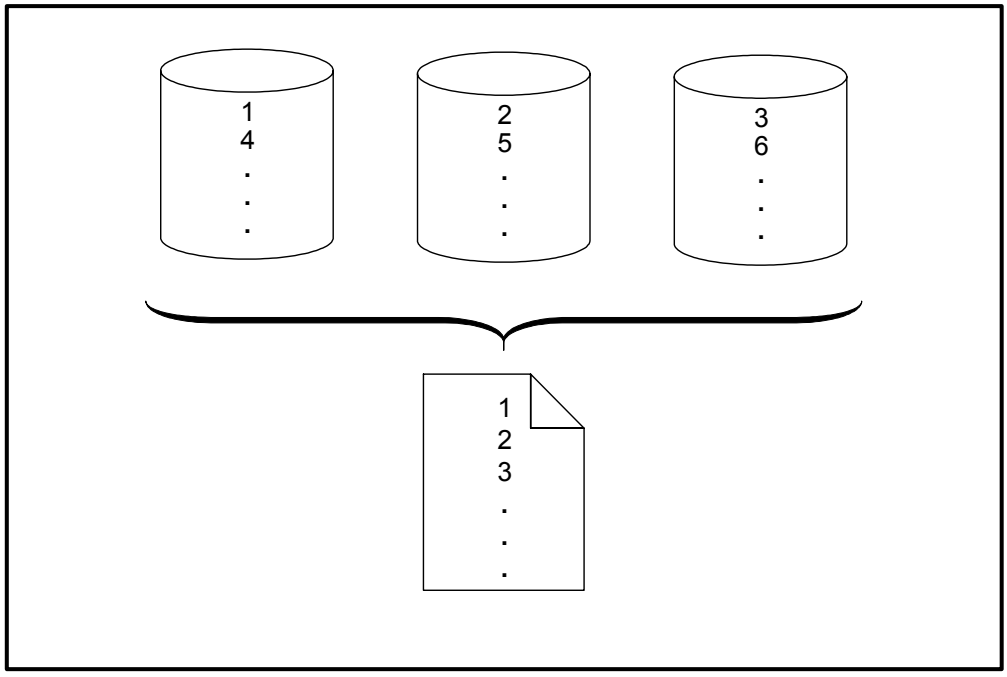


Figure 1–1: RAID 0: Striping

In Figure 1–1 there are three disks. The information that is stored by the user is distributed across all three drives. The amount of information stored on a single device is called a chunk or *stripe*. As the figure shows, the stripes are distributed across the RAID array.

Note: The stripe size is generally tunable, from a small size (8K) to a large size (several megabytes). The ability to tune stripe size varies from vendor to vendor. Reference your vendor's product documentation for details on how to tune this setting.

Testing has shown that 128K and larger stripes are generally the optimal size for performance with an OpenEdge database that has an 8K-block size. (An 8K-block size is generally the most efficient block size for an OpenEdge database). In Windows, these stripes appear to the operating system or to the user as one file system or "drive." In Figure 1–1, each of these disks is assumed to be 128K stripes. Therefore, if a file of 384K crosses all three disks, the disks work together to send information to the user. While this arrangement helps performance, it can cause a potential problem. If one disk fails, the entire file system is corrupted.

RAID 1: Mirroring

RAID 1: Mirroring has the following characteristics:

- **Medium performance** — Superior to conventional disks due to “optimistic read.”
- **Expensive** — Requires twice as many disks to achieve the same storage and twice as many controllers if you want redundancy at that level.
- **High reliability** — Loses a disk without an outage.
- **Good for sequential reads and writes** — The layout of the disk and the layout of the data is sequential, promoting a performance benefit, provided that you can isolate a sequential file to a mirror pair.

Figure 1–2 shows a simple example of mirroring.

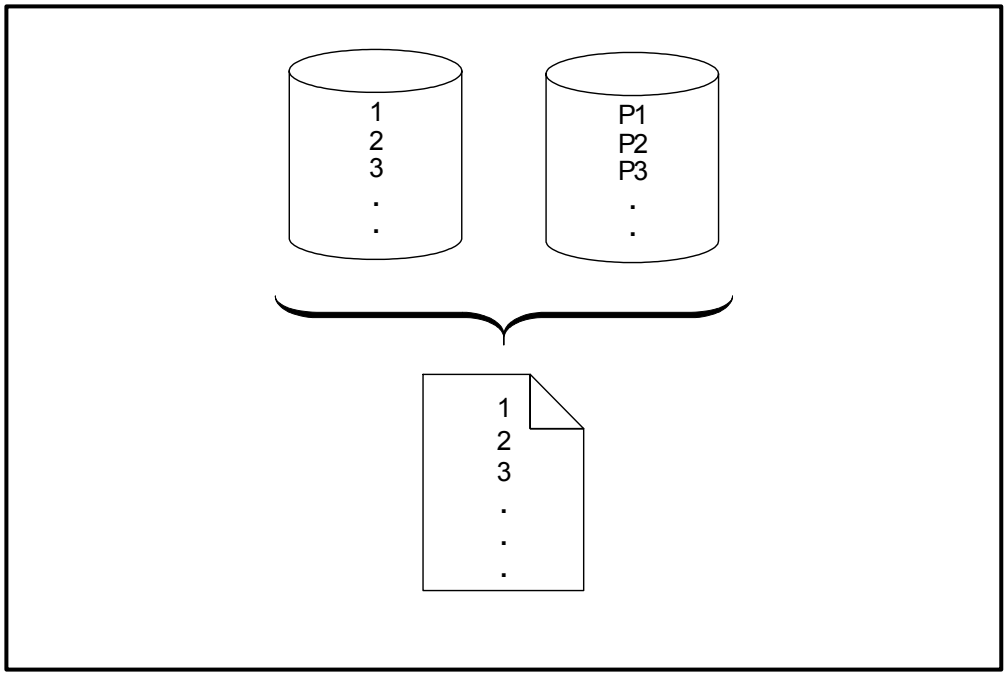


Figure 1–2: RAID 1: Mirroring

In Figure 1–2, the first disk is the primary disk and the second disk acts as the parity or mirror disk. The role of the parity disk is to keep an exact synchronous copy of all of the information stored on the primary disk. If the primary disk fails, the information can be retrieved from the parity disk.

Ensure that you have “hot swappable” disks so repairs can be made without bringing down the system. Most RAID 1 disks are “hot swappable.” Note that there is a performance penalty during the resynchronization period.

Both disks are actually primary, and store parity and data. On a read, the disk that has its read/write heads positioned closer to the data will retrieve information. This data retrieval technique is known as an *optimistic read*. An optimistic read can provide a maximum of 15 percent improvement in performance over a conventional disk. When setting up mirrors, it is important to look at which physical disks are being used for primary and parity information, and to balance the I/O across physical disks rather than logical disks.

RAID 10 Or 1 + 0

RAID 1 + 0, more commonly referred to as RAID 10, has the following characteristics:

- High reliability. Provides mirroring and striping.
- High performance. Good for randomized reads and writes.
- No more expensive than RAID 1 Mirroring.

RAID 10 resolves the reliability problem of striping by adding mirroring to the equation.

Figure 1–3 shows the disks on the left-hand side as primary disks. The primary disks are striped together and then mirrored to the disks on the right-hand side. All four disks are acting as both primary and parity disks.

The performance of this setup is an improvement over most other configurations because RAID 10 supports striping and optimistic reads. This is the preferred configuration for most applications because it provides the highest performance and availability with the lowest maintenance in terms of load balancing.

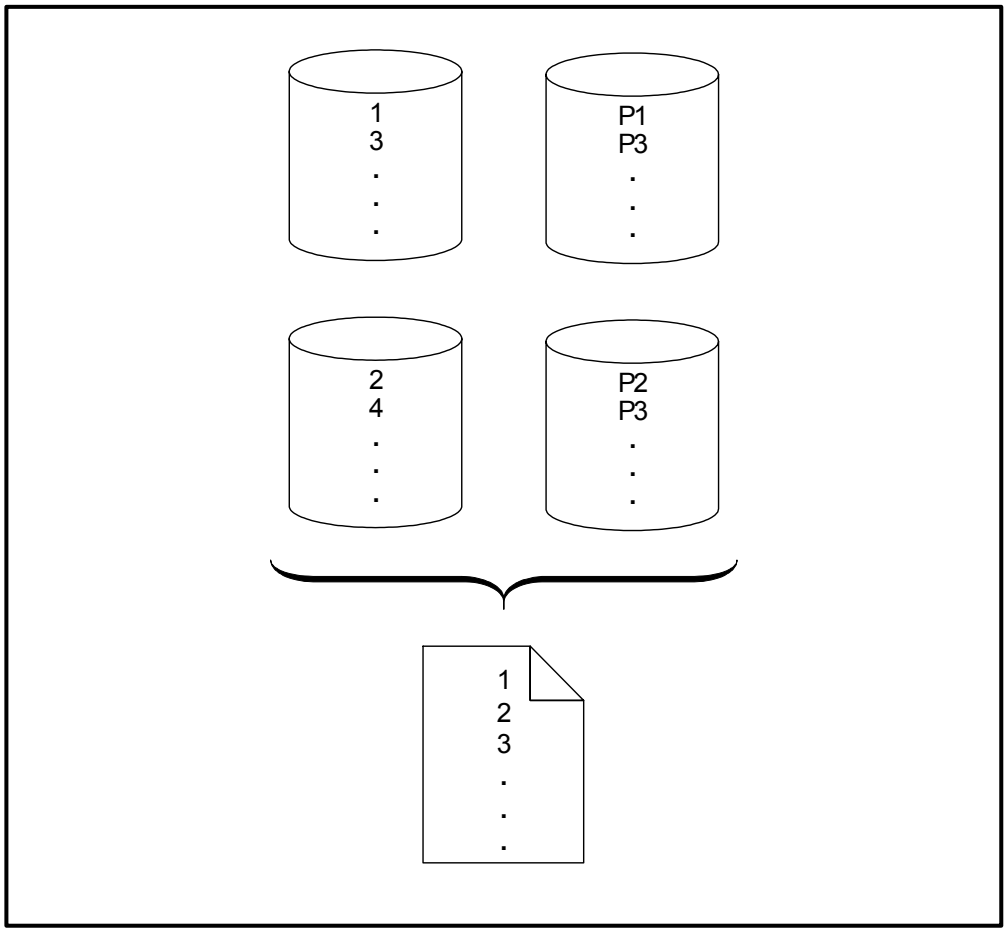


Figure 1–3: RAID 10

RAID 5

RAID 5 has the following characteristics:

- High reliability—provides good failure protection.
- Less expensive than RAID 1.
- Low performance—performance is poor for writes due to the parity's construction.
- Running in an "absorbed" state will provides diminished performance throughout the application because the information must be reconstructed from parity.

RAID 5 is less expensive to install than other RAID types. Also, there is only a 20 percent reduction in storage capacity to store the parity information as compared to a 50 percent reduction for either RAID 1 or RAID 10. However, this initial cost savings will be consumed over the life of the array in lost performance.

The drawback of RAID 5 is magnified during disk writes, which On-Line Transaction Processing (OLTP) does quite often. For every write, the primary information is written and then the parity is calculated and stored. This is generally done through the same controller, so writes are three times as costly as on conventional disk: write primary, calculate parity, and write parity.

A mirrored system is commonly configured with two controllers to enable dual writes to occur in parallel. If your application is 100 percent read, you might want to consider RAID 5. Otherwise, RAID 1 or RAID 10 would be the better way to protect your data.

Claims that the RAID 5 are "just as fast" as RAID 1 or RAID 10 should be closely evaluated. In benchmarks or low-volume situations that might not adequately test a RAID 5 type, the performance wall is generally not encountered. However, with actual production usage, when you exceed the ability of cache to meet your needs, you will hit this wall very hard.

Note: Progress Software Corporation recommends against using RAID 5 with OpenEdge databases when performing updates during online transaction processing.

Figure 1–4 shows a five-disk RAID 5 array. Notice how the parity information, preceded by the letter “P,” is interleaved with the primary data throughout all the disk. If you experience a disk failure, you will not lose data. However, your performance will be severely degraded because information from the lost disk will have to be extracted from calculated parity.

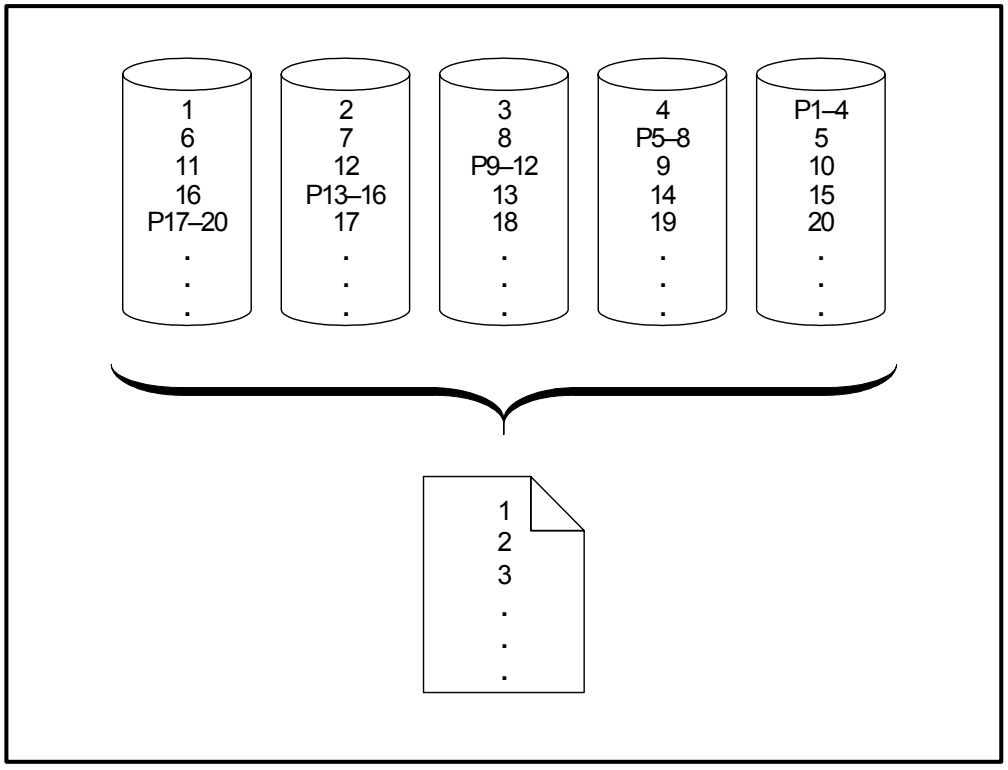


Figure 1–4: RAID 5

OpenEdge in a network storage environment

OpenEdge supports reliable storage of data across the network, provided your operating system complies with the rules of NFS Protocol Version 3.0. Though this environment is supported, it is not recommended because there can be significant performance issues with this configuration.

Storage Area Networks (SANs) are becoming more popular than NFS Protocol. You can purchase one storage device and attach it to several hosts through a switching device. The same rules apply in a SAN environment as they do in a conventional or RAID environment. The SAN maintains database integrity. Multiple machines have access to the disks in a SAN. Therefore, you need to be very aware of the drive/controller usage and activity of other hosts using the SAN. This heightened awareness is due to the SAN's multi-machine support.

See your SAN vendor's documentation regarding monitoring to see if they provide an interface into the activity levels of the disks in the array.

Summary

Disks are your most important resources. Begin by purchasing reliable disk array systems, configure them properly to allow consistent, fast access to data, and monitor them for performance and fill rate. Remember to monitor them and keep track of their fill rate so you do not run out of space. Additionally, you should trend the consumption of storage space to allow planning time for system expansion.

Managing memory usage

The primary function of system memory is to reduce disk I/O. Memory speed is orders of magnitude faster than disk speed. From a performance perspective, reading and writing to memory is much more efficient than to disk. Memory is not a durable storage medium. Long-term storage on memory is not an option. There are RAM disks that do provide durable data storage; however, they are cost prohibitive for most uses.

While memory is fairly reliable, it is not infallible. Normally, if one memory chip is bad, it will not compromise the system.

One way to improve memory reliability is to configure your system without interleaved memory. *Interleaved* memory is like disk striping. It is good for performance but bad for reliability. If one chip fails, all of interleaved memory is compromised. Therefore, for high availability systems, it is best to avoid interleaved memory. The trade off of performance versus high availability is not a good choice for any business activity that requires high system availability. There will be exceptions to this rule as vendors devise ways of adding redundancy at the memory level.

Maximizing memory management includes:

- Allocating memory for the right tasks.
- Having enough memory to support your needs.

OpenEdge-specific memory allocation is covered later in this manual.

How memory works

To understand how to allocate memory, you need to understand how memory works. There are older and newer memory models.

Older systems employ the concept of swapping to manage memory. *Swapping* is the process of taking an entire process out of memory, placing it on a disk in the “swap area” and replacing it with another process from disk. Newer systems swap only in extreme, memory-lean situations. Swapping is very performance-intensive and should be avoided at all costs.

New systems manage memory with paging. There are two types of paging:

- Physical

  Physical paging identifies when information is needed in memory. Information is retrieved from disk (paging space).

- Virtual

  Virtual paging occurs when information is moved from one place in memory to another.

Both kinds of paging occur on all systems. Under normal circumstances, virtual paging does not degrade system performance to any significant degree. However, too much physical paging can quickly lead to poor performance.

How much paging is too much?

The answer to this question varies according to these elements: hardware platforms, operating systems, and system configurations. Because virtual paging is fairly inexpensive, a significant amount can be done with no adverse affect on performance. Physical paging will usually be high immediately after booting the system, and it should level off at a much lower rate than virtual paging. Most systems can sustain logical paging levels of thousands of page requests per second with no adverse effect on performance. Physical paging levels in the thousands of requests would be too high in most cases. Physical paging should level into the hundreds of page requests per second on most systems.

If physical paging continues at a high rate, then you need to adjust memory allocation or install more memory. It is important to remember that these numbers should only be used as guidelines because your system might be able to handle significantly more requests in both logical and physical page requests with no effect on performance. This underscores the need to baseline your system, as outlined in Chapter 3, “Performing System Administration,” to establish the right levels of activity for your system.

Estimating memory requirements

To determine memory usage needs, perform an inventory of all of the applications and resources on your system that use memory. For OpenEdge-based systems, these processes include:

- Operating system memory.
- Operating system required processes.
- Operating system buffers.
- OpenEdge-specific memory.
- OpenEdge executables.
- Database broker.
- Remote client servers.
- Other OpenEdge servers.
- Client processes (batch, self-service, and remote).

Operating system memory estimates

Operating system memory usage varies from machine to machine. However, the following list identifies reasonable memory estimates:

- Small systems—range from 32MB to 64MB.
- Large systems—range from 128MB and higher.

Generally, a small system has fewer than 100 users and a large system has hundreds of users. This number varies because there are systems that have few users, but many processes that require a large system to run the application.

Operating system buffers are generally a product of how much memory is in the machine. Most systems will reserve 10 to 15 percent of RAM for operating system buffers. Operating system buffers are tunable on most systems. Review your operating systems vendor’s product documentation for details.

OpenEdge-specific memory estimates

OpenEdge uses demand-paged executables. *Demand-paged executables,* also known as *shared executables,* reserve text or static portions of an executable that is placed in memory and shared by every user of that executable. For brokers and servers, the dynamic or data portion of the executable is stored in memory (or swap/paging files) for every user or instance of the executable.

Other OpenEdge memory allocation is estimated based on the number of users and a some of the startup parameters for the brokers.

Broker parameters

To estimate the amount of memory used by the database broker, add 10 percent to the database buffers parameter (`-B`). However, if you have a high value for lock table entries (`-L`) or index cursors (`-c`) you will need to increase this estimate. Record locks consume 14 to 18 bytes each and index cursors consume 64 bytes each. Also, if you have a very low setting for database buffers (less than 2000), the overhead for the other parameters will be greater than 10 percent.

For example, if database buffers (`-B`) are set to 20,000 on an 8KB-block-size database, you allocate 160,000KB in database buffers. If you add 10 percent to this amount, your total allocation will be approximately 176,000KB, or 176MB for the database broker.

Remote client servers are fairly straightforward to estimate, as each server will use approximately 3MB to 5MB. The number of remote client servers is limited by the `-Mm` parameter. The default number is 4.

Client or user parameters

Client processes will vary, depending on the startup options chosen. However, with fairly average settings for `–mmax` and `–Bt`, the new memory allocated per process will be 5MB to 10MB. This range applies to application server processes, too. Remote users generally use more memory (10MB to 20MB per process) because they require larger settings for `–mmax` and `–Bt` to provide acceptable performance across the network. The memory requirements for a remote user (that is, `–mmax` and `–Bt` settings) do not impact the memory requirements on the host.

Creating a budget for memory

Here is an example of a machine with 1GB of RAM, 50 local users, and one 8KB-block-size database using 10,000 database buffers.

Operating system memory:

- 28MB OS.
- 100MB OS buffers.

OpenEdge memory:

- 16MB executable.
- 88MB database broker ((8KB * 10000) * 1.1).
- 250MB to 500MB for users.

Total memory requirement: 582MB to 832MB.

The system can run without significant paging or swapping, allowing you to use the additional memory for other applications or to further increase the memory utilization for OpenEdge by increasing database broker parameters, like `-B`. Once the broker is as efficient as possible, you can look into increasing local user parameters like `-mmax`.

In many cases there will be third-party applications running on the system, too. Consider the memory used by these additional applications to accurately determine memory estimates.

Trending analysis

The key to good capacity planning is conscientious, well-structured trending analysis. You need to look at your system throughout the day and throughout the week to see when you are hitting peak memory usage. Once this peak is established, watch the system more closely at this peak time to determine specifically where memory is being used.

There are several ways to write your own trending application and several applications that will allow you to look at memory over time to ensure that you have enough memory to run your business now and in the future. Various operating system utilities allow you to examine, in real time, how much memory the system is using. For example, the UNIX `sar` command allows you to accumulate this information over time. On Windows, the Performance Monitor utility displays current information about various system resources including memory. To run the Performance Monitor utility, type **perfmon** at the command prompt.

The amount of "free memory" available is not a helpful statistic. Most operating systems will not reclaim memory until there is little or no free memory. The absence of free memory does not indicate a problem. It is important to look at other metrics like reclaims, scan rate, physical page faulting, and swapping to understand the true memory situation.

It is common to see spikes in the readouts. However, if those spikes are short-lived—less than 1 minute—you can disregard them. Most monitoring utilities provide short sampling periods (5 to 10 seconds) because the screen is more interesting to look at if it is actively moving. It is better to look at samples that span 1 to 5 minutes to give yourself a more accurate picture of performance.

On UNIX systems you can also look at the process size from the process status (`ps`) command. However, this does not yield accurate results because the size of the process will include both local and shared memory. This will cause you to believe that you have allocated more memory for these processes than is actually in use. Therefore, the number of physical pages per second is a much better indicator of your memory usage status.

Managing CPU activity

All resources affect CPU activity. Slow disks increase CPU activity by increasing the waits on I/O. If there is significant context switching, system delay time will increases.

Understanding CPU activity

To understand the meaning of a busy CPU, you have to understand the components of CPU activity. CPU activity is broken into four categories:

- **User time** — The amount of time spent performing user tasks. This is the main component that you paid for when you bought the CPU capacity for the system.
- **System time** — The amount of time devoted to system overhead like paging, context switches, scheduling, and various other system tasks. You can never completely eliminate this overhead, but you can keep it to a minimum. Refer to Chapter 3, "Performing System Administration," for details.
- **Wait on I/O time** — The amount of time the CPU is waiting for another resource (such as disk I/O).
- **Idle time** — The amount of unallocated time for the CPU. If there are no jobs in the process queue and the CPU is not waiting on a response from some other resource, then the time is logged as idle. On some systems, wait on I/O is logged as idle. This is because the CPU is idle and waiting for a response. However, this time does not accurately reflect the state of performance on the system.

On Windows, CPU activity is broken into the following categories:

- **User time** — The amount of time spent doing user tasks.
- **Privileged time** —The amount of time devoted to system overhead like paging, context switches, scheduling, and various other system tasks.
- **Idle time** — The amount of unallocated time for the CPU. Note that Windows does not track wait on I/O time. On Windows, wait time is logged as idle time.
- **Processor time** — An accumulation of other processor metrics. You can think of 100 – processor time = idle time.

Tuning your system

To the inexperienced administrator, time spent waiting on I/O seems highly undesirable. If I/O wait is having a noticeable impact on performance, it is clearly an issue. However, not all wait on I/O is cause for concern. Time is reported to this statistic whenever the CPU has completed a task and is waiting on some other system resource, such as disk or memory. Given that CPUs are much faster than other resources, the CPU needs to wait on these slower resources some percentage of the time.

In tuning a system, it is desirable to have some idle time available, but this is not a necessity. It is possible to use 100 percent of the CPU time on a two-CPU system with two processes. This would not mean that the system was performing poorly; this means the system is running as fast as it can run. In performance tuning, you are trying to push the bottleneck to the fastest resource (CPU). If your processors were 100 percent busy on user time, you would have the system tuned perfectly. At this point, the only way to improve performance would be to purchase faster or more CPUs. Similarly, if wait on I/O time increases, this does not necessarily mean that you have an I/O bottleneck. You need to look at the numbers as ratios of each other. Given ideal circumstances, it is best to have 70 percent user time, 20 percent system time, 0 percent wait on I/O, and 10 percent idle time.

The ratio of user time to system time should be approximately 3 to 1. This ratio varies widely when user time is below 20 percent of total CPU usage, as system time tends to consume a much larger portion. In some cases, system time is greater than user time due to poor allocation of resources. However, as you increase user time through performance tuning, system time should level off at around one-third or less of user time. This can be determined by looking at your CPU resources from any monitoring tool of your choice.

Figure 1–5 shows an example of the information displayed in the Performance Monitor.

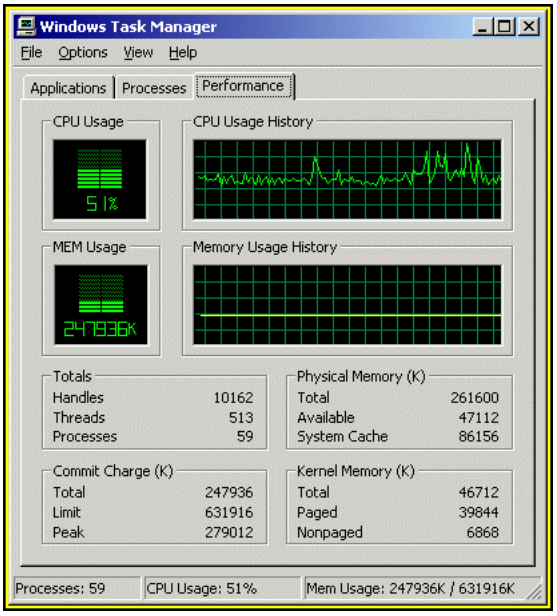


Figure 1–5: Windows XP Performance Monitor

Understanding idle time

Idle time can be positive because it means that the CPUs have capacity to support growth. It is not necessary for idle time to always be greater than zero. If idle time is zero for prolonged periods, and there is no significant amount of time logged to wait, you need to look deeper into CPU activity to determine the correct course of action.

For example, look at the CPU queue depths. *CPU queue depth* is the number of processes waiting to use the CPU. If there are always several processes in the queue, you need to either:

- Increase CPU capacity or increase CPU efficiency. (See the "Optimizing CPU usage" section on page 2–41.)
- Reduce demand by fixing code or moving processing to slow periods of the day.

If wait on I/O is high and there is no idle time, you need to increase disk efficiency (see Chapter 2, "Managing OpenEdge Database Resources"), reduce I/O throughput, eliminate disk variance, or modify your processing schedule. If wait on I/O is 10 percent or less and you still have idle time, you do not need to urgently work on the problem. You might want to look at those items outlined in the previous paragraph, but there might be nothing you can do to resolve the issue.

Monitoring your system

Monitoring your system throughout the day can help you to determine how to increase efficiency. Your system might look fine throughout the business day while you are there to see what is happening, but at night there might be significant *bottlenecks*. There are some applications that do over 70 percent of the total processing in the evening hours. Most environments have significantly different application makeup in the evening than in the day. A typical system might do On-Line Transaction Processing (OLTP) from 9 to 5, some end of day processing from 7 to 12, and decision support after midnight.

There might be things you can modify to take better advantage of system resources. There might be times of the day where more page writers are needed or you might need to change the times when things are run.

For example, consider when your organization schedules system backups. If the backup occurs at the same time you are doing decision support or running your end-of-day programs, you run the risk of exacerbating any disk bottleneck issues, which in turn might slow performance and increase waiting on I/O time on your CPU statistics. Sampling your data throughout the day brings visibility to problems. Tracking this information over time helps you plan new hardware purchases before the bottleneck becomes a significant problem.

Monitoring CPU performance in Fathom

CPU information is a good item to put on your **My Fathom** page so you have a visual clue to CPU performance on your system. When defining your private **My Fathom** page, you can select CPU on the **Other system resources to show** viewlet to have the viewlet shown in Figure 1–6 displayed on your **My Fathom** page.

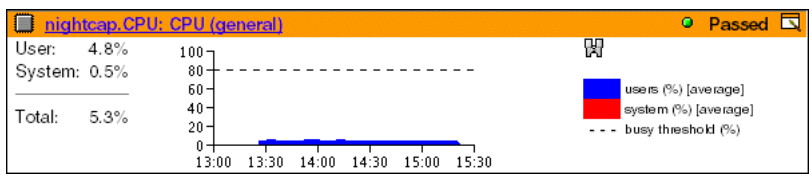


Figure 1–6: Monitoring CPU activity in Fathom

Fast CPUs versus many CPUs

In a best-case scenario, you would have a large number of fast CPUs in your system. However in most cases, you can only afford one or the other: multiple, slower CPUs, or a single, faster CPU. The benefit of fast CPUs is that they run single-threaded operations very quickly. An example of a single-threaded OpenEdge operation is an index rebuild. This process scans the entire database and rebuilds indexes. A machine with one fast CPU will run this process faster than a machine with multiple slower CPUs, given that all other resources are equal. The index rebuild process is only run on a periodic basis, so why should you consider this when making CPU choices? The primary reason is that during an index rebuild your users do not have access to the database, therefore you want this rebuild to complete as quickly as possible.

However, in most businesses, there are other examples of single-threaded operations that will be addressed by this architecture decision. End-of-day processing is a good example; you might need to apply all of your payments prior to being able to run your general ledger trial balance. On the other hand, multiple CPUs allow you to do two different operations simultaneously. One user can be entering orders while the other is shipping products. This has obvious benefits to the business in terms of efficiency.

So, how do you decide? The best way to decide is to look at your options and your application to determine the best solution. For example, an application that does a significant amount of single-threaded operations will benefit from a design that has fast CPUs, even at the expense of having fewer total CPUs in the system. An application that is mostly data entry with little or no single-threaded operations will benefit from a design that has more CPUs, even at the expense of each CPU being slower. This is another case where having a deep understanding of your application and workload will allow you to make intelligent decisions in the system area.

Summary

In this chapter we learned that to effectively manage your system resources, you need to:

- Analyze the requirements for your business.
- Configure the system from those business requirements.
- Measure your resources to ensure system availability.
- Trend resource usage over time to allow for advanced planning.

2

Managing OpenEdge Database Resources

Managing resources includes the process of moving potential processing conflicts, or *bottlenecks*, to the fastest resource. OpenEdge attempts to make this process transparent, but there are still some areas that require tuning.

This chapter discusses OpenEdge database internals and various ways to optimize OpenEdge-based resources to best take advantage of system resources, as outlined in the following sections:

OpenEdge database internals

OpenEdge database internals are presented in this section to help you to better understand how data is managed by the RDBMS. This topic is not directly related to system management or best practices. However, you can use the information in this section to help you design your system. Topics discussed include how the database is laid out on disk and database broker memory allocation.

Understanding database blocks

Many types of database blocks are stored inside the OpenEdge database manager, and most of the work to store these database blocks happens behind the scenes. However, it is helpful to know how blocks are stored so that you can create the best database layout.

The most common database blocks can be divided into three groups:

- Data blocks
- Index blocks
- Other block types

Data blocks

Data blocks are the most common blocks in the database. There are two types of data blocks: RM blocks and RM chain blocks. The only difference between the two is that RM blocks are considered full and RM chain blocks are not full. The internal structure of the blocks is the same. Both types of RM blocks are social. *Social blocks* contain records from different tables. In other words, RM blocks allow table information (records) from multiple tables to be stored in a single block. In contrast, index blocks only contain index data from one index in a single table.

The number of records that can be stored per block is tunable per storage area. See the "Optimizing data layout" section on page 2–20 for more information.

Each RM block contains four types of information:

- Block header
- Records
- Fields
- Free space

The block header contains the address of the block (dbkey), the block type, the chain type, a backup counter, the address of the next block, an update counter (used for schema changes), free space pointers, and record pointers. The block header is 16 bytes in length. Each record contains a fragment pointer, which is used by record pointers in individual fields, the Length of the Record field, and the Skip Table field (used to increase field search performance). Each record needs a minimum of 15 bytes for overhead storage and contains a Length field, a Miscellaneous Information field, and data.

Figure 2–1 shows the layout of an RM block.

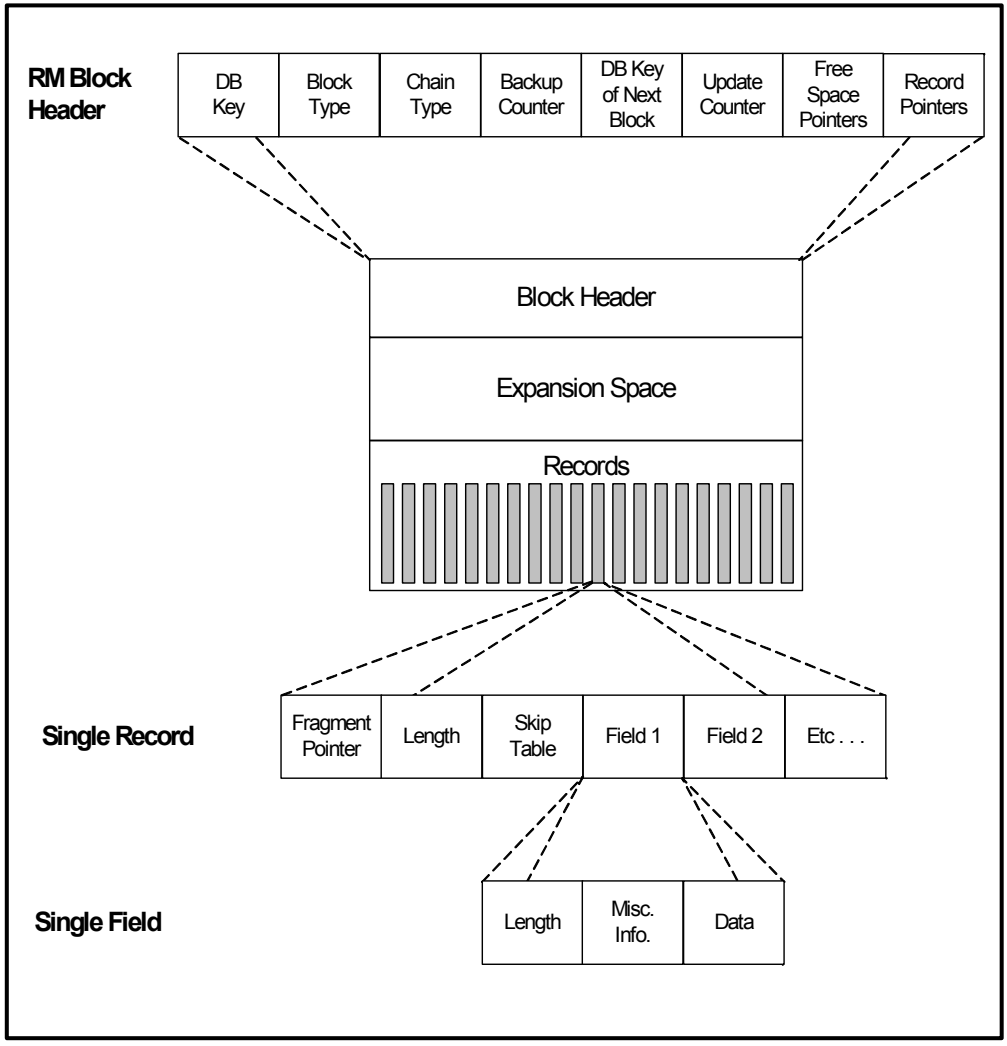


Figure 2–1: RM block

Index blocks

Index blocks have the same 16 bytes of header information as data blocks. Index blocks can store the amount of information that can fit within the block, and that information is compressed for efficiency. As stated earlier, index blocks can only contain information referring to a single index.

Indexes are used to find records in the database quickly. All indexes in the OpenEdge RDBMS are structured as B-trees, and are always in a compressed format. This improves performance by reducing key comparisons.

There are several key points to consider when discussing the structure of multiple indexes:

- A database can have up to 32,767 indexes.
- There is always one B-tree per index.
- Each B-tree starts at the root.
- The root is stored in an `_storageobject` record.

For the sake of efficiency, indexes are multi-threaded, allowing concurrent access to indexes. Rather than locking the whole B-tree, only those nodes that are required by a process are locked.

Other block types

There are several other types of blocks, but only a few that are valuable to understand. These include:

- Master block
- Storage object block
- Free blocks
- Empty blocks

Master block

The master block contains the same 16-byte header as other blocks, but this block is used to store status information about the entire database. It is always the first block in the database and is found in Area 6. This block contains the "tainted" flags for OpenEdge. *Tainted flags* tell OpenEdge there is a problem or abnormality with the database. You can retrieve additional information from this block using the Virtual System Table (VST) `_mstrblk` from the OpenEdge procedure editor:

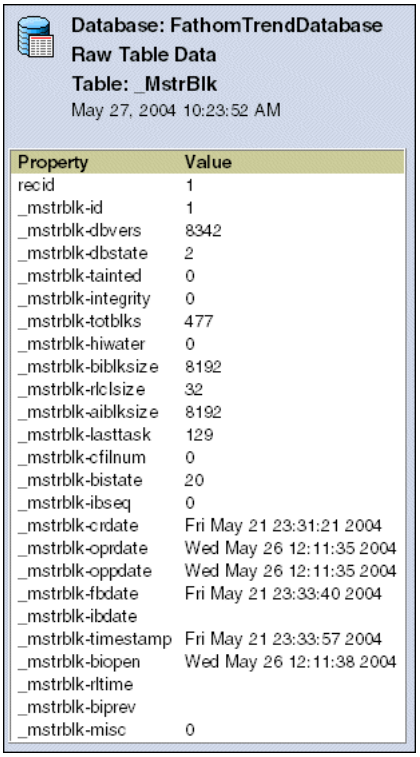
```
FIND _mstrblk.
DISPLAY _mstrblk WITH SIDE-LABELS.
```

You can also view this information in the Fathom Management console by viewing raw VST data.

To view raw VST data for a particular database in Fathom Management:

1. Select **Resources** from the menu bar.
2. Select the database that contains the data you want to view in the list frame.
3. Select **Raw VST Data** in the detail frame. A drop-down list of the available VST data for the selected database appears.
4. Select the VST and format you want to view.

Figure 2–2 shows an example of the HTML display of the `_MstrBlk` VST.



Database: FathomTrendDatabase
Raw Table Data
Table: _MstrBlk
May 27, 2004 10:23:52 AM

Property	Value
recid	1
_mstrblk-id	1
_mstrblk-dbvers	8342
_mstrblk-dbstate	2
_mstrblk-tainted	0
_mstrblk-integrity	0
_mstrblk-totblks	477
_mstrblk-hiwater	0
_mstrblk-biblksize	8192
_mstrblk-rlclsize	32
_mstrblk-aiblksize	8192
_mstrblk-lasttask	129
_mstrblk-cfilnum	0
_mstrblk-bistate	20
_mstrblk-ibseq	0
_mstrblk-crdate	Fri May 21 23:31:21 2004
_mstrblk-oprdate	Wed May 26 12:11:35 2004
_mstrblk-oppdate	Wed May 26 12:11:35 2004
_mstrblk-fbdate	Fri May 21 23:33:40 2004
_mstrblk-ibdate	
_mstrblk-timestamp	Fri May 21 23:33:57 2004
_mstrblk-biopen	Wed May 26 12:11:38 2004
_mstrblk-rltime	
_mstrblk-biprev	
_mstrblk-misc	0

Figure 2–2: Viewing raw VST data in Fathom Management

Storage object block

Storage object blocks contain the addresses of the first and last records in every table by each index. If a user runs a program that does a find first or find last, it is not necessary to traverse the index. The find command obtains the information from the storage object block and goes directly to the record. Storage object blocks are frequently used, so these blocks are pinned in memory. This availability further increases the efficiency of the request.

Free blocks

Free blocks have a header, but no data is stored in the blocks. These blocks can become any other valid block type. These blocks are below the high-water mark. The *high-water mark* is a pointer to the last formatted block within the database storage area. Free blocks can be created by extending the high-water mark of the database, extending the database, or reformatting blocks during an index rebuild. If the user deletes many records, the RM blocks are put on the RM Chain. However, index blocks can be reclaimed only through an index rebuild or an index compress.

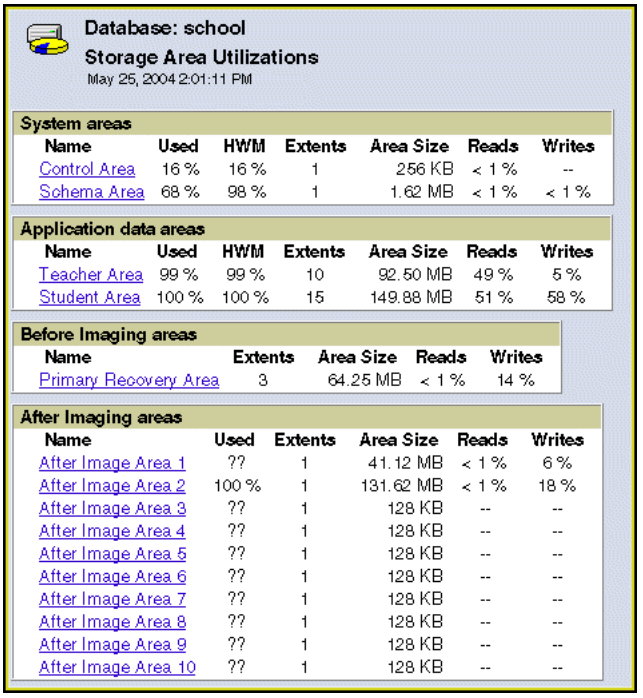
Empty blocks

Empty blocks do not contain header information. These blocks need to be formatted prior to use. These blocks are above the high-water mark but below the total number of blocks in the area. The *total blocks* are the total number of allocated blocks for the storage area.

To display a storage area utilization page in Fathom Management:

1. Select **Resources** from the menu bar of the Fathom Management Console.
2. From the list frame, browse to the applicable database and select it.
3. Select **Storage Areas** in the **Operation Views** section of the detail frame.

Figure 2–3 shows a database with no free space available. The percentage used and high-water mark are both at 99 to 100 percent, which means that the database is extending into the variable extent. If this database did not contain a variable-length extent, the database would crash if a user tried to extend the database.



Database: school

Storage Area Utilizations

May 25, 2004 2:01:11 PM

System areas

Name	Used	HWM	Extents	Area Size	Reads	Writes
Control Area	16 %	16 %	1	256 KB	< 1 %	--
Schema Area	68 %	98 %	1	1.62 MB	< 1 %	< 1 %

Application data areas

Name	Used	HWM	Extents	Area Size	Reads	Writes
Teacher Area	99 %	99 %	10	92.50 MB	49 %	5 %
Student Area	100 %	100 %	15	149.88 MB	51 %	58 %

Before Imaging areas

Name	Extents	Area Size	Reads	Writes
Primary Recovery Area	3	64.25 MB	< 1 %	14 %

After Imaging areas

Name	Used	Extents	Area Size	Reads	Writes
After Image Area 1	??	1	41.12 MB	< 1 %	6 %
After Image Area 2	100 %	1	131.62 MB	< 1 %	18 %
After Image Area 3	??	1	128 KB	--	--
After Image Area 4	??	1	128 KB	--	--
After Image Area 5	??	1	128 KB	--	--
After Image Area 6	??	1	128 KB	--	--
After Image Area 7	??	1	128 KB	--	--
After Image Area 8	??	1	128 KB	--	--
After Image Area 9	??	1	128 KB	--	--
After Image Area 10	??	1	128 KB	--	--

Figure 2–3: Displaying storage area utilization in Fathom Management

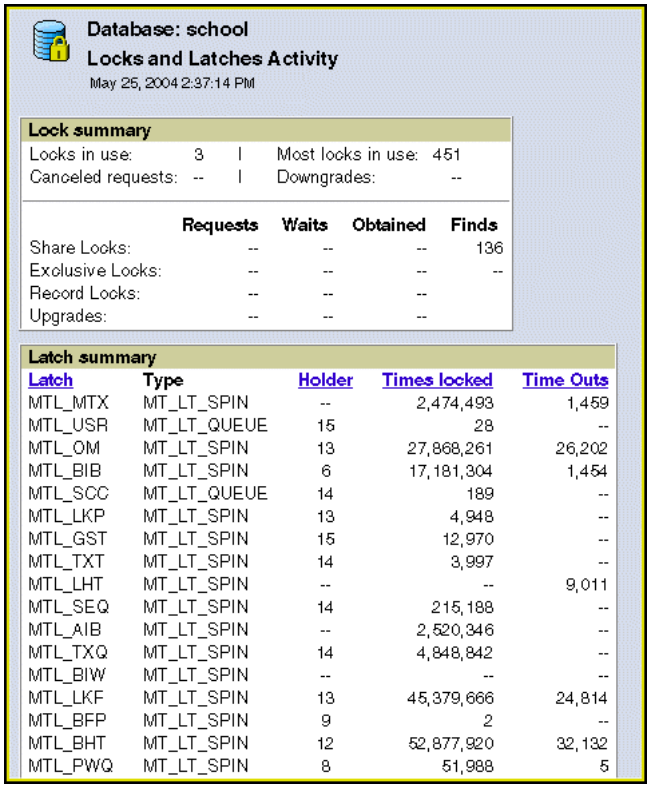
Understanding memory internals

The primary broker process allocates shared memory for users to access data within the database. The users also use structures within memory to allow for concurrent access to information without corrupting this information. For example, if two users are able to update the same portion of memory with different updates, then only one user's updates would be reflected in the database. This type of situation would lead to the incomplete updating of memory.

It is important to understand the concept of locking and how it applies to this situation. A *locked record* allows an update to complete without interference from other users. A *latch* is a lock in shared memory that allows a user to make modification to a memory block without being affected by other users. This latching approach has been evolutionary since its inception in Progress Version 6.3. The original latches were crude and heavy handed. A latch was taken for the entire shared memory pool making other users wait to make their modifications. As of Progress Version 8, there are multiple latches per resource.

Viewing locks and latches activity in Fathom Management

As shown in Figure 2–4, you can view these latches using Fathom.



Database: school
Locks and Latches Activity
May 25, 2004 2:37:14 PM

Lock summary

Locks in use:	3	Most locks in use:	451
Canceled requests:	--	Downgrades:	--

	Requests	Waits	Obtained	Finds
Share Locks:		--	--	--	136
Exclusive Locks:		--	--	--	--
Record Locks:		--	--	--	
Upgrades:		--	--	--	

Latch summary

Latch	Type	Holder	Times locked	Time Outs
MTL_MTX	MT_LT_SPIN	--	2,474,493	1,459
MTL_USR	MT_LT_QUEUE	15	28	--
MTL_OM	MT_LT_SPIN	13	27,868,261	26,202
MTL_BIB	MT_LT_SPIN	6	17,181,304	1,454
MTL_SCC	MT_LT_QUEUE	14	189	--
MTL_LKP	MT_LT_SPIN	13	4,948	--
MTL_GST	MT_LT_SPIN	15	12,970	--
MTL_TXT	MT_LT_SPIN	14	3,997	--
MTL_LHT	MT_LT_SPIN	--	--	9,011
MTL_SEQ	MT_LT_SPIN	14	215,188	--
MTL_AIB	MT_LT_SPIN	--	2,520,346	--
MTL_TXQ	MT_LT_SPIN	14	4,848,842	--
MTL_BIW	MT_LT_SPIN	--	--	--
MTL_LKF	MT_LT_SPIN	13	45,379,666	24,814
MTL_BFP	MT_LT_SPIN	9	2	--
MTL_BHT	MT_LT_SPIN	12	52,877,920	32,132
MTL_PWQ	MT_LT_SPIN	8	51,988	5

Figure 2–4: Viewing locks and latches activity in Fathom Management

To view locks and latches in Fathom Management:

1. Choose **Resources** from the menu bar of the Fathom Management console.
2. From list frame, browse to the desired database.
3. Select **Locks and Latches** in the **Operation Views** section from the database's detail frame.

Understanding shared memory resources

Figure 2–5 shows an example of shared memory resources. Note that it does not include all resources and is only used as an example. Also, this illustration is not to scale because database buffers account for more than 90 percent of shared memory, and the various latch control structures account for less than 1 percent.

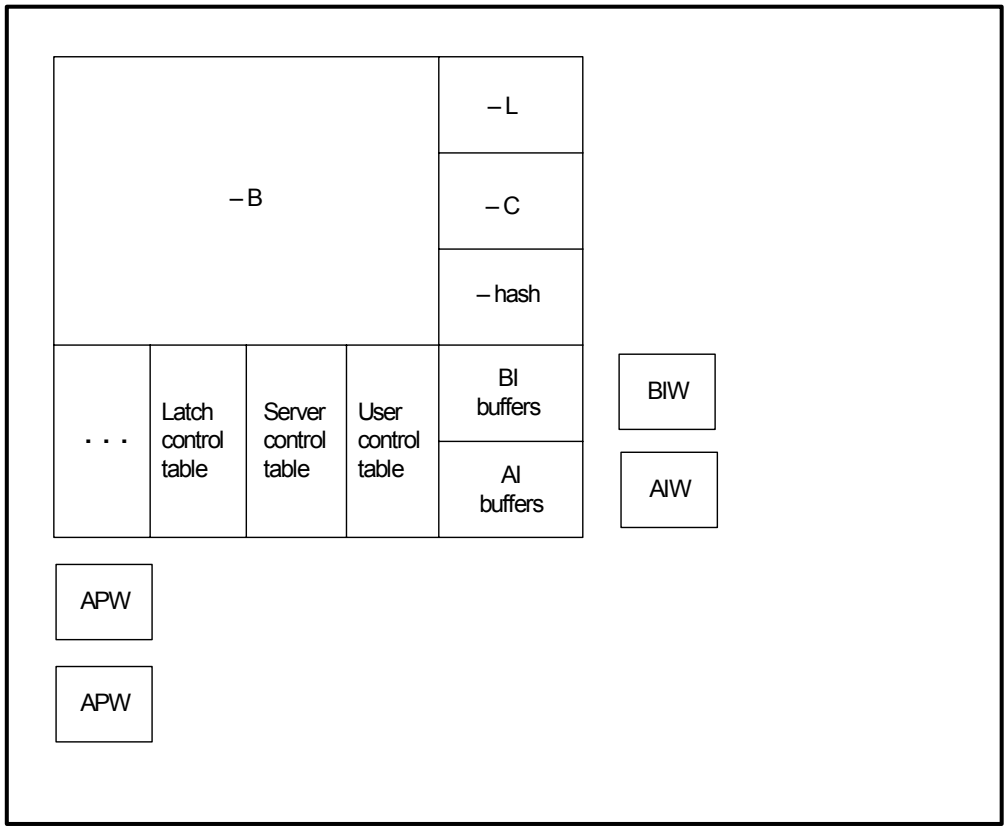


Figure 2–5: Shared memory resources

The database buffers are vitally important. They provide a caching area for frequently accessed portions of the database so that information can be accessed from disk once and from memory several times. Because memory is so much faster than disk, this provides an excellent performance improvement to the user if tuned properly. The concept of database buffer tuning is explored further in the "Profiling your system performance" section on page 3–41.

As Figure 2–5 illustrates, there are many resources inside of shared memory. Local users (both end-user processes and batch processes) update these structures. If two users access this database simultaneously and both users want to make an update to the lock table (`-L`), the first user requests the resource by looking into the latch control table. If the resource is available, the user establishes a latch on the resource using an operating system call to ensure that no other process is doing the same operation. Once the latch is enabled, the user makes the modification to the resource and releases the latch. If other users request the same resource, they retry the operation until the resource latch is available. For more information on latching, see the "Optimizing CPU usage" section on page 2–41.

The other processes shown in Figure 2–5 are page writers. These Asynchronous Page Writer (APW) processes write modified database buffers to disk. You can have more than one APW per database. The other writers, After-image Writer (AIW) and Before-image Writer (BIW), write after-image and before-image buffers to disk. There can only be a single BIW and a single AIW per database.

Adding remote clients

Figure 2–6 illustrates how the process of adding remote clients adds a TCP/IP listen socket and server processes.

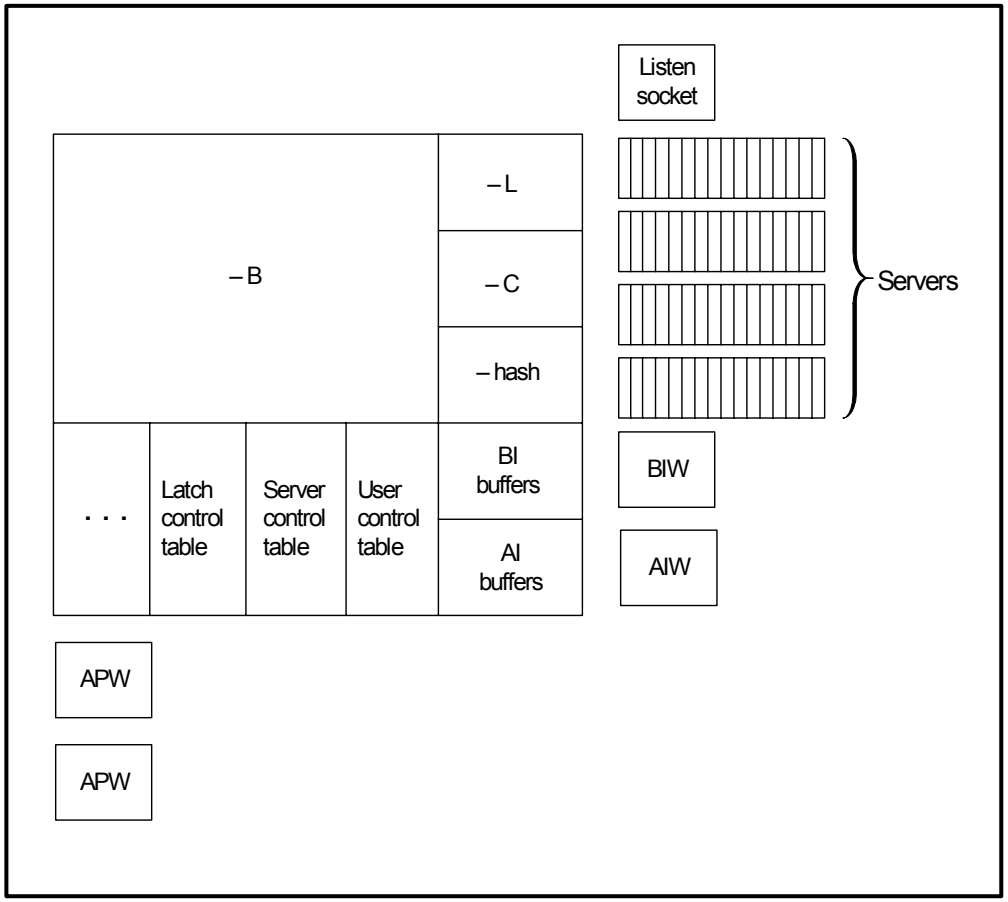


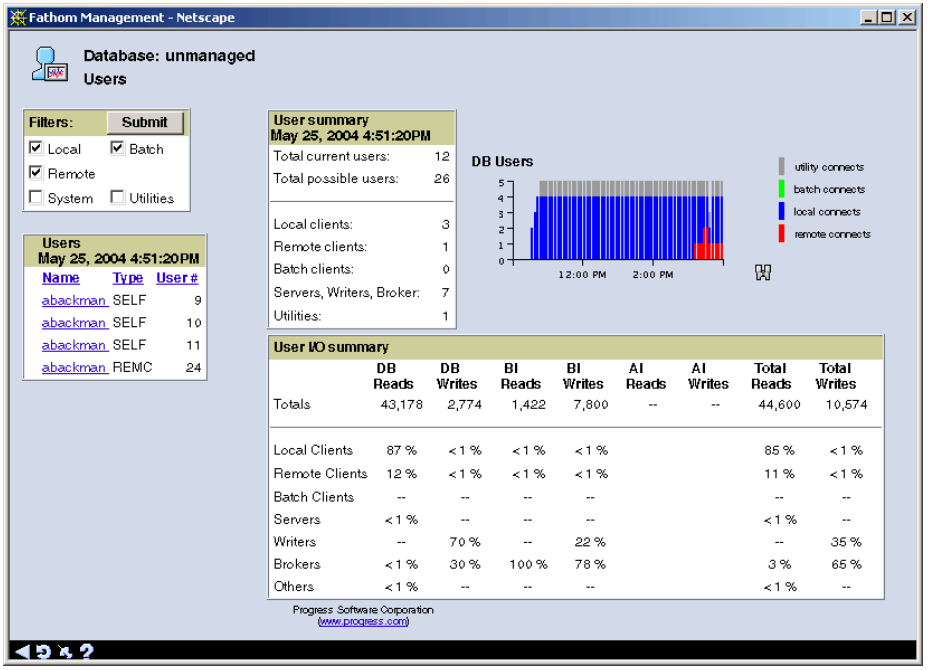
Figure 2–6: Shared memory resource—adding remote clients

The remote clients send a message to the listen socket, which in turn alerts the broker process. The broker process references the user control table and the server control table to determine if the user can log in and to which server the user can attach. If the servers are not active, a server is started, depending on the server parameters for this broker (parameters such as `-Mn`, `-Mi`, `-Ma`, and so forth). See *OpenEdge Data Management: Database Administration* for details. Once the proper server has been determined, a bidirectional link opens between that server and the remote client. This link remains open until the user disconnects or the broker is shut down.

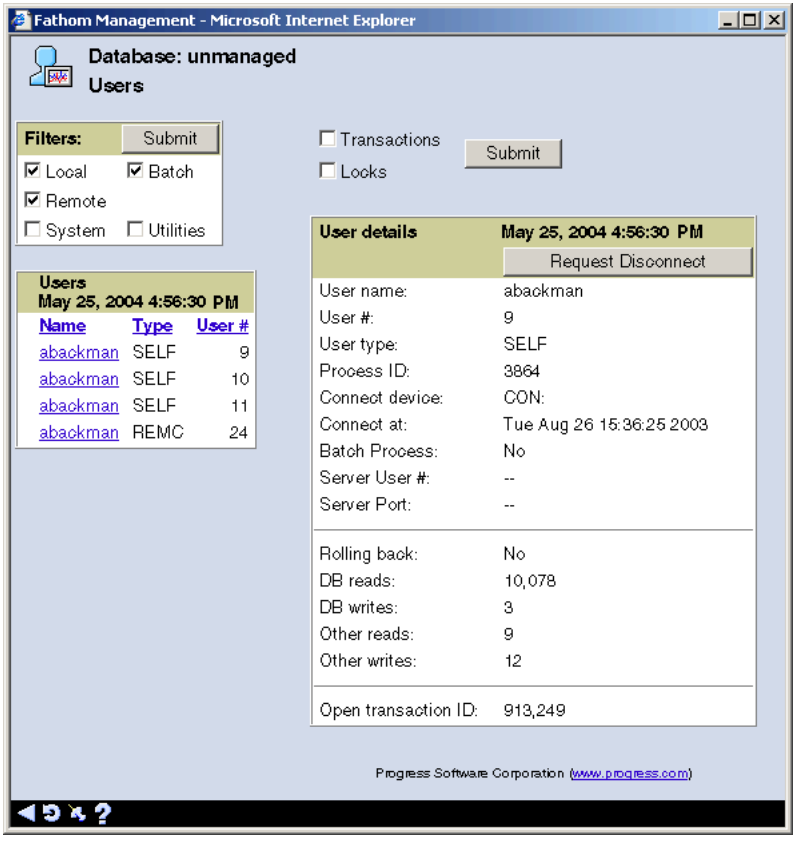
You can use Fathom Management to monitor the user-to-server relationship.

To monitor the user-to-server relationship:

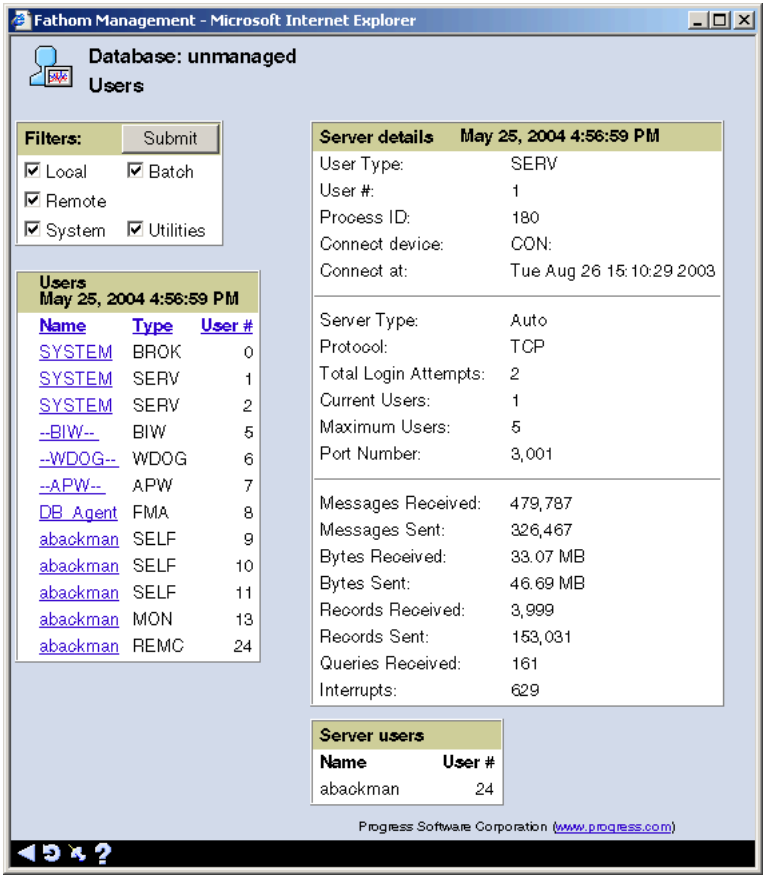
1. Select **Resources** from the menu bar.
2. From list frame, browse to the desired database and select it.
3. Select **User Activity** in the **Operation Views** section from the detail frame. The **Users** page appears:



4. Select the user you are concerned about to display information about the user and the server to which the user is attached:



5. Select the server to which the user is attached to determine if there are other users attached to the same server. Details for the selected server are displayed, as shown:



This detail is particularly helpful when you are seeing performance problems that are only impacting a subset of clients. If those clients are sharing a server with a report process or some other read-intensive operation, you can determine the problem and take corrective action.

Understanding how blocks are manipulated

There are some operations that take place behind-the-scenes for which you have limited control except during the setup of a database. These are internal database functions such as block manipulations. Although you do not have direct control over how these operations are accomplished, you can set up your database to take advantage of their behavior.

Record block manipulation

The process of adding a new record to the database is important because you want to balance the compaction of data while avoiding data fragmentation. High compaction rates allow you to read more records from disk in a single operation, and this process allows you to increase efficiency. If you have many fragmented records, efficiency will decrease because each fragmented record will require multiple block reads.

Adding new records to a database

Figure 2–7 illustrates that when records are added to the database for the first time, space is allocated for these records using a specific decision process.

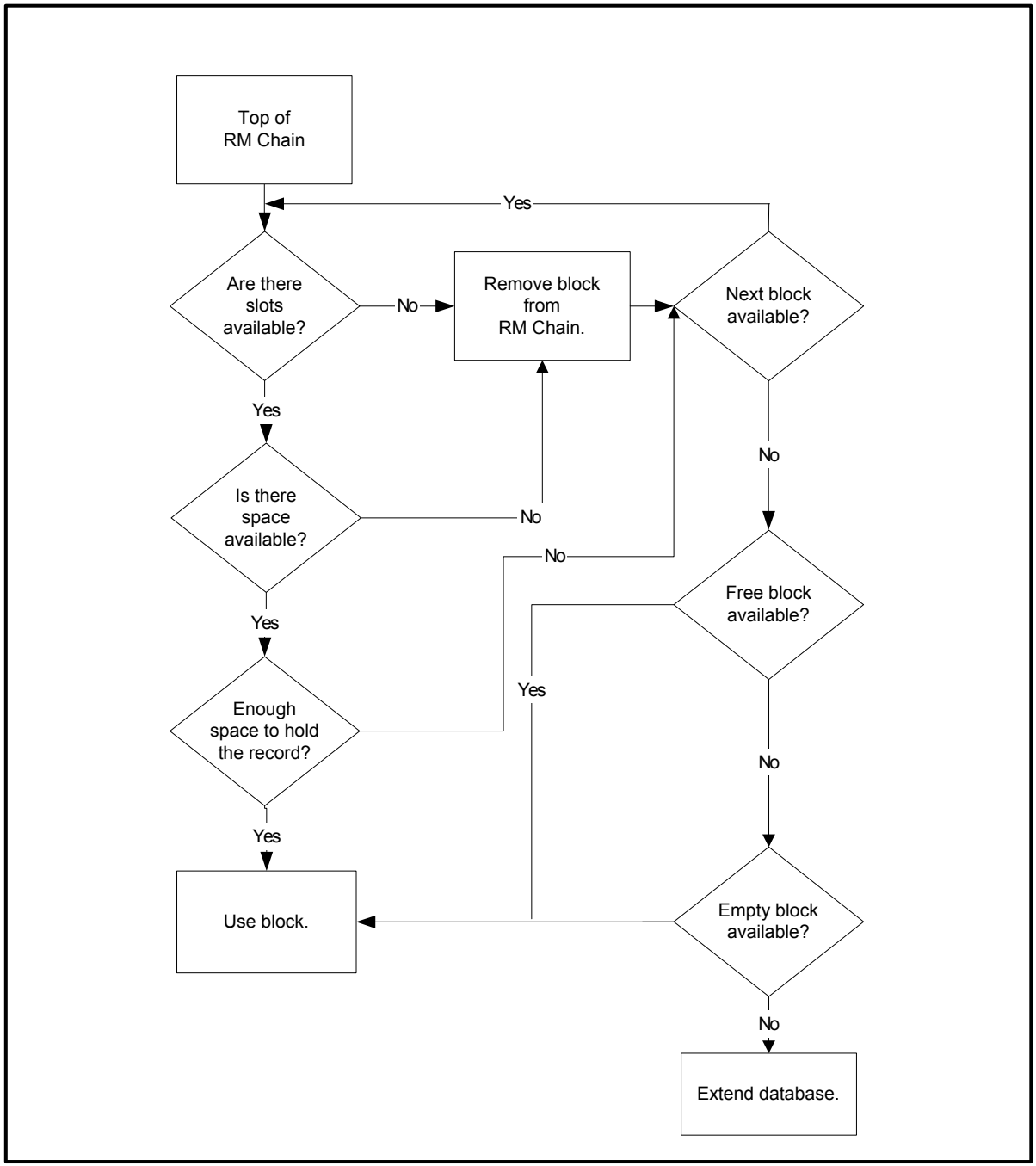


Figure 2–7: RM block allocation decision tree

To add new records to a database:

1. If the block is neither on an RM Chain block nor on the RM Chain, look at the next block. This can be repeated 20 times.

2. If the block from the RM Chain is full or contains no open slots to accept the record, remove the block from the RM chain. This can be repeated 100 times.

3. If the block from the RM Chain has available space but not enough to store the record, move the block to the end of the RM Chain and test the next block. This can be repeated three times.

4. If no RM chain blocks are available, allocate a free block.

5. If no free blocks are available, allocate several empty blocks and update the high-water mark.

6. If there are no empty blocks available, the last extent of the database area is variable length, and there is available space at the operating system level, then extend the database.

Updating existing records

The process to update an existing record is much simpler than adding new records. Keep in mind that you can cause record fragmentation by extensive record extension. There are documented cases of sequential searches of records taking three to four times longer due to record fragmentation. If some care is taken during setup and application design, you can minimize record fragmentation.

Consider the following when updating an existing record:

- If the new record is the same size as the original record, replace the original record.

- If the new record is smaller than the original record, replace original record and adjust the free space in the block.

- If the new record is larger than the original record and there is enough free space to store the new record, replace the original record and use the available free space.

- If the new record is larger than the original record and there is not enough free space to store the additional information in the existing block, the new record is divided among two or more blocks. This is called *fragmenting the record*.

Deleting records

Within the OpenEdge *database engine*, changes have been implemented to improve the efficiency of resource-intensive operations. Deleting a record is one of these types of operations. The key to efficiency is to postpone the deletion until the space is reused for storage, thus eliminating an unnecessary write.

Consider the following when deleting records:

- A placeholder record replaces the record to support transaction rollback.
- The placeholder record can be removed once the transaction is complete.
- The actual removal is done later for performance reasons.

Index block manipulation

Until this point, only record (RM) blocks have been discussed. We will now discuss index blocks.

Index blocks are easier to deal with at a high level and are not as structured as record blocks. Index blocks can store a variable number of entries per block, and this number does not need to be determined at database creation as do record blocks.

Index information is tightly packed. When information is put in the middle of an index, it is sometimes necessary to make additional space for the inserted index entry. This process is called an *index block split*. This can occur when, for example, you have an index block that contains entries 1, 2, 4, and 5 and contains no free space. If you insert entry 3 into this index, the information causes the block to split into two blocks, with half going into each block. In this example, the original block contains entries 1 and 2, while the new block (retrieved from the free chain, an empty block, or by extending the database) contains entries 4 and 5. Once the split is complete, entry 3 is added to the original block.

One issue you might encounter when indexes perform a significant number of block splits is that the index can become fragmented and need reorganization. Doing an index compress accomplishes this while the database is running. The user experiences only minimal impact. An offline index rebuild deletes all of the entries and inserts them back into the database to yield the most efficient index structure. However, this operation must be done when the database is offline.

The key thing to remember about blocks is to avoid fragmentation. This can be done through database setup and application design on the record block side and through the use of utilities on the index side.

The importance of understanding the internal structure cannot be overemphasized. By understanding the basics of OpenEdge database internals, you can make better decisions regarding the setup and tuning of your system.

Optimizing data layout

This section presents a general review of the elements that comprise a database to discuss the importance of database design. A database is made up of storage areas. Each storage area can contain one or more objects. A *database object* is a table or an index. There are other objects, such as sequences and schema. At this point, you have no control over the storage location of these objects.

Each storage area can contain one or more extents or volumes on disk. The *extents* are the physical files stored at the operating system level. Each extent is made up of blocks, and you can determine the block size for your database. The block sizes that you can choose from are: 1KB, 2KB, 4KB, and 8KB. You can only have one block size per database, but each area can have differing numbers of records per block.

Proper database design is essential because all data originates from a storage area. There are several items to consider when determining the layout of a database. The first is your mean record size. This is easy if you have an existing OpenEdge database. When you run a database analysis, this information is included in the output. The other information is generally not as easy to obtain for the database administrator because it requires knowledge of how the application is used. Even the designers of an application might not know how the users are taking advantage of the design. For example, it is important to know if a table is generally accessed sequentially or randomly. Is the table frequently used, or is it historical and thus used infrequently? Do the users access the table throughout the day or only for reporting? Are the individual records growing once inserted or are they mostly static in size? The answers to these questions will help determine the size and layout of a database and allow you to take best advantage of your disks.

Sizing your database areas

When trying to determine the size of an area, you need to look at the makeup of the information being stored in that area. As stated before, an area can contain one or more tables or indexes. The default area should generally be reserved for schema and sequence definitions, as this will make conversions easier in the future. The first step in this process, if you already have a OpenEdge database, is to do a table analysis. See *OpenEdge Data Management: Database Administration* for details.

The following shows a portion of sample output for a table analysis:

```
                ---Record---              ---Fragment---          ---Scatter---

Table         Records  Bytes      Min  Max     Mean   Count    Factor  Factor

Work Orders   12,383   6,109,746  60   10,518  493    21,131   1.6     4.0
```

After doing a table analysis, you need to focus on record count (**Records**) and mean record size (**Mean**). Look at every table and split them according to mean record size. In the vast majority of cases, use an 8KB-block size to better conform to the operating system; the major exception to this rule is on Windows where a 4KB-block size is more appropriate. Each record contains approximately 20 bytes of record overhead, so 20 is added to the mean record size of each record prior to doing any calculations. These 20 bytes of overhead take into account the record and the RM block header overhead, as outlined in the "OpenEdge database internals" section on page 2–2.

Block sizes

Why is an 8KB-block better on one system than another? The answer is how the operating system handles files and memory. On Windows, the operating system assumes that files and memory are handled in 4KB chunks. This means that all transfers from disk to memory are 4KB in size.

It is good practice to match or be a multiple of the operating system block size, if possible. This means that an 8KB block would work fine too, right? Well, not really. The Windows operating system has been highly optimized for 4KB and performs worse at an 8KB setting in the majority of cases. On UNIX operating systems the block size is generally 8KB or a multiple of 8K. The block size is tunable. Generally, an 8K-block size is best on UNIX systems. There are exceptions to every rule. The intention is to make a best estimate concerning that which will aid performance and assist OpenEdge in meshing with your operating system better. In most cases, it has been proven that 8KB works best, with the exception of Windows.

The only way to prove this is to benchmark performance on your system. You can obtain hard data about block sizes using settings with the best performance characteristics.

Method to determine the number of records per block

Use the following formula to determine the number of records per block:

1. Take the mean record size.
2. Add 20 to the record size.
3. Divide 8192 (8KB-block size) or 4096 (4KB-block size) by the number in step 2.

OpenEdge allows you to have anywhere from 1 to 256 records per block per area. The number of records per block must be a binary number (1, 2, 4, 8..., 256).

Most of the time, the record length will not divide into this number evenly so you need to make a best estimate. If your estimate includes too many records per block, you run the risk of fragmentation (records spanning multiple blocks). If your estimate includes too few records per block, you waste space in the blocks. The goal is to be as accurate as possible without making your database structure too complex.

Distributing tables across storage areas

Now that you know how many records can fit in each block optimally, you can review how to distribute the tables across storage areas. Some of the more common reasons to split information across areas include:

- Controlled distribution of I/O across areas.
- Application segmentation.
- Speed offline utilities.

The last reason—to speed offline utilities—is valid until all utilities are brought online. There is no reason to perform application segmentation now. However, in the future having your data segmented might allow you greater flexibility in maintenance. The cost to do this work now is fairly low if you are already splitting some data, or moving from a previous version of Progress to OpenEdge.

Another reason to break out a table to its own area concerns how the table is populated and accessed. In those cases where records are added to a table in primary index order, most of the accesses to these records are done in sequential order via the primary index. There might be a performance benefit in isolating the table. If this is a large table, the performance benefit gained through isolation can be significant.

There are two reasons for the performance improvement:

- The first reason is that one database read will extract multiple records from the database, and the other records that are retrieved are likely to be used. This approach improves your buffer hit percentage.
- The second reason is that many disk drive systems have a feature that reads ahead and places in memory items that it believes are likely to be read. Sequential reads take best advantage of this feature.

Finally, databases can contain different types of data in terms of performance requirements. Some data, such as inventory records, is accessed frequently, while other data, such as comments, is stored and only read as necessary. By using storage areas you can place frequently accessed data on a “fast disk.” However, this approach does require knowledge of the application. You can determine the activity per table using either Fathom or Virtual System Tables (VSTs). Fathom Management allows you to trend the table usage over time to better determine the true activity level of a table.

The overall goal in obtaining this information is to provide enough areas to achieve the following:

- Greater control of information.
- Speed for offline utilities.
- Maximum record efficiency without sacrificing ease of maintenance.

The calculations previously made are used to determine the number of records per block. In some cases you must make a borderline decision when choosing to store a table in an area. For example, what should you do if a table has records per block set to a lower number than the mean record size required to fill the block, or a greater number of records per block that might cause record fragmentation? The following example shows that the decision is fairly easy to make.

Example

Assume there is a table in a database with 1 million records and a mean record size of 41 bytes.

To distribute a table across storage areas:

1. Add the record overhead (approximately 20 bytes) to determine the number of the actual size of the stored record:

```
Mean record size (41) + record overhead (20) = actual storage size (61)
```

2. Divide that number into your database block size to determine the optimal records per block:

```
Database block size (8192) / actual storage size (61) = optimal records
per block (134)
```

Here is your decision point. You must choose a power of 2 from 1 to 256 for the records per block. This leaves you with two choices: 128 and 256. If you choose 128, you will run out of record slots before you run out of space in the block. If you choose 256, you run the risk of record fragmentation. Make your choice according to the nature of the records. If the records grow dynamically, then you should choose the lower number (128) to avoid fragmentation. If the records are inserted and are static in size, you should choose the higher number (256) of records per block because generally OpenEdge will not fragment a record on insert. Most fragmentation happens on update; be careful with records that are updated frequently and are likely to increase in size.

Also, if you choose the lower value, you can determine this cost in terms of disk space. To do this, take the number of records in the table and divide by the number of records per block to determine the number of blocks that will be allocated for record storage:

```
Number of records (1,000,000) / records per block (128) = allocated blocks
(7813)
```

Next, calculate the number of bytes wasted per block by multiplying the actual storage size of the record by the number of records per block and subtracting this number from the database block size:

```
Database block size (8192) – (Actual storage size (61) * records per block
(128) = Wasted space per block (384)
```

Take the number of allocated blocks and multiply them by the wasted space per block to determine the total wasted space:

```
Allocated blocks (7813) * wasted space per block (384) = total wasted space
(3000192)
```

In this case, the total wasted space that would result in choosing the lower records per blocks is less than 3MB. In terms of disk space, the cost is fairly low to virtually eliminate fragmentation. However, you should still choose the higher number for static records, as you will be able to fully populate your blocks and get more records per read into the buffer pool.

Considering wasted slots

Another issue to consider when choosing to use 256 records per block is that you will be wasting slots in the block. Since the number of records determines the number of blocks for an area, it might be important to have all of the entries used to obtain the maximum number of records for the table.

The following table shows the maximum number of blocks for an area by the records-per-block setting:

Records per block	Maximum number of blocks for area
4	536,870,911
64	33,554,432
256	8,338,607

In the case of small tables, put the indexes with their associated tables for ease of maintenance. For large tables (tables with many rows) isolating an index or a subset of indexes to their own location will generally improve performance.

Determining space to allocate per area

You must determine the quantity of space to allocate per area. OpenEdge is fairly good about keeping data and index storage at good compaction levels. Most data areas are kept from 90 to 95 percent full and indexes are generally maintained at 95 percent efficiency in the best case. It is generally advisable to use an 85 percent ratio. This is a reasonable ratio. Using the 1-million-record example previously discussed, you can see that the records plus overhead would take 61 million bytes of storage:

```
(41 bytes (Mean record size) + 20 bytes (overhead)) * 1,000,000 (Number of
records) = 61 million bytes
```

This is only actual record storage. Now, take this value and divide it by the expected fill ratio. The lower the ratio, the more conservative the estimate:

```
61,000,000 /.85 = 71,764,706 bytes (total storage needed)
```

To determine the size in blocks, divide this number by 1KB (1024 bytes). This step is necessary because the amount of space needed will be expressed in the structure description file (`dbname.st`) in kilobytes regardless of the block size of the database:

```
71,764,706 / 1024 = 70083 (1KB blocks)
```

If there are other objects to be stored with this table in a storage area, you should do the same calculations for each object to determine the total amount of storage necessary. If this is the only object to store in this area, consider future growth requirements.

Using extents

Most users prefer binary numbers for their extent sizes because these numbers are easier to monitor from the operating system level and see if problems occur, such as growing into a variable extent. In this case, you can choose one 102,400KB extent to store the data, with room for expansion, and one variable extent. Each area should have a variable extent as the last area to allow for growth. Monitoring and trending should keep you from needing this last extent, but it is preferable to have it available if you need it.

Extents allow you to distribute your data over multiple physical volumes if you do not have striping on your system. For example, you could split the 70MB of data in the previous example across several physical volumes by reducing the size of each volume. You could have eight fixed 10MB extents and one variable extent and "stripe" your information across three drives.

As shown in Figure 2–8, you could put the first, forth, and seventh extents on the first drive, the second, fifth, and eighth extents on the second drive, and the third, sixth, and variable extents on the third drive. OpenEdge fills these extents in order, so the first 10MB of data goes into the first extent, the second 10MB of data goes into the second extent, and so on. By striping the extents, you will have a mix of old and new data. While this is probably not as good as a hardware stripe of 128KB stripes, it does help you eliminate variance on your drives.

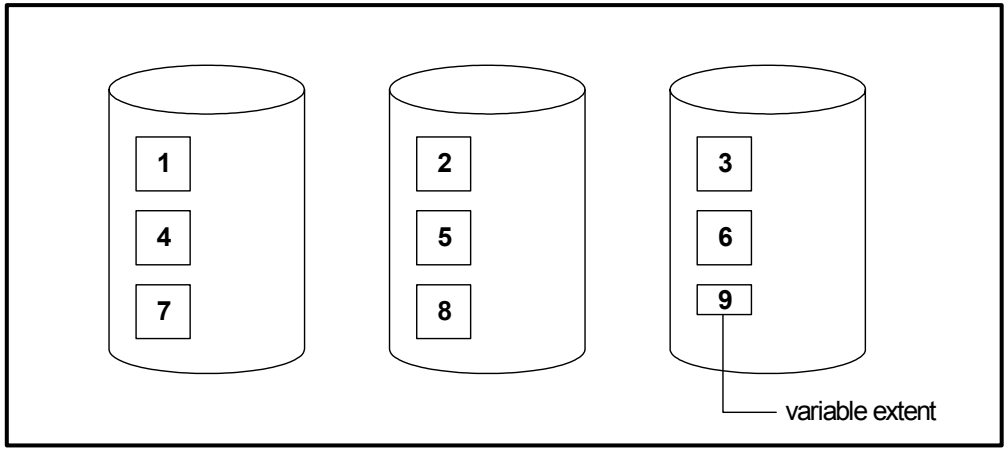


Figure 2–8: Extents

Even if you do have striping, you might want to have multiple extents. In Version 9.1C, Progress introduced large file support to Progress databases. This allows the user to allocate extents up to 16GB in size. To turn on this feature, you need to run `proutil` on your database with the enable large files parameter. (See *OpenEdge Data Management: Database Administration* for details.) The old and default limit is 2GB per extent. If you want to store more than this amount, you need to have multiple extents per area.

Another reason to have extents is indirection. *Indirection* occurs when a single inode table cannot address all of the physical addresses in a file. An *inode table* is a table of contents on a disk that is used to translate logical addresses to physical addresses. If a second inode table is needed, you will do an extra I/O operation for every physical request into the database. This activity is not good for performance. Most operating systems claim to be able to directly address a 4GB file under best-case scenarios. Through testing, the real number varies across operating systems, but 1GB seems to be a safe number across all operating systems with modern file systems. On Windows NT, you want to use NTFS file systems for best performance.

Index storage

This chapter has only discussed record storage. The reason for this is that record storage is fairly easy to calculate while index storage is not. Index compression makes calculation difficult. The fact that the compression algorithm has been modified over the years makes the calculation even harder. In an effort to make things easier, you can look at a database analysis and use the information from that activity to make your decisions. Again, remember to add room for growth and general overhead, just like with data storage.

If you have an existing database, you can take statistics to determine index storage size. Without a database, you have to estimate the size. The number and nature of indexes can vary greatly between applications. Word indexes and indexes on character fields tend to use more space, while numeric indexes are significantly more efficient in terms of storage. There are databases where indexes use more storage than data, but these are the exception and not the rule.

In general, indexes account for approximately 30 percent of total storage. Therefore, you can take 50 percent of your data storage as an estimate of index storage. Remember that this percent might vary greatly, depending on your schema definition. Consider this estimate as a starting point and adjust and monitor accordingly.

The following example highlights a portion of a database analysis report that shows the proportion of data storage to index storage within an existing database. Use this information to determine the allocation of disk resources to the areas that are going to contain the data:

```
SUMMARY FOR AREA "Student Area": 8
----------------------------------
                    Records               Indexes                 Combined
Name              Size   Tot percent           Size   Tot percent           S
ize   Tot percent
PUB.stuact        18.9M  12.6             9.7M   6.4             28.6M   19.0
PUB.student       30.3M  20.1            20.1M  13.4             50.5M   33.5
                  ------------------------------------------------------------
Total            115.3M  76.4            35.6M  23.6            150.8M  100.0
```

Primary recovery area

The size of the primary recovery area, also known as the before-image file, is another area for concern. This area is responsible for the recoverability of your database on an ongoing basis. This area is very important to the system. The primary recovery area is written to frequently, and if it is on a slow disk your update performance will be affected. The size of this area varies depending on the length of transactions and the activity on your system.

The primary recovery area is made up of clusters, which are tunable in size. When records are modified, notes are written to this area. If a problem occurs or if the user decides to "undo" the changes, this area can be used to ensure that no partial updates occur.

For example, assume you want to modify all of the records in a table to increase a value by 10 percent. You would want this to happen in an all-or-nothing fashion because you could not determine which records were modified if the process terminated abnormally. In this case, you would have one large transaction that would modify all of the records. If a problem occurs during the transaction, all of the modifications would be rolled back to the original values. Why is this important? If you have several of these processes running simultaneously, the primary recovery area could grow quite large.

Let us review the structure of this area and how it is used and reused. The structure of the area is a linked list of clusters. The cluster size can be modified from small (8KB) to large (greater than 256 MB), as shown in Figure 2–9.

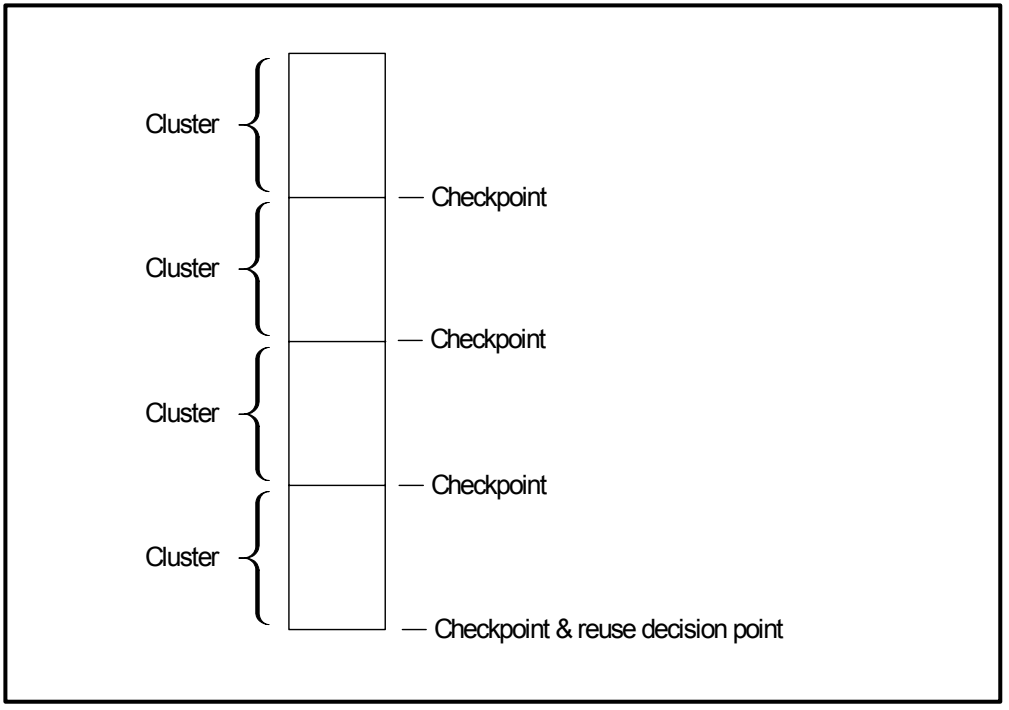


Figure 2–9: Primary recovery area

The smaller the cluster size the more frequent the checkpoints occur. A *checkpoint* is a synchronization point between memory and disk. While there is a potential performance benefit from infrequent checkpoints, this must be tempered with the amount of time it takes to recover the database.

The best way to determine the before-image cluster size is to:

- Monitor the database at the time of the day when you make the most updates to the database.
- Trend the data from the database with Fathom Management.
- Review the duration of your checkpoints throughout the week.

Ideally, checkpoints should happen no more than once every two minutes. If you are checkpointing more often than necessary, you should increase your before-image cluster size. This does not mean you should decrease the cluster size if it is happening less frequently. The default of 512KB is fine for smaller systems with low update volume, while a value of 1024KB to 4096KB is best for most other systems.

To view a checkpoint summary in Fathom Management:

1. Select **Resources** from the menu bar, and from the list frame browse to the database.

2. Select the **Page Writers** option from the **Operations Views** section of the database's **Resource** page.

Figure 2–10 shows an example **Checkpoint summary** section.

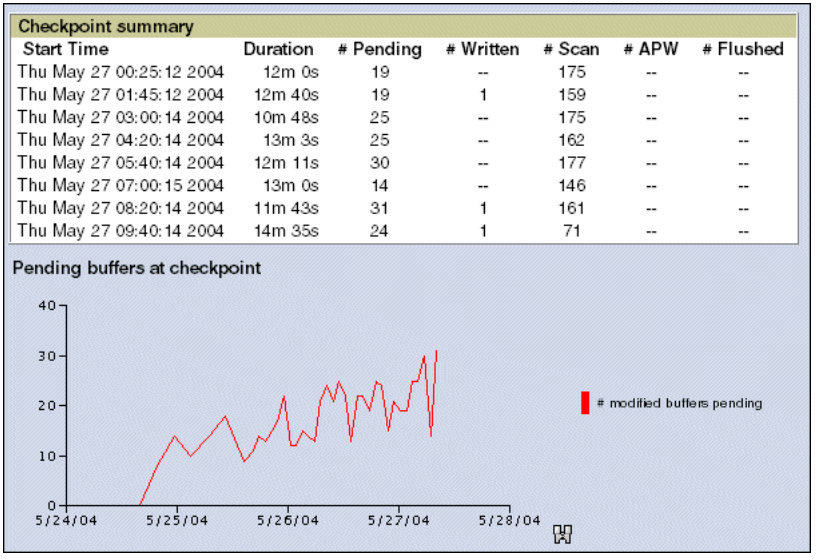


Figure 2–10: Checkpoint summary

As previously stated, the cluster size influences the frequency of the checkpoints for the database. As users fill up a cluster with notes, they are also modifying shared memory. The page writers (APWs) are constantly scanning memory, looking for modified buffers to write to disk. At the first checkpoint, all of the modified buffers are put in a queue to be written prior to the next checkpoint. The buffers on the modified buffer queue are written by the page writers at a higher priority than other buffers. If all of the buffers on the queue are written prior to the next checkpoint, it is time to schedule the current modified buffers. This is the goal. If all of the buffers are not written, then you must write all of the previously scheduled buffers first and then schedule the currently modified buffers. If you are checkpointing at the proper frequency and you are still flushing buffers at checkpoint, you should add one more APW and monitor further. If adding the APW helps, but does not eliminate the problem, add one more. If adding the APW does not help, look for a bottleneck on the disks.

The format of the primary recovery area has been discussed, but not its size. There is no formula for determining the proper size because the size of the area is so dependent on the application. The recommendation is to isolate this area from other portions of the database for performance reasons. If you only have one database, you can isolate this area to a single disk (mirrored pair), as the writes to this area are sequential and would benefit from being placed on a single nonstriped disk. If you have several databases, you might want to store your primary recovery areas on a stripe set (RAID 10) to increase throughput.

Optimizing database areas

This section details database area optimization. Although some of the information presented in this section might be found in other sections of this book or other manuals, it is repeated here to present the most common solutions to area optimization in one place for easy reference. The goal of area optimization is to take advantage of the OpenEdge architecture and the operating system.

Data area optimization

Following a few simple rules can make your data areas easy to maintain and provide optimal performance for users.

Splitting off the schema

By splitting off the schema and sequences from other portions of your database, you can make future upgrades to OpenEdge and your application more transparent to the user. One of the operations that takes place during most major version changes of OpenEdge is a metaschema change. The *metaschema* is a set of definitions for the internal structure of data. Your schema definitions create the rules by which data can be added or modified within your application. Metaschema serves the same purpose for the definition of your schema. By eliminating information other than your schema definitions and sequences from the metaschema area, the task of updating this information is reduced.

Choosing an appropriate block size

Matching the database block size to the operating system allows for a more efficient transfer of data. If you have a block size that is too small, the operating system retrieves more blocks than your request, and that transfer of additional information might or might not be useful. If the additional information is not used by the application, then the transfer was wasted. Larger blocks are generally better because of the way indexes behave. If each block contains more information, then you will require fewer index blocks and fewer index levels to contain the data. Index levels in a B-tree are important. If you can eliminate a level from an index, you can save yourself one additional I/O operation per record request.

Keeping extents small to eliminate I/O indirection

I/O indirection happens when a single inode table cannot address all of the addresses within a file. The *inode* is a mapping of logical to physical addresses in a file. Think of the inode table as a table of contents for the file. In theory, a file can be large (more than 4GB) before indirection occurs, but the conditions need to be perfect.

In the real world, it is possible to see indirection at a file size of 500MB, but on most systems you do not see indirection until 1GB or higher. The penalty for smaller database area extents is fairly low. Generally, you only need to increase the number of file descriptors (open files) at the operating system.

Keeping areas small for offline utilities

Currently utilities for OpenEdge are available online; there are still some utilities that require you to shut down the database prior to running them. For these utilities, limit the amount of information per area to reduce the amount of downtime needed for the utility to run.

The best example of this is an index rebuild. If you do an index compact and an index fix, you can achieve nearly the same thing. However, you might still want to do an index rebuild due to index corruption or some other reason. If you only need to rebuild one index, you scan the entire area where the records for that index are stored to ensure you have a pointer to every record. If all of your records are in one area, this can take a significant amount of time. It is much faster to scan a portion (only those records in the same area) of your records than the entire database.

Always have an overflow extent for each area

The last extent of every area, including the primary recovery area, but not the after-image areas, should be variable length. Though monitoring, trending storage capacity, and growth should eliminate the need to use the variable extent, it is preferable to have it there if you are going to fill all of your fixed extents and cannot shut down to grow the database. The variable extent allows the database to grow as needed until it is possible to extend the database.

Enabling large files

You should always have large files enabled for your database. However, just as you never want to grow into the variable extend of your data areas, you want to avoid using the large files feature. This feature should be viewed as a safety valve for unanticipated growth.

Enabling large files allows you to support extent sizes up to 1TeraByte (TB), which is 1000GB, provided that the operating system supports large files. On UNIX, you need to enable each file system where the database resides for this feature to work. On Windows a file can fill the entire volume. By default, large files are disabled; this was done for compatibility reasons.

To enable large file support for a database, execute the following command:

```
proutil dbname -C enablelargefiles
```

Partitioning data

Partitioning data by functional areas is a reasonable way to split your information into small pieces to reduce the size of a given area. This activity allows you to track the expansion of each portion of the application independent of other portions. At some future point, it might be possible to manipulate areas independently of each other. When and if this time comes, you will be well positioned if you partition your tables when you are doing the initial split of your data.

Another benefit of partitioning your data deals with a corrupted database. For example, you can more easily identify data in a corrupted area of your database.

Primary recovery (before-image) information

On systems where updates are done frequently, it is important to make the read and write access to this database area as efficient as possible. The write access is more important.

The following sections provide simple tips to create an efficient environment for the primary recovery area.

Extent size rules

These rules apply to both the primary recovery area and the data areas:

- Do not make them too large.
- Always make the last extent variable length.
- Enable large files as a safety valve.

Enabling large files is particularly important on the primary recovery area because this is the place you are most likely to experience issues. A large update program with poor transaction scoping or a transaction held open by the application for a long period of time can cause abnormal growth of this area. If the fixed portion of an area is 2GB in size, you start extending your variable portion in that same transaction. Only then do you notice that you might need more than 2GB of recovery area to undo the transaction. If you are large-file enabled, and have enough disk space, there is no problem. If you are not large-file enabled the database might crash and not be recoverable because there is no way to extend the amount of space for the recovery area without going through a proper shutdown of the database.

Sequential access

The primary recovery area is sequentially accessed. Items are written to and read from this area in a generally linear fashion. If you are able to isolate a database's primary recovery area from other database files and other databases, then it is a good idea to store the extents for this area on a single disk (mirror).

While striping increases the throughput potential of a file system, it is particularly effective for random I/O. The striping of the primary recovery area can cause additional disk head movement if this area is isolated. If the area is not isolated from the database or you are storing the primary recovery areas of several databases on the same disk, it makes more sense to use striped disks because the I/O will be fairly randomized across the databases.

BI grow option

The BI grow qualifier of `proutil` allows you to preformat `BI` clusters before the user enters the database. Preformatting allows you to write more recovery notes and to fill more clusters before the database needs to make a reuse decision. If the reuse decision is made less often, then it is more likely that the oldest cluster is ready for reuse and is reused. If the primary recovery area does not experience any abnormal growth, you can keep a contiguous format to this area, which is good for performance. Your database must be down for you to grow the `BI` file (primary recovery area).

The command to execute is:

```
proutil dbname -C bigrow #
```

where `#` is the number of clusters you want to add to your primary recovery area. These are additional clusters to the number of formatted clusters that you already have in the primary recovery area. Generally, this command is run after a truncation of the `bi` file so there are zero allocated clusters at that time.

After-image information

To this point in this chapter, only the default portions of the database have been considered. The after-image file is used to enable recovery to the last transaction or to a point in time in the case of media loss. There are several other reasons to implement after-imaging, but its role in a comprehensive recovery strategy is the largest.

The after-image file is like the before-image file in the sequential nature of its access. It does not have automatic reuse like the before-image file because it requires intervention from the administrator to reuse space. After-imaging is the only way to recover a database to the present time in the case of a media failure (disk crash). It also provides protection from logical corruption by its “point-in-time” recovery ability.

For example, assume a program accidentally runs and incorrectly updates every customer name to “Frank Smith.” If you have mirroring, you now have two copies of bad data. With after-imaging, you can restore last night’s backup and roll forward today’s after-image files to a point in time just prior to running the program. After-image should be a part of every high availability environment.

Always use multi-volume extents

OpenEdge supports one or more after-image extents per database when after-imaging is enabled. Each extent of the after-image file is its own area with a unique area number, but it is more common to refer to them as extents. You need more than one extent for each after-image file to support a high availability environment.

Each extent has three possible states: empty, busy, or full:

- An *empty extent* is empty and ready for use.
- A *busy extent* is one that is currently active. There can be only one busy extent per database.
- A *full extent* is a closed extent that contains notes and cannot be written to until the extent is marked as empty and readied for reuse by the database administrator.

Multiple extents allow you to support an online backup of your database. When an online backup is executed the following occurs:

- A latch is established in shared memory to ensure that no update activities take place.
- The modified buffers in memory are written (pseudo-checkpoint).
- An after-image extent switch occurs (if applicable).

- The busy after-image extent is marked as full and the next empty extent becomes the busy extent.
- The primary recovery area is backed up.
- The latch that was established at the start of the process is released.
- The database blocks are backed up until complete.

Sequential access

The after-image file is sequential like the primary recovery area. See the "Primary recovery (before-image) information" section on page 2–35 for recommendations.

Isolate (disaster recovery)

Isolate the primary recovery area from the other portions of your database for performance reasons. Isolation is required with the after-image extents. The after-image files must be isolated to provide maximum protection from media loss. If you lose a database or before-image drive, you can replace the drive, restore your backup, and use the after-image file to restore your database. If you lose an after-image drive, you can disable after-imaging and restart the database. You will only lose active transactions if the after-image extents are isolated from the rest of the database. Sometimes this is difficult to do because you might have several file systems accessing the same physical drive, but the isolation needs to be at the device and file system levels.

Sizing after-image extents

The after-image area differs from all other areas because each extent can be either fixed or variable length. Each extent is treated as its own area. It is fairly common for people to define several (more than 10) variable-length extents for the after-image file.

To choose a size, you must know how much activity occurs per day and how often you intend to switch after-image extents. You can define all of your extents as variable length and see how large they grow while running your application between switches. To accommodate above-normal activity, you need extra extents. If they can all be variable length why would you want to make them fixed length? If you are concerned about performance, you would want to have the after-image extents fixed length so you are always writing to preformatted space.

Preformatting allows you to gain:

- Performance by eliminating the formatting of blocks during the session.
- Use of a contiguous portion of the disk.

Most operating systems are fairly good at eliminating disk fragmentation. However, if you have several files actively extending on the same file system, there is a high risk of fragmentation.

Optimizing memory usage

Optimizing memory usage can be best described as taking advantage of the memory that you have. In most cases, a system benefits from additional memory. Sometimes, it is not possible to purchase additional memory.

This section focuses on the trade-off between user and broker memory and how to use the available memory to your best advantage.

Why is buffer hit percentage important?

A short explanation is that the greater the percentage of time that the buffer is utilized, the lower the percentage of time the disks are utilized. Since memory is faster than disks, you will get better performance with a better buffer hit percentage.

A longer explanation has to do with the meaning of the numbers. For example, a 90 percent buffer hit percentage equates to 10 disk reads for every 100 requests to the database manager. If you increase your buffer hit percentage to 95 percent, you are only doing 5 disk reads for the same number of requests, which is a 50 percent reduction in requests to the disk. A small change in buffer hit percentage can equate to a large reduction in disk I/O. This is especially noticeable at the high end. Changing from 95 percent buffer hit percentage to 96 percent represents a 20 percent reduction in disk I/O, so it is important to read into the numbers, not just monitor the numbers themselves.

Increasing memory usage

Memory is a limited resource. It is important to use it properly. In the case of a database application, it is important to increase broker parameters first because the payback extends across all users and the system as a whole. This fact is demonstrated with the previous database buffer hit percentage example. This example shows how a small change on the broker side can dramatically affect the entire system. The size of the change on the broker is usually smaller relative to any self-service client changes you might make.

For example, a 1000 buffer increase for the `-B` parameter on the broker of an 8KB-block-size database will cost you 8MB of RAM; an 80KB increase on the `–mmax` parameter for a 100 user system will cost the same 8MB of RAM. In the majority of cases, a buffer increase has a greater overall impact than the client change. This is a simple example, but it points out that you should always tune your broker parameters first. Once that is complete and you still have RAM to allocate, you can focus your attention on the self-service client parameters.

Decreasing memory

Determining where to cut back on memory is difficult. In the previous section, we discussed increasing broker memory first and then looked at client parameters. However, when decreasing memory you should look at client parameters before broker parameters.

Considering where you can reduce memory might or might not be obvious. First to consider are operating system buffers. The database engine bypasses operating system buffers with the use of `-directio`, so the need for operating system buffers is limited. Operating system buffers will still be used for temporary file I/O for any client processes that reside on that machine. Most operating system manufacturers allow you to modify the number of buffers that can be allocated to the operating system. If you are using the `-directio` startup option, then you can reduce the amount of operating system buffers to approximately 10 percent in most cases. One major exception is systems with very limited memory (less than 256MB), where leaving the parameter at its default value is the best practice.

Using OpenEdge memory-mapped procedure libraries also helps to reduce memory usage by allowing the users to use a common version of the code rather that loading a copy into `-mmax`. This reduces the amount of `-mmax` needed for each client. (See *OpenEdge Deployment: Managing 4GL Applications* for details about memory-mapped procedure libraries).

Private buffers (-Bp)

Private buffers allow a read-intensive user to isolate a portion of the buffer pool. Up to 25 percent of buffers can be allocated as private buffers. Private buffers work as follows:

1. The user requests a number of buffers to be allocated as private.
2. As the user reads records, if the corresponding buffers are not already in the buffer pool, the records are read into these buffers.
3. Instead of following the rules for buffer eviction, the user only evicts buffers that are in their private buffers. By default, the buffer that was least recently used is evicted. Private buffers are maintained on their own chain and are evicted by the user who brought them into memory.
4. If another user wants a buffer that is currently in another user's private buffers, this buffer is "transferred" from the private buffers to the general buffer pool. The transfer is a process of removing the buffer from the user's private buffer list and adding it to the general buffer list. The buffer itself is not moved.

The general idea is to share memory where you can, use memory efficiently where possible, increase memory on the broker side first, and then increase client memory usage.

Optimizing CPU usage

As a database or system administrator, there are only a few things you can do to more efficiently use the CPU resources of your machine. The major consumer of CPU resource for your system should be the application code. Therefore, the greatest impact on CPU consumption can be made with application changes. Other resources are affected by application code as well, but there are things that you can do as an administrator to minimize problems associated with other resources. This is not the case with CPU resource.

Understanding the -spin parameter

The broker allocates shared memory, and each portion of memory can be updated independently. This design is both positive and negative. It adds complexity to the database on how to keep one portion of memory from being updated by two users simultaneously.

OpenEdge solves this problem through the use of *spin locks*. Each portion of memory contains one or more locks to ensure that two updates cannot happen simultaneously. These locks are called *latches*, to differentiate them from record locks. When a user modifies a portion of shared memory, the user gets a latch for that resource and makes the change. Other users that need the resource respect the latch. By default, if a latch is established on a resource and a user needs that resource, that user tries for the resource once and then stops trying. On a single CPU system, you want to try the operation only once because the resource cannot be freed until the resource that has the latch can use the CPU. This is the reason for this default action.

The default action is not very efficient on a multiple CPU system because a significant amount of resource time is used to activate the user on the CPU. Effort is wasted if the resource is not available. Typically, the resource is only busy for a very short time. It is more efficient to ask to obtain the resource many times rather than ask once, go to the end of the CPU queue, and when arriving at the top of the CPU queue, ask a second time to get the resource. Using the `–spin` parameter, you can ask for a resource thousands of times. The `–spin` parameter determines the number of retries before giving up.

How to set –spin

Generally, a setting between 2,000 and 10,000 works for the majority of systems, but this varies greatly. The best way to set `-spin` is to start with a setting of 2,000 and then monitor the number of “naps per second” per resource. If the naps per second value for any given resource exceeds 30, try changing the value of `–spin`. You can do this while the system is running through `promon`, provided the value of `–spin` is not 0 through the `promon` R&D option.

Note: The adjustment of `–spin` is generally an increase, but there are documented cases where a decrease in `–spin` has had a positive effect on performance.

Viewing latches in Fathom Management

The easiest way to view latches is to use Fathom Management. Figure 2–11 shows an example of the **Latches** section within the **Locks and Latches Operations** view page for a selected database resource. This page is cumulative; you need to note the number of latches for each sample and the amount of time between samples. (You can change auto refresh frequency rate in **User Preferences**.) You can calculate the values.

Latch summary

Latch	Type	Holder	Times locked	Time Outs
MTL_MTX	MT_LT_SPIN	14	14,429	7
MTL_USR	MT_LT_QUEUE	14	17	--
MTL_OM	MT_LT_SPIN	12	633,364	591
MTL_BIB	MT_LT_SPIN	14	316,882	1
MTL_SCC	MT_LT_QUEUE	14	63	--
MTL_LKP	MT_LT_SPIN	--	--	--
MTL_GST	MT_LT_SPIN	14	12,954	--
MTL_TXT	MT_LT_SPIN	14	110	--
MTL_LHT	MT_LT_SPIN	--	--	295
MTL_SEQ	MT_LT_SPIN	14	1,260	--
MTL_AIB	MT_LT_SPIN	14	114,829	--
MTL_TXQ	MT_LT_SPIN	14	28,185	--
MTL_BIW	MT_LT_SPIN	--	--	--
MTL_LKF	MT_LT_SPIN	12	1,204,542	861
MTL_BFP	MT_LT_SPIN	8	2	--
MTL_BHT	MT_LT_SPIN	10	1,372,930	773
MTL_PWQ	MT_LT_SPIN	--	--	--
MTL_AIW	MT_LT_SPIN	--	--	--
MTL_CPQ	MT_LT_SPIN	8	18,509	--
MTL_LRU	MT_LT_SPIN	10	1,347,214	847
MTL_LR2	MT_LT_SPIN	--	--	--
MTL_LR3	MT_LT_SPIN	--	--	--
MTL_LR4	MT_LT_SPIN	--	--	--
MTL_BF1	MT_LT_SPIN	--	--	22
MTL_BF2	MT_LT_SPIN	--	--	18
MTL_BF3	MT_LT_SPIN	--	--	13
MTL_BF4	MT_LT_SPIN	--	--	11
MTL_BF5	MT_LT_SPIN	--	--	--
MTL_BF6	MT_LT_SPIN	--	--	--
MTL_BF7	MT_LT_SPIN	--	--	--

Figure 2–11: Latch summary in Fathom Management

Note: Naps are listed as **Time Outs** in Fathom Management.

CPU bottleneck: Look at your disk drives

Indications that you are out of CPU resources might only be masking an issue with another resource. In most cases it is a disk issue. If you see a CPU bottleneck, first ensure that you do not have a runaway process. Second, make sure that your other resources are working efficiently.

3

Performing System Administration

This chapter covers the following system administration topics:

- Understanding the database administrator’s role
- Ensuring system availability with trending
- Ensuring system resiliency
- Maintaining your system
- Profiling your system performance
- Advantages and disadvantages of monitoring tools
- Common performance problems
- Other performance considerations
- Periodic event administration

Understanding the database administrator's role

The specific tasks of the database administrator vary from company to company, but the role includes the following common responsibilities:

- **Providing predictable system availability** — The administrator must understand the state of the system at any given moment, and where it is going in terms of resource utilization to provide reliable system availability to users. This involves more than just knowing how the system works. It demands a deeper understanding of the trends in system utilization over time. It also requires an understanding of the business as a whole, which can only be provided by the management of the company. However, management will need your help in translating business plans into technology choices.

- **Providing a resilient system that can recover from disasters** — The database administrator must look at potential problems and determine if and how these problems are addressed by the disaster recovery plan. It is important to document items that will not be addressed and those that will be addressed in the disaster recovery plan. The focus of the plan should be on recovery rather than the backup process; a backup is useless if you cannot recover it.

- **Providing reliable system performance** — Users need to know how long a task is going to take so they can plan around that expectation. The application plays a huge role in overall performance. However, it is the administrator's job to ensure that tasks take the same time every day by taking advantage of system resources and eliminating bottlenecks.

- **Performing periodic maintenance** — This task might only be performed once a year, but must also be taken into consideration in a complete administration plan. The small details of each task might change at the time of execution, but the overall process should be laid out in advance.

Ensuring system availability with trending

The primary goal of the database administrator is to make sure that data is available to users. Most OpenEdge-based applications require multiple resources to function properly. This section centers on trending of those resources.

Running out of resources is the second most likely cause of system failure. (Hardware failure is the most common.) It takes persistence to ensure maximum system availability. Exercising this persistence is precisely the role of the database administrator.

The OpenEdge database is very reliable and as we consider our options when making hardware decisions, we will be covered there, too. Our focus shifts to trending, maintenance, and contingency planning for the database.

When determining the source of a problem, the first question asked is: What has changed? If you are trending your system, you can determine if there is a difference in the amount of work (reads, writes, commits) being done, the number of users on the system, or if the system is consuming more resources than usual.

The trending of resources is extremely important; if you know how fast you are using additional resources, you can determine when you will run out. Trending also allows you to plan for growth and avoid potential problems within the database and at the system level. On most systems, disk capacity and database storage areas are the most dynamic areas in terms of growth. These areas are the best places to consider trending.

Trending database areas

It is important to understand not only how much data is in the area today, but also how much growth to expect. On existing databases, you should first consider the storage area high-water mark. The high-water mark is established by the number of formatted blocks (RM, index, and free) in an area. In an area with many empty (unformatted) blocks, data is allocated to the empty blocks before the system extends the last extent of the area. As stated earlier in this chapter, it is important to have the last extent of each area defined as variable length to accommodate unanticipated growth. The goal is to never use the variable extent but to have it there if necessary.

Each environment has a different amount of required uptime. Some systems can come down every evening while others need only be shut down once a year for maintenance. With careful planning you can leave your database up for long periods of time without the need for a shutdown. In most cases, the OpenEdge database does not need to be shut down for maintenance.

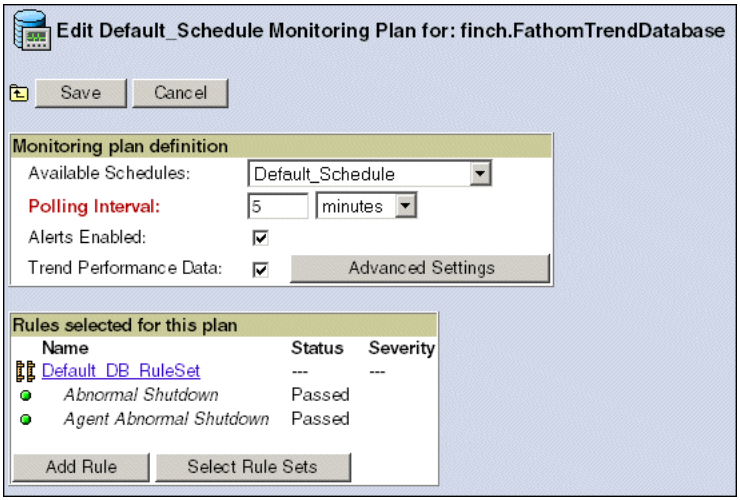
The operating system might need to be shut down periodically for maintenance or an upgrade. Examples of this type of maintenance are: clearing memory, installing additional hardware, or modifying the operating system kernel parameters. On Windows, it is generally necessary to reboot the system every 30 to 90 days to avoid problems, while on most UNIX systems once a year is more common. You need to plan for growth to cover the period of uptime that is appropriate for your system.

Enabling trending for your database in Fathom

Start trending for your database. This can be done using Fathom, which will monitor each database and store the information for trending purposes, if necessary.

To trend database storage areas from within the Fathom Management console:

1. Select **Resources** from the menu bar.
2. Click **New Resource Monitor**.
3. Select **Database** if the database is managed.
4. Select the database that you want to trend or migrate (if it is not listed).
5. From the database's **Monitoring Plans** page, click **Edit**. The **Edit** page appears:



6. Select the **Trend Performance Data** check box.
7. Click **Save**.

Using VST code

You can also view database storage information using custom VST code. The following example shows the basic code you need to trend the size of your database areas:

```
FOR EACH _Area WHERE _Areanum = the area to be monitored:
  FOR EACH _AreaStatus OF _AREA:
    DISPLAY _Area-name _TotBlocks _HiWater.
  END.
END.
```

Note: The program above will display the information to the screen. However, to trend this data without Fathom, you need to write the output to a file or into a database for future reference.

Trending application load

One statistic that most administrators do not keep track of is the amount of work that is completed per day on their system. By trending database activity such as commits and database requests, you can determine when the greatest workload on the system occurs and the growth pattern of this workload over time.

Workload information can be valuable information. If you encounter a problem, you can see whether there is an abnormal workload on the system or if there is some other issue. In cases where additional customer records are added to the database, you might notice that the workload on the system is increasing even though the number of users and the number of transactions are not increasing. This indicates that there might be an application efficiency issue that needs to be addressed before it becomes a problem. It will also help you to understand your need for additional resources prior to loading even more records into the database. By knowing this information prior to the event, you can plan effectively.

There might be other internal database information that you would like to monitor, such as checkpoints or record-locking activity. These items trend in a similar fashion, and if you have enabled trending from within Fathom, these items will be available to report on, too. See Chapter 4, “Guidelines for Applying Fathom,” for more information on reporting your Fathom data.

Trending operating system information

The approach and rules are the same for disk and memory trending, but the tools to obtain this information vary from operating system to operating system. Fathom really simplifies this process. If you are in a heterogeneous environment, you will fully appreciate having one interface to monitor database and operating system resources. To gather operating system information without Fathom, you must gather information from an operating system command like `sar` or `iostat` and put that information into a database or operating system file for future reference. With Fathom, you only need to monitor and trend the resource to gather this trending information for future reference.

The steps to trend an operating system resource are similar to the steps to trend a database resource. Figure 3–1 shows a sample monitoring plan page for a CPU resource.

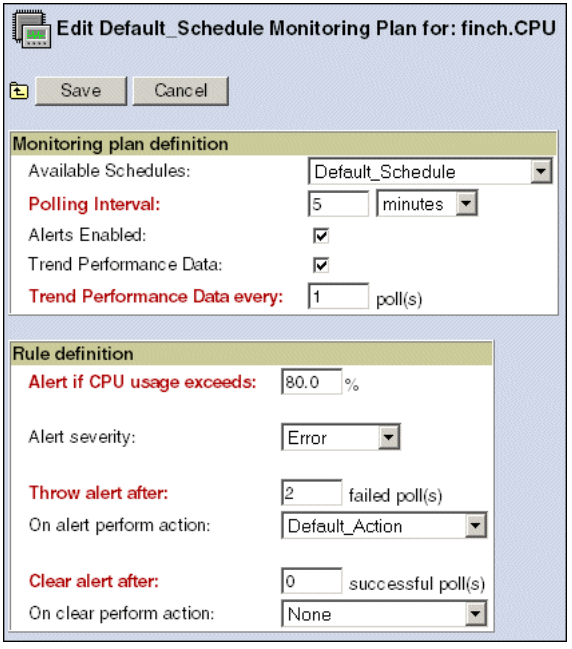


Figure 3–1: Trending a CPU resource

Trending system memory

Memory usage increases as users and functionality are added to the system. There is a dramatic change in performance when the memory resource is exhausted. The amount of paging and swapping are key indicators to monitor. An administrator should focus on physical paging as the primary indicator of memory utilization.

By monitoring operating system utilities like `sar` and `vmstat` during busy periods of the day, you can determine the health of the memory resources. Example 3–1 shows the `sar` command and its output.

Example 3–1: sar command and output

```
Command:
mymachine:    sar -pgw 10 1
```

```
Output:
SunOS mymachine 5.7 Generic_106541-15 sun4d    07/09/03

15:28:28  atch/s  pgin/s  ppgin/s  pflt/s  vflt/s  slock/s
          pgout/s ppgout/s  pgfree/s  pgscan/s   percentufs_ipf
          swpin/s bswin/s  swpot/s  bswot/s  pswch/s

15:28:38    4.48     8.26    106.77     0.60     4.28     0.00
           12.14    27.66     27.66     0.00     0.00
            0.00     0.0       0.00     0.0      1557
```

The `sar` command has many options. In the previous example, virtual paging (`-p` option), physical paging (`-g` option), and swapping (`-s` option) are requested. The command requested 10-second samples and only one sample. The output from the command is broken into one line per command line option. The virtual faulting numbers are not as important as the physical faulting numbers because most systems can do thousands of virtual faults per second without a noticeable affect on performance.

The primary item to focus on is the `pgscan/s` value. This value represents the number of pages that are being scanned per second to find pages for use by active processes. When this number starts increasing by large amounts it can indicate the need to either reduce your memory usage or increase the amount of available memory (that is, buy more memory). The swap values should stay near or at zero because even small amounts of swapping have a negative effect on performance and require you to reduce memory consumption, or increase available memory.

Currently Fathom only tracks total memory consumption for physical and virtual memory. This can be a misleading indicator of memory utilization because operating systems will often use up “idle” memory for noncritical items like operating system buffers. This could falsely represent memory usage. Trending this information can be helpful in circumstances where virtual memory usage is in question.

Trending system disks

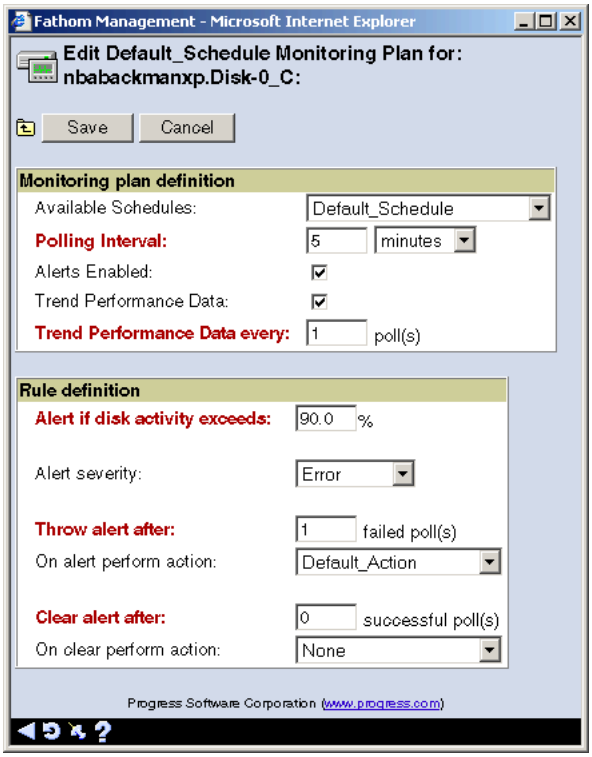
Disks, like storage areas, need to be monitored and trended to see usage patterns and anticipate future needs in terms of both storage and throughput capacity.

Each system resource can be viewed and trended through Fathom.

To enable trending for disk storage capacity:

1. Select **Resources** from the menu bar.
2. Select **System** from the list frame and select the **Disk** resource.
3. Select the disk for which you want to enable trending.
4. Click **Edit for** the monitoring plan.

5. Select the **Trend Performance Data** check box:



6. Click **Save**.

Within Fathom, each disk must be enabled separately if it was not enabled at setup time. If you enabled all disks upon installation, then you can disable by deselecting the **Trend Performance Data** check box for each disk you do not want to trend. You can still spot check unmanaged resources from within Fathom, if needed.

Setting alerts for variable extent growth

You need to be concerned with the use and growth of your variable-length extents. Again, the goal is never to grow into these. You should have alerts on your areas to tell you when you need to add space to eliminate the possibility of growth of these extents.

Another way to alert yourself when one of your variable extents is growing is to place all of the variable extents in one file system. Then put an *alert* on the file system to tell you when one of these extents has grown. You can monitor each variable extent individually, but this single-file-system method simplifies the process by having one alert for all databases. Any simplification you can build into the process is beneficial.

Additional factors to consider in trending

The following list identifies other trending-related factors you should consider:

- One of the greatest impacts on performance and availability of data over time is a fluctuation in the number of users.
- The volume of data involved is another factor to consider when evaluating performance.
- A poorly written query on a large amount of data causes a performance impact.

Keep in mind that performance impact is not always linear. An example of this is a repeating query that scans the entire table for a particular record or pattern. When the table is small, all of the information can be stored in memory. But once the table grows beyond the size of the buffer pool, it will cause a significant amount of degradation to the system due to continual physical reads to the disk. This not only affects the query in question, but all other users accessing the database.

Process monitoring

Applications have dependencies on processes as well as resources. If your application depends on a background job to print reports, monitor this job to ensure your system is working properly.

This is one area that most monitoring tools omit, as the information is vastly different from system to system. Consequently, it is very important to take time to inventory the critical processes on your system, uniquely identify each process, and determine if it is working properly. Most processes need no intervention as long as they are running, and a monitor can be as simple as looking for these processes in the process table.

The status of other processes can be more difficult to determine. In some cases there is a tool or operating system command that can be used to determine a processes status. For example, the `proadsv –query` command can be used to determine the status of the AdminServer process. This command could be put in a job within Fathom to periodically query the status of the AdminServer process, or in a script or batch file at the operating system level.

Testing to avoid problems

The best way to avoid problems on your system is to test prior to implementation. Most users think of testing only their application code. However, it is also necessary to test other aspects of the system including administration scripts and applications, such as backup software, hardware, middleware, and other infrastructure.

Types of testing

Testing is a long and meticulous process if done properly. Users can forget to test the basics. There are three types of testing:

- Limits testing

  Exceeds hardware and software system limits and ensures system reliability.

- End-to-end testing

  Examines an entire operation, checks the integrity of individual processes, and eliminates compatibility issues between processes. For example, looking at the order entry process from the customer's phone call to the delivery of goods.

- Unit testing

  Examines a process in isolation. Unit testing should be done during early development and again as the initial step in the user acceptance process prior to implementation.

You can also run tests on the individual system hardware components in isolation to ensure there are no faults with any item. Once this testing is complete, you can run a stress test to test the items together. A well designed test includes your application components, making an end-to-end test possible. For a complete check of the system, execute the stress test while running at full capacity and then simulate a crash of the system to check system resiliency.

Ensuring system resiliency

Resiliency is the ability to recover in the case of a disaster. The goal in a disaster is to appear to the outside world as if nothing has happened. In the case of a Web site, if a customer or potential customer cannot access your site, that customer might be lost forever. You need to gain the trust of your customers, both internal and external, so they will work with you in making effective use of your system resources. The trust you gain from being able to deal effectively with bad situations will increase your ability to succeed.

This section will answer the basic questions regarding the creation of an effective recovery strategy and show you where to invest your time and energy to create a complete solution. It is referred to as focusing on a recovery strategy. You want to focus on recovering from various situations, and not merely on the process of backing up your system.

Many users tend to focus on the backing up activities and not on recovery. If you had a good backup of your database, but you neglected to back up your application code, you are in as much trouble as if you had failed to back up your database because there is no effective way to access your data. Efforts to possibly obtain third-party application code back from the vendor, or to re-create an in house application, will be a waste of time if you cannot serve your customers.

Why do backups?

More and more companies rely on online systems to do very basic portions of their business. If a system is down for any period of time, it affects the ability of these companies to do business. Consequently, it is important to protect your data and applications. Problems can occur for several reasons, including hardware failure, software failure, natural disaster, human error, or a security breach. The goal of a complete backup strategy is to appear to the outside world as if nothing has happened, or at worst to minimize the amount of time that you are affected by the problem. A secondary, but equally important goal, is to reduce or eliminate data loss in case of a failure.

The best way to increase system resiliency is to prevent failure in the first place. The best way to do this is to implement redundancy, like disk mirrors, into your design to minimize the probability of hardware problems that cause system failure.

Even with redundant hardware it is possible to encounter other issues that will cause a system outage. This is the reason to implement a complete backup strategy.

A complete backup strategy needs to take many things into account, including these factors:

- Who performs the backups.
- Which data gets backed up.
- Where the backups are stored.
- When the backups are scheduled.
- How the backups are performed.
- How often the current backup strategy is reviewed.

A backup strategy must be well-designed, implemented, and periodically reviewed and, if necessary, changed. The only time a problem is found is when the backup is needed. By then it is too late. When systems change it is often necessary to modify the backup strategy to account for the change. You should also periodically test your backup strategy to ensure that it works prior to a problem that would precipitate its use.

Creating a complete backup-and-recovery strategy

A great deal of time and money are spent on backup and recovery strategies. However, they are often not tested and revised based on the discoveries made.

A complete backup strategy should try to balance the probability of the problem occurring with the amount of data loss in a given situation and the amount of time and resources spent on backups. A disk failure is a relatively likely event compared to a fire or a flood. This is the reason to have redundancy at the disk level—to reduce the probability of failure. A fire or flood is less likely, and most people understand that they would lose some data (provided they were informed about this probability prior to the disaster).

It is also important to include the users in the disaster-planning process. They are the real owners of the data and can help with the probability/data loss/cost trade-off decision.

It is possible to provide near-100-percent reliability, but there is a large cost to doing so. Take the case of an emergency services organization. It needs as close to 100 percent reliability as possible, since a failure on its part could cost lives. It needs to have complete, duplicate systems at a second location with available staff in case their primary location is knocked out by a disaster. Your situation might allow you to lose some data and availability of the system in a trade-off for lower cost. It is important to focus on the entire system and not just your databases and applications. You must also weigh the cost/benefit of each strategy. While you are doing this analysis, you need to include the people who run the business to determine their requirements. It is sometimes helpful to put a price on lost data and downtime because it makes it easier to do the final cost/benefit analysis.

The following sections identify the key areas to consider when devising a complete backup-and-recovery strategy.

Who does the backup?

In most cases the system administrator performs backups of the system on a regular basis. When the system administrator is unavailable, some businesses have other personnel handle the backups. It is best to have one central person responsible for the overall process and all archiving activity.

What does the backup contain?

An application consists of many diverse components, including databases, application code, user files, input files, third-party applications, and so on. Remember that your application is made up of an OpenEdge release, your application source and object code, middleware such as the AppServer, and associated operating system files.

The best way to determine what needs to be backed up is to walk through the vital processes within your organization and note activities and systems such as these:

- The systems that are involved.
- The software application files.
- The data that is used throughout the process.

Where does the backup go?

The media that you use for backups must be removable so it can be archived off site to better protect data from natural disaster. Consider the size of your backup in relation to your backup media. For example, tapes with a large storage capacity are a practical and reliable option to back up a 20GB database.

Tape compatibility is also a consideration. You might want to use the backup tapes on more than one system. This will allow you to back up on one system and restore to another system in the case of a system failure. A Digital Linear Tape (DLT) is supported on many platforms and can be used to help move data from one system to another or to retrieve an archive.

Archiving off site is as important as the backup itself. If a fire, flood, or other natural disaster destroys your building, you can limit your data loss by having your backup at a separate location. This can be a formalized service, or as simple as placing the completed backup tapes at a person's house. However, it is important to make sure you have access to your archives 24 hours a day, seven days a week. This is especially important if data is stored in a private residence and the catastrophe occurs when that person is not at home.

How to label a backup

Proper labeling of your backup media is essential. Every label should contain:

- The name of specific items stored on the tape. A tape labeled "nightly backup" has no meaning without the ability to cross reference the items contained in the nightly backup.
- The date and time when the tape was created should appear on the tape. In situations where multiple tapes are made in one day, you will need to know which tape is more current.
- The name or initials of the person who made the tape, to ensure accountability for the quality of your personnel's work.
- Instructions to restore the tape. There should be detailed restore instructions archived with the tape. The instructions should be easy to follow and might include specific command information required to make the backup.
- Volume number and total number of volumes in the complete backup set. Labels should always read "Volume *n* of *n*."

Note: Always write-protect the tape before archiving and write-enable the tape before using it for backup.

When do you do a backup?

Perform a backup as often as practical, balancing the amount of data loss in a failure situation with the interruption to production that a backup causes. To achieve this balance, consider these points:

- Static information, like application files that are not being modified, only needs to be backed up once a week, or less often.
- Most database application data should be backed up at least once a day.

In cases where data is backed up once a day, it is possible to lose an entire day's work if the disks fail or some natural disaster strikes at the end of the day. If you performed multiple backups throughout the day but only archived once a day, you would be better protected from a hardware, software, or user error, but your protection from most natural disasters would be identical. By moving the intra-day tapes from the machine room to a different portion of your office, you decrease the probability of a fire in the machine room destroying your tapes.

Using PROBKUP versus operating system utilities

Among knowledgeable administrators, there are debates concerning the use of the OpenEdge `PROBKUP` utility versus operating system utilities. Also, there is a great deal of misinformation about when and how to back up a system while leaving it online. This section discusses these issues in detail.

Understanding the PROBKUP utility

The `PROBKUP` utility was created to back up databases. It has many nice features that make it unique, including:

- The `PROBKUP` utility is *database-aware*.

  Database aware means that it can scan each block to ensure it is the proper format during the backup process. It takes longer to do this scan, but the added integrity checking of potentially seldom-used blocks is worth the small performance degradation.

- `PROBKUP` has an online option.

  This online option allows you to to back up a database while the database is available to users. Since the `PROBKUP` utility is database-aware, it knows when the structure of the database changes regardless of the number of disks/areas/extents. Therefore, the syntax of the command does not need to change when the structure of the database changes.

How PROBKUP works

The following steps briefly identify the PROBKUP process:

1. Establish the database latch (online only).
2. Do a pseudo checkpoint (online only).
3. Switch AI files (if applicable).
4. Back up the primary recovery area.
5. Release the database latch (online only).
6. Back up the database.

The database is backed up from the high-water marks downward. Free blocks are compressed to save space. Online backups represent the database at the time the backup started. All transactions started after the backup has begun will not be in the database when a restore and transaction rollback occurs.

The reason for the pseudo-checkpoint in an online backup is to synchronize memory with disk prior to backing up the database. This synchronization is critical since the PROBKUP utility can then back up all of the data that is in the database or in memory at that moment. Other utilities can only back up the information on disk, thus missing all of the "in memory" information.

Note: This pseudo-checkpoint affects the "buffers flushed" information in the VSTs for the database, which will in turn affect PROMON and Fathom. Therefore, you need to note the number of flushed buffers before and after the backup to ensure that you only track those buffers flushed during normal operations. The importance of this statistic is discussed in the "Monitoring buffers flushed at checkpoint" section on page 3–28.

Adding operating system utilities to augment PROBKUP

The PROBKUP utility backs up only the database and primary recovery area. This utility does not back up the after-image files or even the database log file.

All complete backup strategies use operating system utilities to back up additional important files on the system such as programs, application support files, and user home directories. Some administrators choose to back up their database to disk with PROBKUP and use an operating system utility to augment the database backup with other files to get a complete backup on one archive set. Also, you should back up your after-image files to separate media from the database and before-image files to increase protection, as discussed in the "After-imaging implementation and maintenance" section on page 3–20.

Using PROBKUP or operating system utilities

Table 3–1 lists the advantages and disadvantages of the of PROBKUP utility.

Table 3–1: Advantages and disadvantages of PROBKUP

Advantages	Disadvantages
Has online backup capability — Because PROBKUP is the only utility that is database aware, it is the only utility that is designed to back up a running database. There are methods using the PROQUIET utility that achieve similar results in conjunction with an operating system utility, but generally those methods are more cumbersome and time consuming. **Performs block level checking of database** — The PROBKUP utility retrieves each block from the database and, because it is database aware, it performs some tests on each block to ensure it has the proper format. This is not a total check of each record, but a block level verification. **Knows when structure has been modified so it is easier to maintain** — Because the PROBKUP utility uses the database structure file to determine the location of all of the information inside the database, it is aware when changes are made to the structure. With other utilities, you need to keep track of the locations of all of your database areas and extents. **Does not back up empty blocks** — The utility only looks at blocks below the high-water mark, so if you have additional allocated space in the database, it is not necessary to back it up. Other utilities will back up the entire database including blocks above the high-water mark.	**Only backs up the database** — The PROBKUP utility only backs up the database and the primary recovery area. All other components must be backed up using some other utility. **Performs slightly slower** — The slight reduction in performance versus operating system utilities is due to the block level checking that is done by PROBKUP. The performance trade-off is well worth it.

Table 3–2 lists the advantages and disadvantages of the operating system utilities.

Table 3–2: Advantages and disadvantages of operating system utilities

Advantages	Disadvantages
Ability to do a complete backup — With an operating system utility you can back up the entire application. **Tight integration with the operating system** — Operating system manufacturers are better able to support your use of their utilities. This integration makes it easier to combine the backup with other utilities in some cases. **Integration with third-party tools** — The operating system might allow you to integrate with an enterprise backup tool. PROBKUP can also be integrated, but it requires you to first back up the database to disk and then move it to tape with an operating system or third-party tool. **Performance is generally better** — Since the utility does not do any block level checks, it can get information off to tape faster than a PROBKUP in most cases.	**More difficult to maintain** — When adding extents or otherwise modifying the database structure, you need to ensure that your backup takes the new locations into account. **No online capability** — The only way to back up an OpenEdge database in an online fashion using operating system utilities is to "quiet" the database with the PROQUIET command. Once the quiet point has been established, you can back up the database with an operating system or third-party tool. The quiet point can only be disabled after the entire database is backed up. Your users cannot log into or out of the database or make any modifications for the duration of the quiet point. **Limited integrity checking** — Most utilities allow you to check to ensure that the blocks on tape are the same as the blocks on disk. If there are problems with the blocks on disk, these integrity checks will not find the problem.

After-imaging implementation and maintenance

After-imaging provides an extra layer of protection around the database. Every high availability system should implement after-imaging. It is essential that you have a good backup and recovery plan prior to implementing after-imaging.

Once started, the OpenEdge after-image feature keeps a log of all transactions that can be "rolled-forward" into a backup copy of the database to bring it up-to-date.

The primary reason to enable after imaging is to protect you from media loss. This can be the loss of a database, a primary recovery area disk, or a backup tape. In the case of a lost backup, you can go to the previous backup and roll-forward to bring the system up-to-date. This solution assumes that your after-image backup was not stored on the same piece of media as the database backup that you were unable to recover. (This is the main reason for doing a backup of your data on one tape or tape set and your after-image files to a second tape.)

After-imaging provides user error protection. This feature is not available from any other source.

This section describes how this feature works. For example, if a user or developer runs a program that updates all of your customer records to the same name, mirroring would not protect you. It would store two copies of the bad data. With after-imaging, you can restore last night's backup and roll the data forward to a point in time just prior to the moment the program was run. Without after-imaging, the most likely recourse would be to restore from the last backup and manually retype the day's transactions.

After-imaging can also be used to keep a "warm" standby copy of your database. This standby database could be stored on the same system as the primary copy. However, for maximum protection you would normally store it on a different system. Progress can provide replication through its Fathom Replication product with greater flexibility, but after-imaging provides a way to do a "poor man's replication." This form of replication allows you to periodically update your standby database by transferring after-image files from the primary database and applying or rolling-forward those files to the standby database. In the case of a system failure, you could apply the last after-image file to the standby database, and start using the database with significantly less downtime to the users. If it were not possible to apply the last after-image file to the database, you would lose only the data entered since the last application of the after-image file.

Note: Users have been using this method since Progress introduced after-imaging to the product. It is known to be reliable.

Table 3–3 and Table 3–4 present the advantages and disadvantages of using after-imaging replication and site replication, respectively.

Table 3–3: Advantages and disadvantages of After-imaging replication

Advantages	Disadvantages
• Included with the product (inexpensive). • Time-tested reliability.	• Initial setup time (custom code). • Ongoing maintenance. • Read-only access only to the standby database. • No synchronous option.

Table 3–4: Advantages and disadvantages of Fathom Replication

Advantages	Disadvantages
• Synchronous replication ability. • Ease of maintenance. • Read/write access on primary and standby.	• Cost. • Greater infrastructure requirements.

After-image-based replication is a viable alternative for users who want an extra layer of protection for their systems but cannot or do not need to implement site replication.

Testing your recovery strategy

The only valid backup is your last tested backup. This underscores the need to test your backup strategy on a regular basis. This does not mean that you should delete your production database and restore from backup to "see if it works." It is best if you can restore your database to a different location on disk, or better yet to a different system.

How often should you test your backup? If you are using an operating system utility to back up your database, it is a good idea to test your backup any time you modify the structure of your database. Since `PROBKUP` is database-aware you might not need to test for structure changes. However, it is still wise to test at least twice a year.

You need to test the backup and the process when there are changes in staff or times when the administrator might be out of town. Any member of the IT staff should be able to perform a well-documented recovery process. If you do not have an IT staff, you need to ensure that each person who might be required to restore the system can follow the documented recovery procedures.

Your recovery procedures should be scenario-based. Common recovery scenarios, such as a loss of a disk (database, after-image, or application files, for example), fire, flood, and so on, need to be clearly documented with step-by-step instructions to describe how to determine the root cause of the problem and how to recover from the event.

You hope you will never need to use your recovery plan. However, if the plan is well documented and the users were involved in the cost/benefit trade-offs in the beginning, then you will never be second-guessed or lose the trust of your user base because of a disaster.

Maintaining your system

Do not let OpenEdge's ease of maintenance lead you to become complacent about maintaining your system. The health of the system should be monitored every day. This can be done using a tool such as Fathom or manually using scripts.

The areas that are most important to monitor are the areas that will cause a performance problem or cause the database to stop running. While the issues that cause the database to stop running are the most important to identify **before** they fail, performance problems are often equally important to system users. A slow system erodes the users' confidence in the system, and they will begin to look elsewhere for their information. A slow Web site might drive users to go elsewhere for information, and they might never return.

This section identifies which areas to monitor and the ways to monitor these resources.

Note: This section does not present a comprehensive list of resources to monitor. Keep in mind that your system will most likely have additional resources that warrant monitoring.

Daily monitoring tasks

It is too easy to become complacent when you have a smooth, well-running system. Unless you setup a routine of daily tasks, you might end up relegating system monitoring to the bottom of your priorities. It might slip into a problem state when you are least expecting it. Establish and execute a simple set of daily tasks to ward off impending problems.

The following sections describe the resources you should monitor on a daily basis.

Monitoring the database log file

The database log file contains a wealth of information about database activities. OpenEdge places many pieces of information in the file beyond simple logon and logoff information. For example, when the database is started, all of your startup parameter settings are placed in the log file. This information can be used to verify the settings in your `PF` file (parameter file) or your `conmgr.properties` file.

Regularly pay attention to error log entries in the log file. OpenEdge places serious errors in this file when they occur. Most users do not report an error to the administrator unless it happens more than once. Error details are automatically recorded in the log file, providing an administrator accurate data so that appropriate corrective action can be taken.

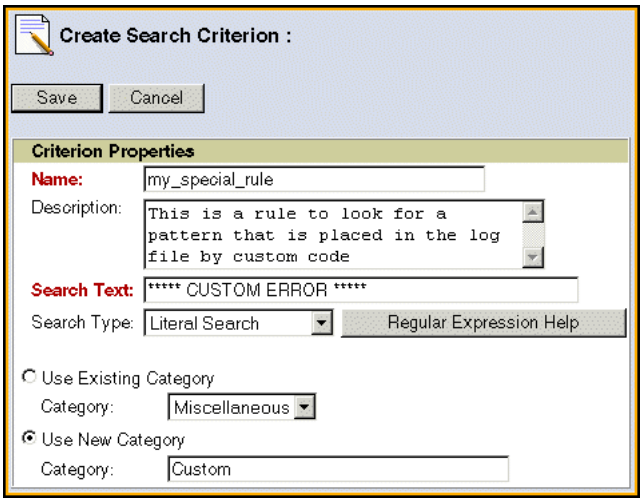
The former method for locating these errors required an administrator to scan each day, or several times a day, looking for specific words or patterns. This could be automated through the use of the `grep` command on UNIX/Linux and through the search function on Windows. Fathom can perform this task for you by searching for multiple patterns in the log file.

To create custom error search criteria in Fathom:

1. Select **Resources** from the menu bar.
2. From the list frame, browse to and select the database that has the log file you want to monitor.
3. Select **Log File Monitor**.
4. Click **LogFileMonitor_RuleSet**.
5. Click **View Library Definition**.
6. Click **Add Rule**.

7. Click **Create Criterion**.

   The **Create Search Criterion** page appears, as shown in the following example:



   The criterion created in this example allows external programs to store items in the database log file and generate alerts from within Fathom. Also, Fathom lets you create your own rule sets and run them against your own log files.

Using the PROLOG utility

The log file for a given database should be truncated on a regular basis to keep the length manageable. The `PROLOG` utility is used to keep the log file manageable. This utility should be run after you back up the log file.

Monitoring area fill

The fill rate of area extents should be checked every day. With Fathom, you can set an alert to tell you if your database is growing past a configurable limit. For example, you can tell Fathom to send you an alert when a particular area is greater than 90 percent full. This allows you to plan an outage when it is convenient for the users to address increasing the size of the area.

In addition to alerts, you might want to have a snapshot view of where the system is today. This can be done with VST code by looking at the `_AreaStatus` table. The two columns that show your area fill status are:

- `_TotBlocks`

  Represents the total space allocation for the area.

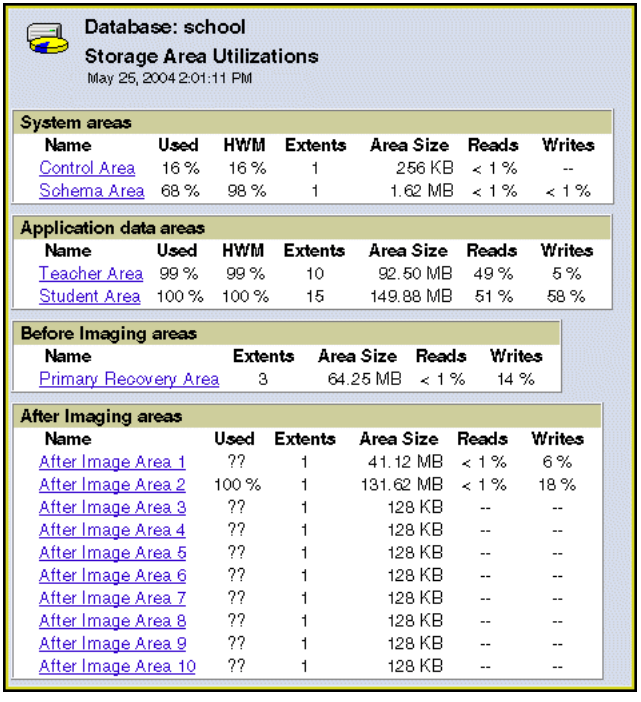
- `_HiWater`

  Represents the amount of used space within the area.

You can graphically display this data in Fathom using the **Storage Areas Utilization** page.

To monitor area fill:

1. Select **Resources** from the menu bar.
2. From list frame, browse to and select the database.
3. Select **Storage Areas** from the **Operation Views** section in the detail frame. The **Storage Area Utilizations** page appears:



Database: school
Storage Area Utilizations
May 25, 2004 2:01:11 PM

System areas

Name	Used	HWM	Extents	Area Size	Reads	Writes
Control Area	16 %	16 %	1	256 KB	< 1 %	--
Schema Area	68 %	98 %	1	1.62 MB	< 1 %	< 1 %

Application data areas

Name	Used	HWM	Extents	Area Size	Reads	Writes
Teacher Area	99 %	99 %	10	92.50 MB	49 %	5 %
Student Area	100 %	100 %	15	149.88 MB	51 %	58 %

Before Imaging areas

Name	Extents	Area Size	Reads	Writes
Primary Recovery Area	3	64.25 MB	< 1 %	14 %

After Imaging areas

Name	Used	Extents	Area Size	Reads	Writes
After Image Area 1	??	1	41.12 MB	< 1 %	6 %
After Image Area 2	100 %	1	131.62 MB	< 1 %	18 %
After Image Area 3	??	1	128 KB	--	--
After Image Area 4	??	1	128 KB	--	--
After Image Area 5	??	1	128 KB	--	--
After Image Area 6	??	1	128 KB	--	--
After Image Area 7	??	1	128 KB	--	--
After Image Area 8	??	1	128 KB	--	--
After Image Area 9	??	1	128 KB	--	--
After Image Area 10	??	1	128 KB	--	--

Monitoring buffer hit rate

The buffer hit rate measures the percentage of time the system is retrieving records from memory versus from disk. In Fathom, you can trend this rate and setup alerts for when it falls below a certain level. However, you should view the buffer hit rate throughout the day to ensure that the users are getting consistently acceptable performance. A good goal for buffer hit rate is 95 percent. As stated in Chapter 2, "Managing OpenEdge Database Resources," a buffer hit rate of 95 percent equates to five physical I/O operations for every 100 requests to the database manager. Some applications never achieve this level, but perform fine. Other applications need higher buffer hit rates to meet user requirements.

Figure 3–2 shows a portion of the Fathom **Vital Sign**s page that can be displayed from the **Operations View**. It displays content related to a test database. This example shows that the commit workload varied throughout the day but the buffer hit percentage remained constant above 98 percent. This percentage indicates consistent performance for the users.

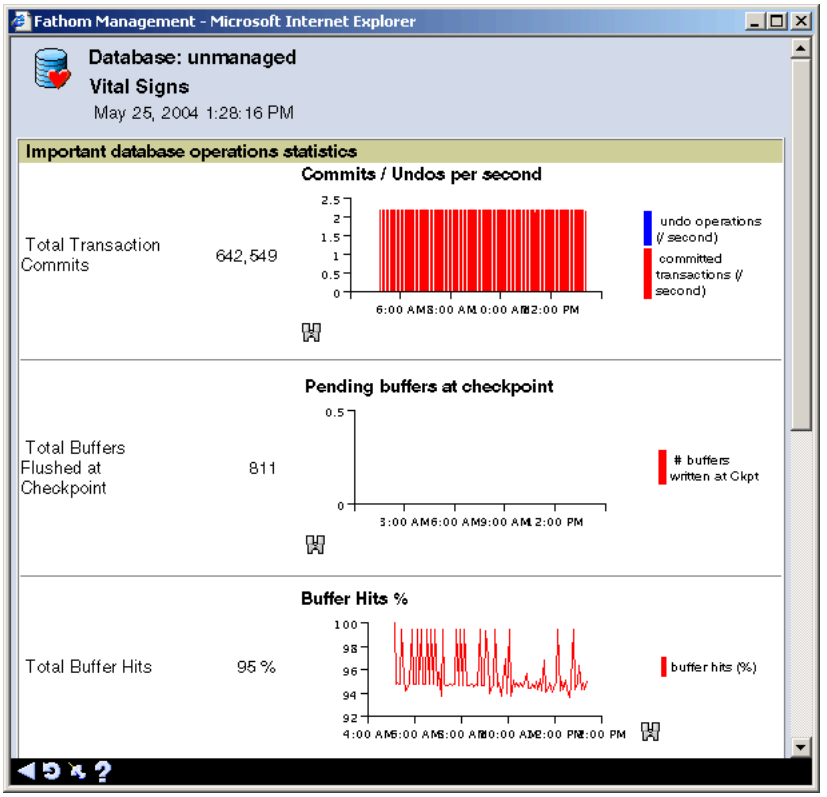


Figure 3–2: Monitoring buffer hit rate in Fathom

Monitoring buffers flushed at checkpoint

Buffers flushed at checkpoint are a good indicator of APW and checkpoint efficiency. The APW's job is to keep writable buffers at a low count. Also, you do not want frequent checkpoints where the database and memory are synchronized. If you are seeing an increase in buffers flushed during your prime operating times, and they cannot be attributed to an online backup, you should make adjustments. Solutions to this issue are discussed in the "Monitoring system resources (disks, memory, and CPU)" section on page 3–28.

If you were monitoring using `PROMON`, you would obtain the buffers flushed at checkpoint readings at the beginning and end of the prime operating hours and store them. Storing a reading can be as simple as writing it down when you read it from `PROMON`. If you create VST code, you can store the data to a file. If you are using Fathom, you can look at the information in the FathomTrendDatabase as seen in the **Operations View** in Figure 3–2.

While the `PROMON` option requires multiple checks throughout the day, the other methods let you check your APW and checkpoint efficiency once, at the end of the day. The example in the "Monitoring buffer hit rate" section on page 3–27 shows that despite high load during certain times of the day, it is possible to eliminate buffers flushed at checkpoint. This can be done with proper tuning of the before-image cluster size and the proper number of APW processes.

Monitoring system resources (disks, memory, and CPU)

The system resources that require monitoring vary from system to system, but disk fill rate — along with memory and CPU utilization — are generally resources that need to be watched daily. Memory and CPU are less important because if overused, these resources will generally only cause a performance issue. Overusing disk resources can cause an outage. The following sections present a few examples that explain how to use different utilities to monitor these system resources.

Using sar

Example 3–2 shows a `sar` output for CPU activity (`-u` option to `sar`) taken during a peak period of the day. The samples are 60 seconds in duration and repeated ten times to give a ten-minute picture of the CPU activity.

It is important to take longer samples to yield more accurate results. Many users take 5- to 10-second samples so they can see the samples more quickly, but it is very difficult, if not impossible, to tune the system down to this level. Longer samples (5 to 10 minutes) provide a good general picture.

Note: Keep in mind that samples that range from 5 to 10 minutes will miss short duration CPU spikes that you can capture with 1-minute samples.

Also, the average at the end provides you with (in this case) a 10-minute sample for overview purposes. This example shows that the amount of system CPU and wait on I/O time has been kept to a minimum, and there is still idle time. This indicates that there is additional capacity to accommodate growth in workload or the number of users on this machine.

Example 3–2: Using sar to monitor system resources

```
mymachine: sar -u 60 10

SunOS mymachine 5.7 Generic_106541-15 sun4d    07/10/02

09:55:15      percentusr       percentsys       percentwio       percentidle
09:56:15          67               5                3                24
09:57:15          68               5                5                23
09:58:15          56               3                3                39
09:59:15          54               2                2                42
10:00:15          54               2                2                43
10:01:15          53               3                3                41
10:02:15          52               3                3                41
10:03:15          53               2                2                43
10:04:15          54               2                2                42
10:05:15          65               3                4                28

Average           58               3                3                37
```

Using Task Manager

On Windows, this same CPU and memory information is available at the operating system level. It displays on the **Performance** tab in the **Windows Task Manager**, as shown in Figure 3–3. The issue with the **Windows Task Manager** is that the refresh rate for the CPU is too fast. Even on the lowest setting the screen refreshes every five seconds. This is useful for a general picture during peak periods, but it does not provide a mechanism for keeping a long-term history or good granularity of the data. It will, however, let you know when you are out of CPU resource.

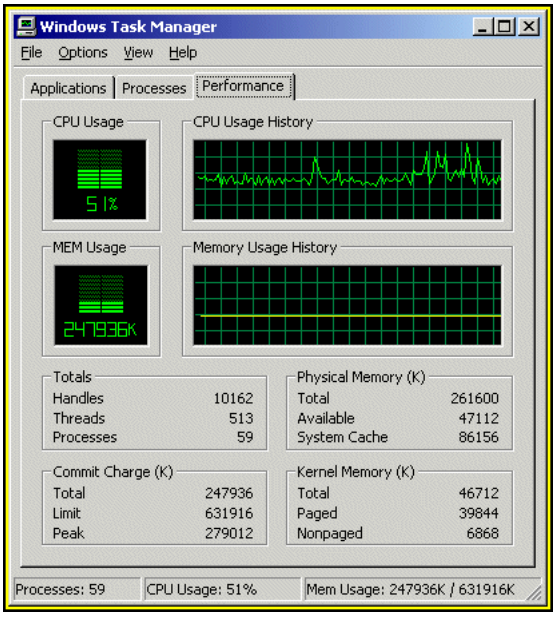
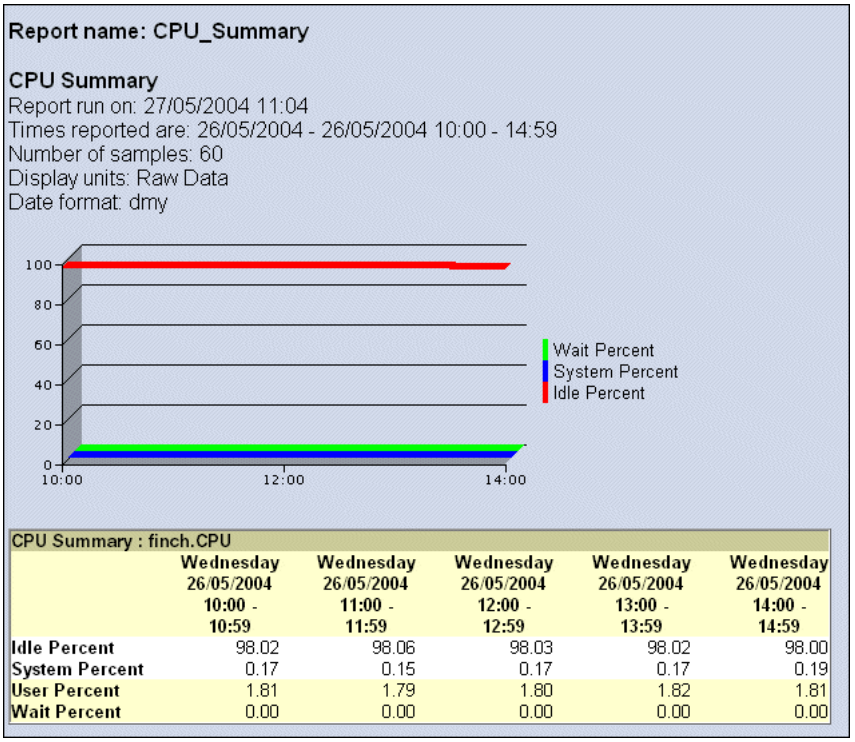


Figure 3–3: Windows Task Manager

Using Fathom

Fathom can store and report on CPU resource usage over time. There is no need to look at the system in the middle of the night while your nightly processing is running because Fathom can gather this information 24 hours a day.

Figure 3–4 shows a portion of the standard **Fathom CPU Summary** report. Fathom provides hourly averages so you can see how the system ran in your absence. Also, each sample is stored in the FathomTrendDatabase. If you need to look at each sample to perform a more in-depth analysis, you can extract the information from the FathomTrendDatabase using a custom report. Customized reports are discussed further in Chapter 4, "Guidelines for Applying Fathom."



CPU Summary : finch.CPU

	Wednesday 26/05/2004 10:00 - 10:59	Wednesday 26/05/2004 11:00 - 11:59	Wednesday 26/05/2004 12:00 - 12:59	Wednesday 26/05/2004 13:00 - 13:59	Wednesday 26/05/2004 14:00 - 14:59
Idle Percent	98.02	98.06	98.03	98.02	98.00
System Percent	0.17	0.15	0.17	0.17	0.19
User Percent	1.81	1.79	1.80	1.82	1.81
Wait Percent	0.00	0.00	0.00	0.00	0.00

Figure 3–4: CPU Summary Report

As shown in Figure 3–5, Fathom's **File System Operations view** displays:

- Storage information about all file systems.
- Database files that are on each file system.

This information makes it easy to determine disk space utilization considerations. The file system level is a better indicator than than the physical disk level. Administrators should visit this page at least once a day.

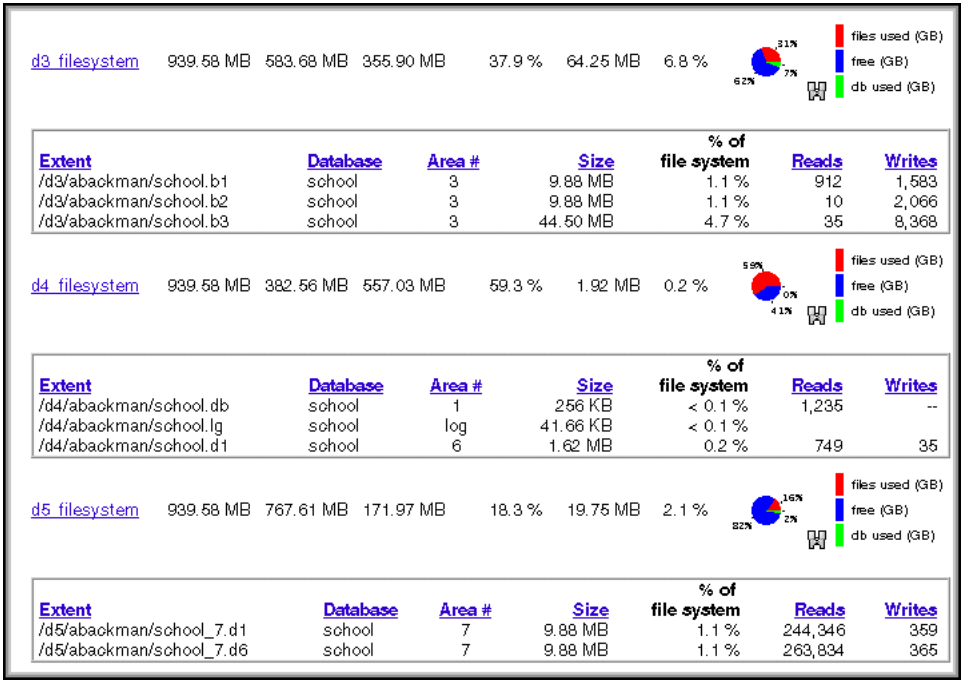


Figure 3–5: File System Operations view

Periodic monitoring tasks

There are tasks that you need to perform on a periodic basis to promote monitoring of your system. Some of these tasks, like database analysis, should be done monthly. Other tasks should only be done as needed, such as an index compress. On UNIX and Linux systems, you can schedule these tasks in `cron`. On Windows, you can use the Scheduled Tasks application.

Fathom provides a single interface to schedule all database-related tasks, including periodic maintenance to the database. In fact, you can schedule any type of task inside of the Fathom **Job Scheduler**.

The following sections describe the OpenEdge periodic maintenance utilities.

Database analysis utility

The following points relate to using the database analysis utility:

- Use the database analysis utility to expose table and index storage information.
- Run this utility at least once every three months while users are accessing the database with low-to-moderate impact on performance.
- Use the utility's information to determine if you need to rebuild your indexes or make modifications to your database structure.

The database analysis report details table storage information and helps you to determine if tables need to be moved to other areas or reorganized to reduce scatter. However, the area of index utilization is generally more dynamic and needs to be analyzed on a regular basis.

Index efficiency is important. If your data is 100 percent static, then you want your index utilization to be 100 percent to provide you with the maximum number of index entries per block. Unfortunately, this is not the case with most applications. Most applications perform substantial numbers of inserts and modifications, which impact the indexes. It is best to have sufficient space left in the index blocks to add additional key values without introducing the need for an index block split.

In an index block split, half of the information in an index block is moved to another block, and the index pointers are adjusted to include the new block. Block splits are very expensive operations. Block splits can reduce overall performance during the block split. The split might cause index fragmentation. If your indexes are small, there is no problem with a few cases of block-splitting or some fragmentation. But as indexes grow in size and activity, it is in your best interest to keep them in prime operating condition.

Look at the `_indexstat` table within the database to monitor index block splits. This VST is not enabled by default. It must first be enabled and subsequently disabled when you are finished gathering data. There is overhead associated with this feature, so it should only be enabled for the time needed to gather your statistics and make the necessary modifications to your administrative plan.

You can enable this VST using these options:

- Use the Progress Explorer to start the database, then specify index base and index limit options in the **Other Server Arguments** field on the **Default Configuration Properties** window.
- From the command line, use the `-indexbase` and `-indexlimit` options on broker startup (`PROSERVE`).

You should be most concerned with index splits. When the trend in the number of splits for a given time period increases, run index compaction to reorganize the data.

Index rebuild utility

The purpose of an index rebuild is to increase index efficiency and to correct errors. You will get significantly better organization within the indexes by sorting the indexes prior to merging them back into the database. You can do this by answering "yes" to the question "Do you have enough room for sorting?" when running the utility. Be aware that sorting requires substantial disk space (50 to 75 percent of the space of the entire database when choosing all indexes). If you have a database that is greater than 3GB, you should investigate and use the `<dbname>.srt` file to distribute these sort files.

Only use this utility when either of the following conditions exists:

- You can afford the downtime it requires.
- Index corruption forces a rebuild of an index. In other cases, you can use the index compress utility to improve the efficiency of the indexes with minimal impact to users.

Drawbacks to the index rebuild utility

The primary drawback to using this utility is that the database must be offline when the utility is run. You can use a combination of online utilities, including index fix and index compact, to approximate the effect of an index rebuild.

Another drawback is that you cannot choose a level of compression with this utility. The utility tries to make indexes as tight as possible. While high compression is good for static tables, dynamic tables tend to experience a significant number of index splits right after this utility has been run. This affects the performance of updates to the table.

Even with the decrease in update performance, the overall benefit of this utility is often desirable. The performance hit is limited in duration, and the rebuild will reduce I/O operations and decrease scatter of the indexes.

Index compact utility

This utility is the online replacement for index rebuild. Run this utility when you determine that the indexes are not as efficient as possible. You can determine inefficiencies, for example, by the output of the database analysis utility.

The benefit of this utility over index rebuild is that it can be run with only minimal performance impact while users are accessing the database. The resulting index efficiency is generally not quite as good as a sorted index rebuild, but the cost in downtime and performance impact is so much lower that it is generally the preferred option.

The index compact utility is the only option for high availability applications. Also, this utility allows you to choose the level of compression that you want for your index blocks. The default compression level for this utility is 80 percent—ideal for non-static tables. If your tables are static, you might want to increase the compression level. The limitation of the utility is that it does not let you choose a level of compression that is less than your present level of compression. In any case, your index entries will still be reorganized, but they will not have more space overall in the blocks for additional index entries.

Index fix utility

The index fix utility corrects leaf-level index corruption while the database is up and running. Run this utility if you get an error indicating an index problem.

The limitation of this utility is that if the index error is at the branch level rather than at the leaf level, you will need to use the index rebuild utility to correct the problem. The major benefit is that you can run this utility while users are accessing the database with minimal performance impact.

Table move utility

The purpose of this utility is to move a table from one storage area to another. This allows you to balance your I/O or group similar tables. You can run this utility while users are accessing the database, but users will be locked out of the table being moved for the duration of the move. This effectively prevents them from doing any work. The other drawback of this utility is that a significant amount of logging space is used. This affects the primary recovery and after-image areas. In the primary recovery area, logging of the move process uses three to four times the amount of space occupied by the table itself. If you have not planned accordingly, you run the risk of crashing the database from lack of space in the primary recovery area. This utility should be tested on a copy of your database ***before*** using it against production data.

Index move utility

The purpose of this utility is to move an index from one storage area to another. It works in the same manner as the Table move utility and has the same limitations and cautions. See the "Table move utility" section on page 3–36 for details. Note that, just as with the table move utility, you should test the index move utility on a copy of your database ***before*** using it against production data.

Running the utilities

You can run all of these utilities from the command line. You can also use Fathom to schedule these utilities to run at specific times. Fathom provides templates to run common utilities, or you can run existing script or batch programs.

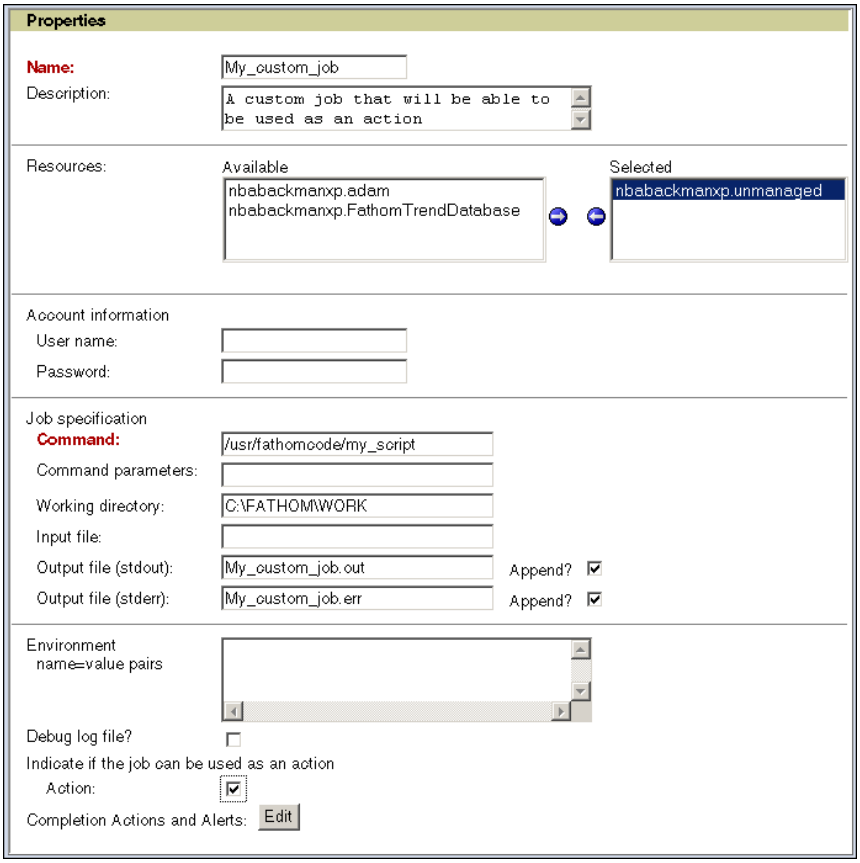
Defining a Fathom job

The steps to define a Fathom job are presented below.

To define a Fathom job:

1. Select **Jobs** from the menu bar.
2. Select **Create Job**.

3. Enter the information requested on the page:



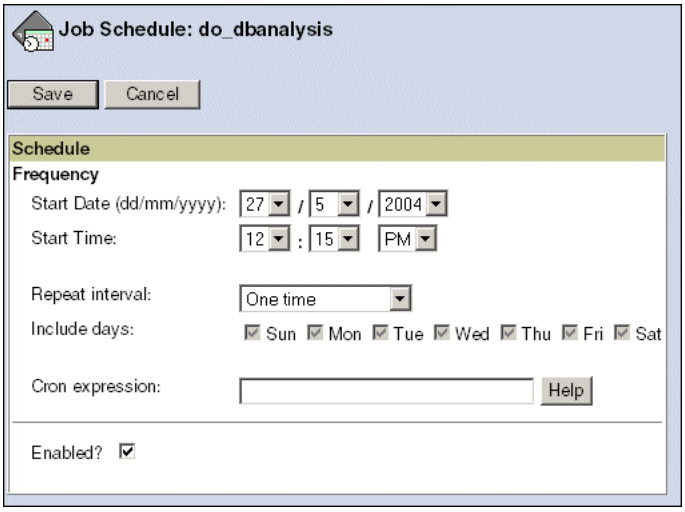
Note: In the **Command line** field, enter only the command itself with no options. Enter all options in the **Command parameters** field.

Scheduling a Fathom job

Once a job has been created and tested it can then be scheduled to run when needed. The output status of a job can trigger an alert, if necessary, so you know if a utility was completed successfully every night.

To schedule a Fathom job:

1. Select **Jobs** from menu bar.
2. Select the job you want to schedule from the list of defined jobs in the list frame.
3. Click **Schedule**. The **Job Schedule** page appears:



From this page, you can schedule the job to run once or on a regular basis.

Truncate BI and BIGROW

The before-image file or primary recovery area varies in size, depending on the transactions scoping within your application. Sometimes you will experience abnormal growth of this area, such as when you make schema changes or wholesale changes to a table in the database. These circumstances warrant truncating the BI file to recover the space.

Generally, you do not want to truncate the BI file every time you shut down the database. When the BI file is truncated for any reason, the database engine must reformat the recovery space to make it usable. If this reformatting is done while the system is running, it can cause noticeable performance degradation. Consequently, you should extend the BI file to accommodate normal growth. You can do this with the PROUTIL command's BIGROW option. With this command, you specify the number of clusters by which you want to grow the BI file. By preallocating these BI clusters and eliminating the formatting of clusters during normal processing, you can eliminate a substantial performance drag on the system.

Understanding dump and load

Determining when to perform a dump and load is a constant struggle. If you set up your database correctly, your records should not get fragmented and only your indexes should need reorganizing. This index reorganization is the primary benefit of a dump and load. Usually, about 80 to 90 percent of the benefit of a dump and load can be achieved with an index compress, as described in the "Index compact utility" section on page 3–35. However, you might need to dump and load to reorganize your database into different areas or extents due to changes in application requirements.

A fast dump and load process is important, since the system will not be available to users during the process. Some administrators have engineered change tracking to allow users access to data during the dump and load process with only a short outage to apply the changes to the target database before providing access. However, this is an application-specific process and outside the scope of this manual. This section focuses on standard OpenEdge utilities.

Before performing a dump and load, it is important to have a good backup of your existing database. This backup provides you with a fallback position should something go awry.

The basic dump and load options are described in the following sections.

Data Dictionary dump and load

Follow these rules when performing a Data Dictionary dump and load:

- Multi-thread both the dump and the load. Generally, you should add sessions on both the dump and load until you cause a bottleneck on the system.
- Pay close attention to avoiding disk variance in the database and with the dump files to achieve maximum throughput. This means using all of the disks on the system evenly. For example, you might want to make use of your `BI` disks because they will be idle during the dump portion of the process. In this instance, we are focused on I/O throughput, not storage capacity.
- Contrary to popular belief, it is better to leave the indexes enabled during reload. This does not make for efficient indexes due to index splits during the load process, but since the indexes are built at the same time the data is loaded, you can take advantage of the multi-threaded nature of OpenEdge. The indexes can be reorganized later through the use of an index compress. Otherwise, the index rebuild is a single-threaded process.

Bulk loader

This option is a good one because it is simple. However, the load process combines two single-threaded operations on the load side with the dictionary dump processes. The bulk loader itself is single-threaded; files are loaded sequentially with the indexes turned off. This option necessitates a single-threaded index rebuild process. The bulk load process itself is fairly quick. However, it is not possible to run multiple copies of this utility simultaneously, so you need to run an index rebuild at the end of the process. Therefore, it does not scale well on multi-processor systems.

Binary dump and load

This option is much faster than the previous methods described. The original implementation allowed for multi-threading of both the dump and the load. However, it did not allow for building indexes during the process and made the user do a single-threaded index rebuild. This limited the overall scalability of the process.

This limitation has been eliminated with the index build option on the binary load utility. Due to its multi-threading abilities, this utility, when used in conjunction with the index build option, provides the best overall dump and load performance in the vast majority of cases.

Profiling your system performance

Performance is a matter of perception. Users say performance is slow or fast based on many factors. It is best to take perception out of the equation by establishing some baselines on important aspects of your system.

Establishing a performance baseline

The primary reason to establish the baseline is to enable you to quantify changes in performance from changes in your load or application. The most difficult part of establishing a baseline is determining which operations are critical to the effective use of the system. You want the list of operations to be complete, but you also want to avoid having too many items on the list. Too many items will increase the amount of work needed to establish reliable baselines.

The basic rules outlined below should allow you to narrow down the number of tasks that need baselines to a manageable number. You might want to have two sets of baselines—one for daytime processing and one for end-of-day processing.

Include the following tasks:

- Tasks that are done many times throughout the day (such as creating orders, process control, and data entry tasks).
- Tasks that are important to the user (such as Web site queries, customer support screens, and order entry tasks).

Do not include the following tasks:

- Periodic tasks (such as monthly and weekly reports).
- Little-used portions of the application.
- Reporting, as it generally falls into the above two categories. Also, you can schedule most reporting outside of your primary operating hours.

Collecting your baseline statistics

Once you have determined what items you want to benchmark, you can plan your strategy.

You can modify the application code to collect this data, which is the most accurate method. However, it is also time consuming and costly. An easier way to perform data collection is to time the operations on a stopwatch. This is fairly accurate and easy to implement. To determine the best timing baseline for each task, perform timing in isolation while nothing is running on the system. When the best timing baselines have been established, repeat the task during hours of operation to establish your under-load baselines. It is best if you can train one or two individuals in each area to do these timings. This way, if there is an unrecognized performance problem, there will be someone trained to detect it and report the results so you can understand its magnitude. This also shows users that you are concerned with performance and helps to establish a rapport with them.

Understanding your results

Once your task times have been established, you need to analyze the results.

As mentioned, it is best to establish the baselines while there are no reports of any problems on the system. This will establish what is normal on your system. If users are reporting problems, you can compare the current timings against your baselines to see if the problem is real or imagined. If there is a material difference in the current timing, you need to start analyzing performance on the system with monitoring tools such as `PROMON`, VSTs, Fathom, and operating system utilities.

Performance tuning methodology

Always analyze problems starting with the slowest resource and moving to the fastest. Thus, the first place to start is disks, then memory, and finally CPU efficiency. Before you start looking at the system, you need to make sure that the application is performing correctly. Correct application performance has the greatest effect on performance. This is easy to determine by looking at the number of database requests per user. If most users have tens of thousands of requests but a few users have millions of requests, you should ask those users what they are doing with the system and look at those portions of the application for inefficiencies.

When looking at the PROMON **Block Access** screen (option 3) output in Example 3–3, you can see that most users are in the tens of thousands for numbers of requests. You should contact these users to determine how they are using the system.

Example 3–3: PROMON Block Access screen

```
Block Access:
Type Usr Name   DB Reqst  DB Read DB Write  BI Read BI Write AI Read AI Write
Acc  999 TOTAL  54844706  3828058  156560    957    12017     911      23145
Acc    0 abackman 100224     3007       7    912      869     911       4256
Acc    5               1        0       0      0    10522       0          0
Acc    6               1        0       0      0        0       0          0
Acc    7               1        0   86649      0        0       0          0
Acc    8               1        0   69823      0        0       0          0
Acc    9 DB_Agent 118388     2094       0      0        0       0          0
Acc   10 adam   49038107  3488666       2      0        0       0          0
Acc   11 lori       7829      153       0      0        0       0          0
Acc   12 john     985551   164519       1      0        0       0          0
Acc   13 connie    13886      292       0      0        0       0          0
Acc   14 michael  213915    30250       0      0        0       0          0
Acc   15 abackman      0        0       0      0        0       0          0
Acc   16 adam    4013915   130250       0      0        0       0      15982
Acc   17 frank      3416       82       0      0        0       0          0
Acc   18 jean      13358      375       0      0        0       0          0
Acc   19 wayne     83985     2063       0      0        0       0          0
Acc   20 arlene    53915     1350       0      0        0       0          0
Acc   21 carl      83742     2094       0      0        0       0          0
Acc   22 sue      114471     2863       0      0        0       0          0
```

If these users are doing the same jobs as everyone else, the cause of the problem might be how users are using the application. This is more common with off-the-shelf applications than with custom applications. The business rules for the off-the-shelf application might not match the business process exactly.

A good example of this is an application that is written for companies with multiple divisions, and the application expects the user to enter a division code as a unique identifier. However, suppose a company that has only one division bought the product and the users were not trained to enter a division code. This application might not have an index defined to find records efficiently without a supplied division code. While this can be rectified on the application side, you first need to discover the problem before you can address fixing it. The solution might be to train users to enter a division code until an application modification can be made.

Once you have ruled out the application, you can start looking for common problems.

Advantages and disadvantages of monitoring tools

The tools needed to examine performance issues range from cryptic (`iostat`) to full-featured and easy-to-use (Fathom). This section describes your tools options, and the benefits and drawbacks of each option.

Table 3–5 shows the advantages and disadvantages of monitoring tools such as `TOP, IOSTAT, VMSTAT, SAR,` and `MONITOR`.

Table 3–5: Advantages and disadvantages of system tools

Advantages	Disadvantages
• Free. • Instant access to information. • Standard.	• Displays only OS information. • Some provide graphical information only without a command-line interface. • With the exception of `sar`, there is little or no history stored unless the user writes custom code.

Table 3–6 shows the advantages and disadvantages of the OpenEdge `PROMON` utility.

Table 3–6: Advantages and disadvantages of PROMON

Advantages	Disadvantages
• Included with OpenEdge server license. • Good current view data.	• Menu interface only. There is no command line. • Difficult to capture and store history for trending. • Database information only. • No comprehensive storage area information.

Table 3–7 shows the advantages and disadvantages of monitoring with OpenEdge VSTs.

Table 3–7: Advantages and disadvantages of VSTs

Advantages	Disadvantages
• Flexible. • Comprehensive database information available.	• High learning curve for interpreting the information. • Requires knowledge of the Progress 4GL. • No operating system performance statistics. • Custom code for all aspects, including storage of trending data.

Table 3–8 shows the advantages and disadvantages of monitoring with Fathom Management.

Table 3–8: Monitoring with Fathom Management

Advantages	Disadvantages
• Easy-to-use interface. • Common tool for both operating system and OpenEdge data. • Simple storage of trending data. • Remote administration capability.	• Databases must be in or migrated to the `conmgr.properties` file for all functions to work. • Limited customization ability. • Additional licensing cost.

Common performance problems

The common performance problems include:

- Disk bottlenecks.
- Memory bottlenecks.
- CPU bottlenecks, including performance issues such as a runaway process.
- Issues such as `BI` cluster size and page writers.

Disk bottleneck causes and solutions

Disks are the slowest resource on a host base system, so it is important to resolve issues in this area first before proceeding to other areas of concern. Generally, when people report performance issues, this is the first place to look for the cause of the problem.

Causes

The most likely causes of disk-related performance problems are listed below:

- Disk variance.
- Application issues.
- Low database buffer hit rate.

Disk variance

All disks are capable of doing approximately 100 I/O operations per second. If one disk is doing all of the work, then the maximum number of I/O operations possible is around 100. Additional disks functioning at the same time will increase the throughput potential of the system and increase performance to the users.

Disk variance can be seen though `sar`. Example 3–4 shows two issues:

- Two disks are doing significantly more work that the other disks.
- The amount of wait time is greater for those disks that are working harder. In this case you could move items off of the heavily used disks and onto the more lightly used disks.

There are many cases where there is extra storage capacity available on the disks, but due to the utilization of the disks for I/O operations, it is advisable to add disks to increase throughput capacity.

Example 3–4: Viewing disk variance using sar

```
16:59:58   device
percentbusy   avque   r+w/s   blks/s   avwait   avserv

17:00:58   nfs2          0     0.0       0        0       0.0        0.0
           sd0           0     0.0       0        5       0.0       23.3
           ssd2          0     0.0       2       29       0.0        1.8
           ssd5          0     0.0       0        1       0.0        4.8
           ssd6          0     0.0       0        3       0.0       12.1
           ssd7          0     0.0       0        1       0.0       12.6
           ssd10,g       0     0.0       0        2       0.0       10.8
           ssd11,g       1     0.0       1       29       0.0       13.8
           ssd12        99     3.6       2      307     128.5      145.8
           ssd15         2     0.0       1       23       0.0       16.0
           ssd16         1     0.0       1        9       0.0       13.7
           ssd17         3     0.0       3       47       0.0       12.0
           ssd20         4     0.0       3       60       0.0       14.6
           ssd21        99     2.1       5      239     107.2      116.5
           ssd26         1     0.0       5       71       0.0        1.7
```

Application issues

A poorly written application can cause a significant drain on the system. When tracking down performance problems, you should always consider the application before you look at the hardware. Prioritize the portions of the application to be investigated and corrected before modifying startup parameters. Changes to startup parameters can improve performance without any application modifications, but the effects of these changes are not as durable as application changes.

In terms of application procedures, it is important to avoid running heavy reports during the hours of heavy On-Line Transaction Processing (OLTP). Running reports during this period can have a significant impact on performance for those users. This is because reports generally look at historical data and OLTP users look at current data. When a report is run, the old data needed by the report replaces the new data required by the OLTP users. When that new data is needed again by the OLTP users, they must retrieve it from disk rather than from memory. While a report is running, it can force the new data to be flushed from memory once again, and this vicious cycle can continue as long as the report is running.

Low database buffer hit rate

As noted previously, it is important to achieve the maximum buffer hit rate possible without causing other problems on the system (such as excessive paging or swapping). If the buffer hit rate is too low, you will be forcing users to access the disk rather than memory for information. Since memory is an order of magnitude faster than disk, the objective is to retrieve as much data as possible from memory and as little as possible from disk.

Solutions

Some of the solutions to the above problems include the following:

- Balancing I/O across available disks.
- Using –Bp to reduce impact of reporting.
- Increasing database buffers.
- Increasing throughput capacity or redistributing I/O load.

Balancing I/O across available disks

If you have many disks on your system, be sure to spread your database and application files across as many physical disks as possible. Remember that application files are generally read only once, so it is more important to spread your database across the maximum number of disks to achieve maximum throughput. You need to monitor the number of I/O operations per second on the disks to determine if you are approaching the throughput threshold for the disk.

Using –Bp to reduce impact of reporting

Private buffers allow users to have a personal or private buffer pool within the database buffer pool. The `-Bp` client startup option allows you to allocate some of the general buffer pool for private use. It is possible to allocate up to 25 percent of the general buffer pool in private buffers.

The private buffer option helps reduce the impact of reporting on other database users by using and reusing the same portion of memory over and over again. Each user of private buffers keeps a list of data in these buffers. Instead of looking for space in the general buffer pool, which would affect other users, they look in their own list of buffers for the least-recently-used buffer and overwrite that buffer. This option is only effective for report or read-only users since only unmodified buffers can reside in a private buffer pool list. If a modification is made to the buffer, then that buffer is taken off the private buffer list and added to the general buffer pool list. These lists are known as least-recently-used (LRU) chains because the least-recently-used item is taken off the top of the list and reused first.

Increasing database buffers

By increasing the amount of memory available for database buffers, you can put a larger percentage of your database into memory. The theory behind this is that you actively use only a small percentage of your database, and if you can put this portion of your database into memory, you can avoid doing disk I/O and increase performance.

However, there is a point of diminishing returns on increasing buffers. Once you get a high buffer hit rate, it takes a large increase in buffers to further increase the buffer hit percentage. Remember, those small percentage increases can have a fairly large effect on performance. Moving from a 95 percent to a 96 percent buffer hit rate represents a 20 percent decrease in disk read operations, as the number of physical reads per 100 requests will be reduced from 5 to 4. This is a consideration when looking at the reasons for increasing buffers and the benefit in performance to users.

Another item to consider is whether there is a more effective use of the memory, such as client memory.

Increasing throughput capacity or redistributing I/O load

It is possible that you do not have enough disk throughput capacity on the system. If you have balanced the disk I/O across the available disks, increased the database buffers to get optimal efficiency, and tried offloading tasks to other periods of the day, and you still have a disk bottleneck, your only recourse might be to purchase additional disk drives. Remember that it is better to buy several small disks than one large disk even though they might have the same storage capacity. The smaller disks will have a greater combined throughput capacity.

Memory bottleneck causes and solutions

Although disk bottlenecks are likely to have the biggest performance impact because they are the slowest part of the system, a memory bottleneck in the wrong place can also cause an immense amount of disk activity. If your disk problems persist after you have done everything possible to correct them, investigate your memory resources and settings.

Causes

The possible causes of memory bottlenecks are:

- Improper allocation of memory resources.
- Operating system using more memory than necessary.
- Other applications.

Improper allocation of memory resources

If you have unused databases running on your system, they are consuming memory you could allocate elsewhere. You might have databases with high user memory consumption, causing excessive physical paging and swapping on the system. Perhaps high memory allocations are excessive and could be reallocated to other resources.

Operating system using more memory than necessary

Some operating systems will allocate excessive amounts of memory if left unchecked. With a few modifications to the kernel, you can limit your exposure to this phenomenon.

Other applications

On many systems OpenEdge is not the only application using resources. Consequently, it is important to analyze all the applications on the system and balance the resources of one application against another. There are many cases where an application is initially very small, so its resource usage is not noticed. However, over time the usage and resource footprint has grown and is adversely impacting the entire environment. Consequently, all applications on a system should be monitored for their memory consumption.

Solutions

Possible solutions to application problems include:

- Use memory for the common good.
- Limit operating system buffers.
- Think outside the box.

Use memory for the common good

Memory should be used first for items that will benefit all users. If there is memory still available, then allocate it to individual users or processes. Generally, you want to look at the buffer hit rate for all of your databases and get the best performance you can before investigating other issues and solutions. There are exceptions to the rule in the individual process area, notably background tasks that are used to do reporting and other common tasks.

Limit operating system buffers

Some operating systems will dynamically allocate memory to accommodate as many modified buffers as possible. However, it is a good idea to limit operating system buffers to 10 percent of physical memory. OpenEdge handles most of its own I/O and does so even more with the use of the `-directio` parameter on broker startup.

Think outside the box

Using memory is not always a choice between different operating systems and OpenEdge parameters. One example of using resources in a nontraditional way is the use of RAM disks. If you do a significant amount of I/O to the temporary files on your system, you might want to create a RAM disk with some of your memory to accommodate these files and eliminate some I/O to the disk subsystem. You must be very careful about how and when to use this option, and you must have a very deep understanding of how much space you are using with temporary files. You do not want to cause a reliability problem with the adoption of this feature.

CPU bottleneck causes and solutions

Once you encounter a CPU bottleneck with no other bottlenecks on the system, your only recourse is to buy more CPU capacity. This is easier said than done. CPU time is divided into user, system, wait, and idle time. Your goal is to achieve the maximum amount of user time as possible. A practical system profile is 70 percent user, 20 percent system, 0 percent wait, and 10 percent idle. The most common problem is that the CPU is waiting for another resource, in most cases the disk, and all of the CPU time is being used waiting for this resource. On most systems, this is very easy to see because the percentage of waiting on I/O will increase. However, on some systems this time is logged as idle time. When the time is logged as idle rather then waiting on I/O, you need to monitor the disks to see if the percentage of idle time increases in direct opposition to performance on the system.

Common causes for CPU issues include the following:

- Runaway processes.
- Improper setting of the OpenEdge –spin parameter.

Runaway processes

It is possible for a single process to use 100 percent of the CPU time on a single-CPU system. In many cases all of the time is logged as user time on the CPU. Although this looks good on paper, it does not tell the real story.

By focusing on the amount of time a single process uses on the system, you can see processes that exceed a particular threshold and report those processes to the administrator. By monitoring your system over time, you can determine both the average and the maximum amount of time a process uses on the system. Processes that use significantly more CPU time need to be investigated by the administrator. Note that in some cases high CPU time is justified for intensive portions of the application, but it is easy to spot a pattern and weed out the “good” from the “bad.” This is easier to do on UNIX-based systems than on Windows systems. Fortunately, the problem is much less prevalent on Windows than it is on UNIX. This is because Windows is a closed environment and therefore has fewer ways to do things in an out-of-the-ordinary manner than do UNIX-based systems.

Improper setting of the OpenEdge –spin parameter

On systems that have multiple CPUs, OpenEdge provides a mechanism that allows users to do multiple simultaneous operations. The problem occurs when these operations “step on each other’s toes.”

OpenEdge implemented latches, which are very fine grained locks in shared memory, to address this issue. Before a user process can change a memory structure, it must establish a latch to ensure the resource is not in use by another user. That way, no other user will modify the resource while the first user is making a change. When establishing the latch, the CPUs check with each other to ensure that two processes (users) are not trying to do the same thing at the exact same time. Once that check has been completed, the first user process gets the latch, makes the change in shared memory, and releases the latch.

All of this occurs at memory and CPU speeds, so thousands of operations can be completed in a second. The problem manifests itself when the first user has established the latch and a second, third, or fourth user tries to manipulate the same resource. By default, each user asks for the resource one time and relinquishes the CPU if the resource is not available. This is very inefficient, since a significant amount of system overhead is used to initially render the process active in the first place. So, instead of asking once for the latch, it is better for the process to ask many times in an effort to get the latch. It is cheaper in terms of system resources to ask thousands of times, and in some cases tens or hundreds of thousands of times, to get the latch. Because of the relative inefficiency of the CPU queue, multiple requests are preferable to asking once, getting refused, and going to the end of the CPU queue only to come to the top of the queue to ask again and finally get the resource. The problem manifests itself as high system time on your CPU monitor.

The OpenEdge `-spin` parameter is set to indicate the number of requests to make for the latch before going to the end of the CPU queue. For most multi-CPU systems, a good starting point is a setting of 2,000 for `–spin`. Settings between 2,000 and 10,000 work well in the majority of cases.

Other performance considerations

Other performance areas to consider include the following:

- BI cluster size
- Page writers
- Database block size
- Procedure libraries

BI cluster size

The `BI` cluster size determines the frequency of checkpoints on the system. The larger the cluster size, the longer the time frame before the database engine will attempt to reuse the clusters. A reuse scenario requires a checkpoint. Your goal is to have checkpoints occur on an infrequent basis. A scenario of one checkpoint every two minutes during the highest update times of the day is still too high. On many systems, a fairly low cluster size of 1024KB (1MB) will yield checkpoints every 30 minutes for most of the day, with some higher frequency during peak periods. On most systems, 1024KB should be the lowest setting to consider. If your workload warrants, you should increase it to 2048KB and then monitor checkpoint frequency. If it is still too high, increase it to 4096KB, monitor again, and so on. Most low-to-mid-update systems fall into the 1024KB-to-4096KB range. Increase your `BI` cluster size only if you are on an enterprise version of OpenEdge that will allow you to implement page writers.

Page writers

The enterprise version of OpenEdge provides a mechanism to expedite the task of writing database buffers to the disk. There are several kinds of page writers: before-image, after-image, and asynchronous. The before-image writer and after-image writer are easy to understand and set. If you have a read-write database, you should have a before-image writer. If you have after-imaging enabled, you should have an after-image writer. You can only start one of each per database. The more complex scenario is the use of asynchronous page writers (APWs). These synchronize the modified buffers in the database buffer pool in shared memory with the database files on disk. If you are doing updates to the system, you should start with one APW and then monitor the buffers flushed at checkpoint.

You can monitor the buffers flushed at checkpoint using the PROMON **Activity** (option 5) screen, as shown in Example 3–5.

Example 3–5: PROMON Activity screen

```
Activity  - Sampled at 07/10/02 17:50 for 30:44:17.

Event               Total  Per Sec    Event               Total   Per Sec
         Commits      853      0.0              Undos          3      0.0
  Record Updates   107290      0.9       Record Reads  259518607   2345.2
  Record Creates   291993      2.6      Record Deletes         1      0.0
       DB Writes   158701      1.4           DB Reads    8213547     74.2
       BI Writes    12092      0.1           BI Reads        967      0.0
       AI Writes    23289      0.2
    Record Locks 181966214   1644.4       Record Waits       217      0.0
     Checkpoints        11      0.0     Buffers Flushed        0      0.0

Rec Lock Waits 0 percent  BI Buf Waits  0  percent  AI Buf Waits 0 percent
Writes by APW 99 percent  Writes by BIW 88 percent  Writes by AIW 0 percent
Buffer Hits       7  percent
DB Size         308 MB       BI Size      64 MB       AI Size     182 MB
FR chain        660 blocks   RM chain     10 blocks
Shared Memory 83332 K        Segments      1

0 Servers, 5 Users (5 Local, 0 Remote, 0 Batch),2 Apws
```

You can also use Fathom to monitor buffers flushed. Figure 3–6 shows the **Checkpoint summary** section of the **Page Writers Operations** view page.

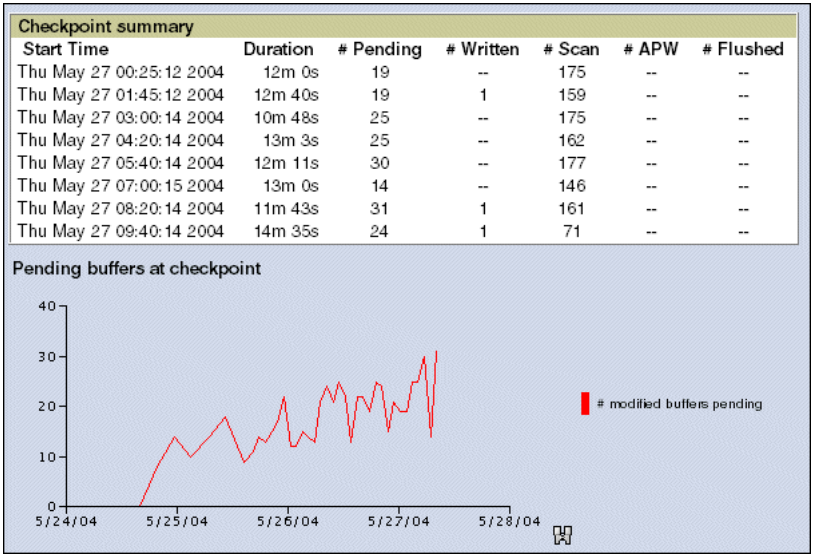


Figure 3–6: Viewing buffers flushed in Fathom

If you see "buffers flushed at checkpoint" increasing during the performance-sensitive portion of the day, and this increase is not attributable to a quiet point or an online backup, you should add one APW process. If you keep increasing the number of APWs and you do not see a corresponding decrease in buffers flushed at checkpoint, then most likely you have a disk bottleneck.

Database block size

As stated previously, in most situations an 8KB database block size is the optimal setting for most operating systems other than Windows. For Windows systems, a setting of 4KB is more appropriate. By synchronizing the database block size with the operating system block size, you will get better performance. When you read a single database or index block on an 8KB block size database, you will get eight times the information you would get for a 1KB block size. Also, if the operating system handles disk to memory transfers in 8KB chunks, as most operating systems do, and you only ask for a 1KB block, you run a high risk of wasting 7KB of transfer space containing information that you will never use.

Procedure libraries

OpenEdge procedure libraries let you eliminate unnecessary disk I/O. Normally, when a program is read from disk to be executed, a copy of it is placed in the local sort (SRT) file in case it needs to be retrieved. This copy process causes a moderate-to-high amount of disk activity. If a program is stored in a procedure library, the client process does not store a copy of the program in the sort file. (You can override this with the -pls parameter.) For host-based systems this can represent a significant reduction in temporary file I/O, which is very likely to increase performance.

Conclusion

Performance tuning is more art than science in many cases. It is necessary to view the system as a whole and realize that an improvement in one area might cause a problem in another. The overall goal is to work the bottleneck to the fastest resource, which is the CPU. Once you have achieved that, you can rationalize additional hardware expenditures to scale your application even higher. It is always important to look at the application itself to make sure that you are not just "throwing hardware at the problem." Once you have an efficient application to work with, you can take advantage of the system to run the maximum number of users at the highest performance level with the fewest resources possible.

Periodic event administration

There are some tasks that are performed only occasionally on the system. These tasks require thought and advance planning to decrease their impact and maximize the benefit to the organization. Many of the items listed in this area could be considered luxury items, but if you are well positioned in the other aspects of your system, you can invest some time in these tasks to round out the management and maintenance of your system.

Annual backups

The annual backup is generally viewed as a full backup of the system that can be restored in the event of an emergency. The most common use of the annual backup is for auditing purposes. These audits can occur several years after the backup is taken, so it is very important to be able to restore the system to its condition at the time of that backup. In the United States, it is possible for the Internal Revenue Service to audit your company as far back as seven years. How likely is it that you will be on the same hardware seven years from now? You might be on compatible hardware if you are lucky; most likely you will be on different hardware with a different operating system. Consequently, it is important to plan thoroughly for such an eventuality.

One way to guarantee platform independence is to dump your important data to ASCII and back it up on a reliable, common, and durable backup medium. Some people prefer optical storage over tapes for these reasons. Also, don't overlook the application code and supporting software, such as the release of OpenEdge being used at the time of backup. If you are not going to dump to ASCII, you must obtain a complete image of the system. If you take a complete image and are audited, you will need to find compatible hardware to do the restoration. It is also important to use relative path names on the backup to give you greater flexibility during the restoration. Finally, you need to document the backup as thoroughly as possible and include that information with the media when sending the backup to your archive site.

Archiving

A good IT shop always has a complete archiving strategy. It is generally not necessary to keep transactional data available online for long periods of time. In most cases, a 13-month rolling history is all that is necessary. This can and will change from application to application and from company to company. You need a thorough understanding of the application and business rules before making a decision concerning when to archive and how much data to archive. In most cases, you should keep old data available offline in case it is needed. In these scenarios, you should develop a dump-and-purge procedure to export the data to ASCII. This format is always the most transportable in case you change environments or want to load some of the data into another application such as Microsoft Excel. Always make sure you have a restorable version of the data before you purge it from the database. An archive and purge can dramatically improve performance, since the system will have far fewer records to scan when it is searching the tables.

Application modifications

Changes to applications require careful planning to reduce interruptions to users. Although there might be a process to test application changes at your site, database administrators should consider it there responsibility to verify expected application changes. The most effective way to do this testing is to have a test copy of your database that is an exact image of what you have in production and a thorough test plan that involves user participation.

Making schema changes

Schema changes can take hours to apply if they are not done properly. If the developers tested the application of schema changes against a small database, they might not have noticed a potential problem. A small database could apply an inefficient schema update in a short period of time and would not raise any red flags. If you have a full-size test environment, you can apply the schema change and know approximately how long it will take to complete. It is important to understand how long this process takes, since the users of the application will be locked out of the system during the schema update.

You should apply schema changes in single-user mode to avoid potential problems with schema locks. If you do not have a full copy of production in your test environment, you can apply the changes in multi-user mode and watch the number of updates that are being done by the process. If you see hundreds or thousands of requests being done, you can estimate the amount of time required based on the relative size of the test database to the production database. This estimate will not be 100 percent accurate, but it will give you a general idea as to how long you need to schedule the outage for this operation.

As Figure 3–7 illustrates, you can use Fathom to determine the number of database requests a user has performed. You need to look at the number of requests before and after doing the schema change operation. This information is also available through the `PROMON` **Block Access** screen (option 1).

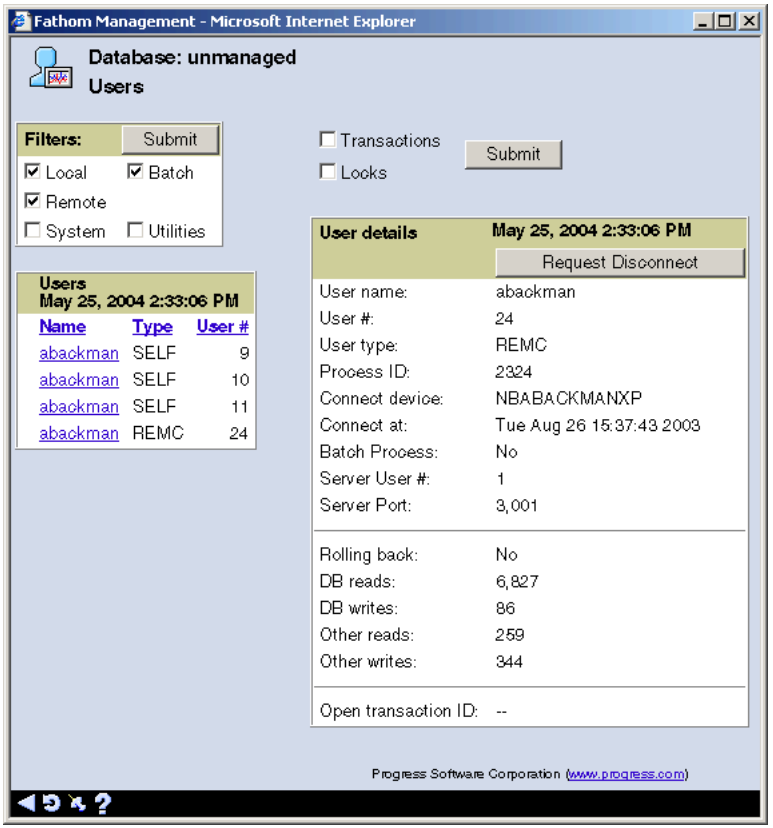


Figure 3–7: Monitoring user requests in Fathom

It is difficult to determine which user number you are. You can log in with a unique ID, figure it out though a process of elimination, or run the following program from within the OpenEdge editor to determine your user number:

```
FIND _MyConnection.
DISPLAY _MyConn-UserId.
```

Making application code changes

The amount of time it takes to apply application code changes can be greatly reduced by an advance compilation of your code against a CRC-compatible copy of your production database. To maintain CRC compatibility, start by creating a basic database, which is one that contains no data—only schema definitions. Use the basic database to seed a production database and a development database. The basic database is also saved, so you will have three copies of your database. If you already have a production database in place, the basic database is obtained by dumping the schema from those databases.

As development occurs on a test copy of the database, the production and basic databases remain unmodified. When you are ready to promote your schema changes from development to production, first make an incremental data definition dump from the Data Dictionary by comparing the development schema with the basic database. The incremental data definitions can be applied to the basic database, and you can compile your application against that database. Second, the incremental data definitions can be applied at a convenient time on the production database (after appropriate testing). While the incremental data definitions are being applied, you can move the r-code you created against the basic database into place avoiding additional downtime to compile the application code.

Migration of OpenEdge releases

Migrating releases of OpenEdge can be as easy as running a conversion utility or as complex as a dump and load of your databases. And, in most cases, minor version changes can be made without running a conversion utility. It is important to test even minor version changes and service packs or patches in your test environment prior to promoting the code to production. When making a major version change, you need to do additional analysis prior to making any changes. Major version changes also require that you test the conversion process for performance and reliability prior to applying the new version to your production environment. In almost all cases, even major version upgrades go very smoothly, but you never want to become complacent.

When you have everything in your database set up properly, you are ready to run the conversion utility. Generally, the actual conversion of the database will only take a few minutes, so it is not a major undertaking to convert the test environment and verify that conversion. After the verification has been done on the test database, you can decide how to proceed with the production application and databases. If you are unhappy with your setup for any reason, it might be wise to do a more complex conversion.

One possible motivation for considering a complex conversion might be that you have a significant amount of record fragmentation in the database. A complex conversion might start with a simple conversion to take advantage of new features in the new version. Then you would do a dump and reload of your tables to establish a new database structure. By using a multi-phase approach, you can minimize your risk by providing a fall back position if there are problems during the migration. It is imperative to have good, tested backups before applying a new version of OpenEdge or application code to the system. The same is true even when applying minor versions and patches to the system.

Summary

The ideal system plan accounts for all aspects of the system administration and maintenance, even those items done on an irregular or occasional basis. If you plan and test properly, you can avoid potential problems and provide a predictable environment to users. In general, people do not like surprises. This is especially true with business systems, so establishing accurate schedules for all tasks is very important and will build confidence in your system.

4

Guidelines for Applying Fathom

This chapter provides guidelines for installing, configuring, and using Fathom, including the following topics:

- Making practical resource monitoring decisions
- Configuring Fathom for your environment
- Remote monitoring
- Performance considerations
- Configuration Advisor
- The File Monitor
- Creating custom reports using 4GL
- Creating custom jobs
- Extending usefulness of existing Fathom functions
- Troubleshooting your Fathom installation

For more information about the tasks discussed in this chapter, refer to the Fathom product documentation where you will find comprehensive installation, configuration, and usage instructions. The Fathom product documentation set includes the following manuals:

- *Welcome to Progress Fathom Management Standard Edition*
- *Installation and Configuration Guide*
- *Resource Monitoring Guide*
- *Database Management Guide*
- *Alerts Guide and Reference*
- *OpenEdge Server Management Guide*
- *Reporting Guide*
- *FathomTrendDatabase Guide and Reference*
- *OpenEdge Revealed: Achieving Server Control with Fathom Management*

Fathom customers receive this documentation with the Fathom product. You can also access this documentation from the Progress Software Corporation Web site at:
`http://www.progress.com/products/documentation`

Introduction

Throughout this manual you have been introduced to Fathom as a tool to monitor and trend computer, operating system, and OpenEdge resources. The benefits of the product include:

- A common interface to monitor and view resources across operating systems and platforms.
- Automatic trending of resources upon installation.
- Alerts to identify potential trouble spots.
- Ability to automatically configure alert thresholds.

While it is easy to see the value in such a product, you also need to be aware that a certain amount of effort is needed to set up and maintain Fathom so that it generates the best results.

Fathom can provide a snapshot of the current status of a resource, or a trend of resource utilization over a period of time. This trending information is stored in an OpenEdge database, which is called the FathomTrendDatabase. This database needs to be maintained like any other database on your system. Backups, tuning, and other maintenance activities are required to run the FathomTrendDatabase properly, and to preserve and protect your data.

Making practical resource monitoring decisions

During the Fathom installation process, you need to make decisions regarding the resources that you want to monitor and trend. You also need to think about the frequency of monitoring. If you monitor too often or have too many monitored resources, you run the risk of affecting performance of your core systems. If you monitor too few resources or monitor the resources too infrequently, you reduce the effectiveness of Fathom.

This section provides guidelines for making effective decisions during the Fathom installation process.

There are two portions of the Fathom installation:

- Installing the Fathom software.
- Configuring the Fathom resource monitors.

Before you install

Before you install Fathom, you need to do the following:

- **Apply necessary service packs** — Before you install Fathom, you must install the service pack from the CDs provided with the software.
- **Decide on what machine you want to store the FathomTrendDatabase** — Install Fathom first on the machine where you are going to store your *FathomTrendDatabase*. Then install Fathom on other network nodes. Your application and network layout will determine the number and location of FathomTrendDatabases, as described in the "Determining the location and number of FathomTrendDatabases" section on page 4–18.
- **Decide on the location where you want to install the FathomTrend Database** — While the default location for the FathomTrendDatabase is under the Fathom installation, it is advisable to specify a directory outside of Fathom and OpenEdge (that is, `$DLC`). This could make it more accessible for standard maintenance. Also, if you uninstall Fathom, the database will be deleted if it resides within the Fathom installation.
- **Know the SMTP host name and port number —** Prior to installing Fathom, you must know the Simple Mail Transfer Protocol (SMTP) host and port number. The SMTP host is your mail server, and the port is the port number that this service is using to provide its service to your network. The port is 25 by default, but some administrators modify this for security reasons. Your mail administrator can provide this information to you in most cases, or if you can log into the mail server machine, you can look at the services file to determine the port that SMTP uses. The services file is located in the `/etc` directory on UNIX and Linux systems. On Windows systems, this file is located in the `\winnt\system32\drivers\etc` directory, by default.

In this file you will see a line similar to the following:

```
smtp    25/tcp    # mail
```

Where:

- `smtp` is the service name.
- `25` is the port number used by the service.
- `tcp` is the network transfer protocol used by the service.
- `# mail` is a comment, which is used to help the administrator identify the service.

The only item that might change on this line is the port number. This is the value that you need when you install Fathom.

Initial installation settings

Allow Fathom to auto-discover all of the resources on the **Getting Started** page. Default trending for resources will be enabled automatically. This will give you a good basic installation to work from and allow you to choose which resources you want to trend. If you have experience with Fathom, you might know which resources to trend, and the installation will provide a convenient method to disable trending on just those resources you specify. For the first-time user, it is best to use the defaults until you know what resources you want to trend. You will still be able to get snapshot information of resources even if you decide to disable trending at a later time.

Post installation configuration tasks

Once your installation is complete, you need to perform some additional tasks to complete the configuration. You might make future modifications to these items as you analyze the data.

In brief, here are the major steps to complete your Fathom installation:

1. Create one or more monitoring plans.
2. Create jobs.
3. Create actions.
4. Create rules.
5. Export your settings.

Create monitoring plans

All Fathom resources must have a monitoring plan. Each monitoring plan includes a schedule.

The first order of business is to change Fathom's default schedule, as this is the basis for trending on your system. The default schedule is initially set to monitor 24 hours a day, every day. This schedule can be modified, but not deleted or renamed.

To provide additional flexibility in scheduling, additional schedules can be built during the initial installation. These schedules are used only if requested by the user. Additional schedules can be modified to better suit your needs. The most likely modification is to have your default schedule cover your prime business hours (for example, Monday through Friday 8AM to 6PM), with a second schedule for after hours and perhaps a third schedule for weekends. This approach allows you to have more aggressive monitoring during peak periods and less monitoring during times of inactivity. If no one is on the system and there is little or no batch activity on the system between 7PM and 8AM, you could leave monitoring off until 8:30AM. This would allow users to fill the buffer pool during their normal ramp-up time, which is not indicative of "normal" running operation of the system.

There are many systems that do significant amounts of work during nightly processing. If this is the case at your site, it is wise to leave monitoring and trending enabled during this period. The point is to have Fathom poll and trend only when necessary.

Create jobs

After schedules and actions have been created, you can create jobs for your system. Each job is individually scheduled on the **Job** page in Fathom. A job can be run once, repeated on a schedule, or used as an *action*. You can use this process to start your end-of-day processing at the same time each night, your weekly processing on the same day and time each week, or to repeat a process on a monthly basis.

When you define a job, you can also use the job as an action. Any jobs defined as actions can be used within compound actions, as described in the "Create actions" section on page 4–7.

Create actions

After you complete your schedules, you want to create actions or modify the default action. The actions provide a mechanism for doing something in response to an alert. Alerts are discussed in the "Rules" section on page 4–12.

Fathom provides the following actions:

- **E-mail Action** — This action is used to send an e-mail message in response to an alert. Many pager companies allow an e-mail message to be sent directly to a pager, so you can use this action to notify a person of an urgent issue. You might want to set up several e-mail actions, depending on the severity of the problem. One action could be used to notify administrators only, while another could inform the help desk, and a third could notify a distribution list about very urgent issues, and so on. Information e-mails can be sent to a location that is checked only periodically to minimize the interruptions for low-level messages.

- **Log Action** — This action allows users to place a message in the file of their choice. This is another way of handling informational messages. Instead of sending an e-mail to a user or account that is looked at infrequently, you can place the message in a file that is viewed on an ongoing basis. This way, the volume of e-mail messages is minimized.

- **SNMP Trap Action** — This action generates an SNMP trap or message. Simple Network Management Protocol (SNMP) is the Internet standard protocol for managing nodes on an IP network. For example, if you are running an operating system tool that supports SNMP to manage your system such as HP OpenView, you can configure Fathom to send traps to your tool to inform you of issues in your OpenEdge environment. This SNMP trap action allows you to extend the capabilities of your existing infrastructure without having to train your operators to use Fathom.

- **Compound Action** — This action allows you to combine actions in response to an alert. The best example of this might be a situation where you wanted to send an e-mail (e-mail action) and log the issue to a log file (log action) in response to an alert.

Figure 4–1 shows the Job Create view.

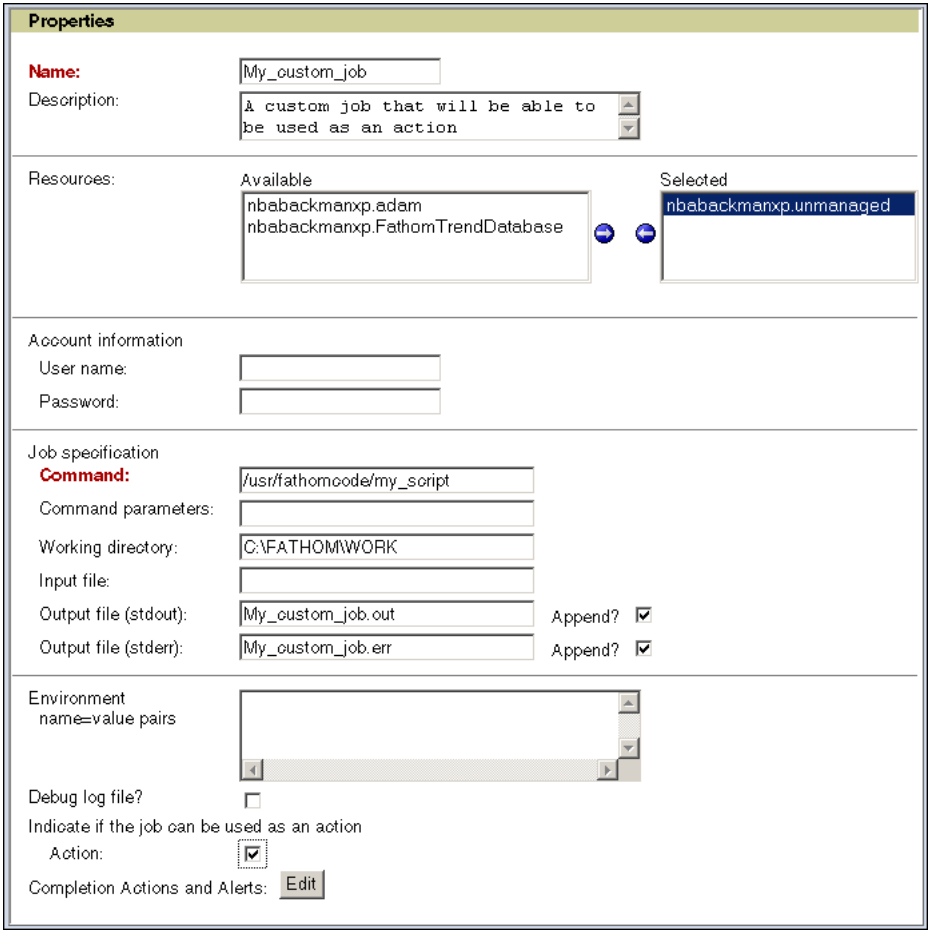


Figure 4–1: Job Create view

Note: You need to insure the **Action** checkbox has been selected in order to use the defined job action.

- **Command Action** —You can also use actions to take corrective steps through the use of jobs as actions, to generate informational messages, or a combination of the two. You are only limited by your imagination and the amount of time you have to spend on development of the background application code. In most cases you want Fathom to deliver notification prior to a problem occurring. This is important because it provides an opportunity to correct problems prior to an outage. In a best-case scenario, users will keep and analyze trending information and take corrective action so alerts will never be fired in the first place. Alerts should be treated like the warning lights in your car. If the light turns on there is already a problem, which could have been taken care of with routine maintenance.

Defining environment-specific rules

Once your installation is complete, you can focus your attention on the nuts and bolts of resource monitoring and trending. This includes determining which resources to trend, which rules to define, and how to determine the actions that those rules will take when a rule is broken.

Scheduling and polling

The default schedule for Fathom is 24 hours a day, 7 days a week. However, each resource can have its own schedule. Once you select a schedule, you can choose a polling interval. The polling interval tells Fathom how often to poll or query the resource for information. A poll without trending will support the alert system within Fathom. The more frequent the poll the greater the impact to the system. You might want to poll some resources frequently, but trend only the resources a few times a day.

Trending

Trending is important because it allows you to gain insight into how and potentially why your system demands are growing. However, it is equally important to know what resource not to trend. If you are trending every resource, you can gain insight into all facets of your application, but the additional load on the system to support this trending can cause a noticeable decrease in performance. The goal is to trend what you need because only a limited number of samples are stored in memory and it is difficult, or impossible, to get information about resources that you are not trending. At the same time you do not want to trend too many resources that would incur significant additional hardware expense to support the monitoring and trending tool.

If you are frequently monitoring and storing trending information, you might want to compress data older than 180 days with the FathomTrendDatabase. This approach will take your individual samples and compress these records into hourly, daily, and weekly records to conserve space in the FathomTrendDatabase. It is also important to have an archiving strategy for your FathomTrendDatabase data just like any other database. You should not discard the data just because it is over one year old. Spend the time and resources to gather the data. This historical data will allow you to provide the business with year-over-year statistics from which to plan.

You can enable trending for the default schedule associated with this database resource by selecting the **Trend Performance Data** check box, as shown in Figure 4–2.

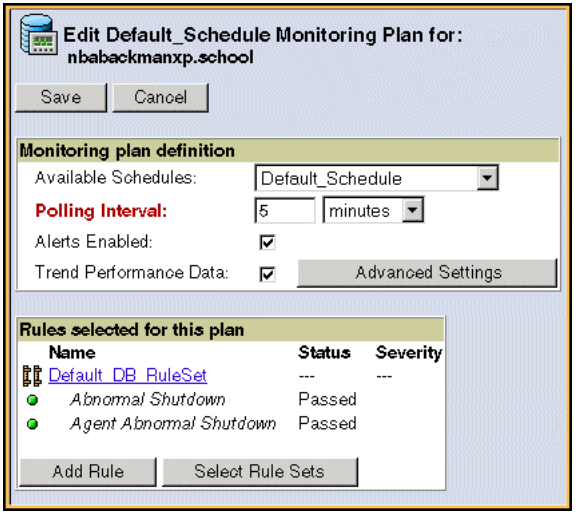


Figure 4–2: Enabling trending

Clicking the **Advance Settings** button displays the page shown in Figure 4–3.

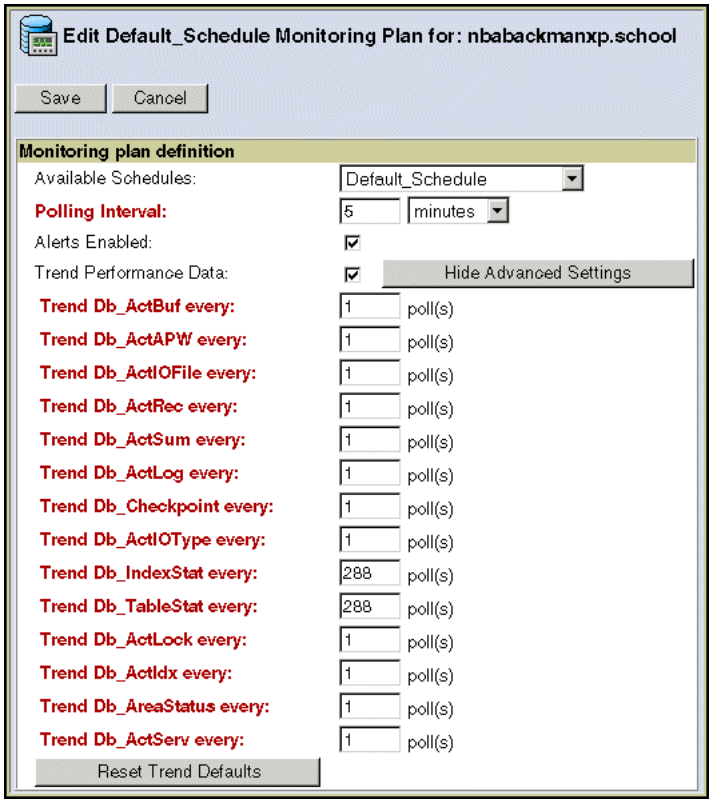


Figure 4–3: Advanced trending settings

On this page, you can specify how often you want Fathom to capture trending information. This is a product of the frequency of polls and the number of polls between trending. If you trend data on every poll, a trending data record is stored for this resource every polling interval.

For example, if you set the **Trend Performance Data Every** field to 1 and the polling interval is set to every 5 minutes, you will store 288 trend records for this resource per day:

```
24 hours * 60 minutes per hour / 5 minutes per sample = 288 samples per day
```

If you want two trending samples per day, change the **Trend Performance Data Every** field to 144:

```
144 samples * 5 minutes per sample / 60 minutes per hour =
    12 hours per trended sample
```

If you want to exercise greater control over when samples are taken, you can create a separate trending schedule. You could set up this schedule to execute at specific times of the day to capture the data you require. This allows you to do your polling at peak periods after your end-of-day cycle or at other specific periods of the day. You could have a single poll or a short defined period to capture a specific period, task, or series of tasks. You should first try using the default values, and then modify these over time to suit your needs.

Rules

Fathom rules provide boundaries for the system. If a rule is broken, an alert or action can be taken based on the rule. Turn on rules only as you need them. This will limit the number of alerts and make the alerts that you do get more meaningful.

It is important to keep your rules simple and focused on your most critical areas. Use the following guidelines when setting up rules:

1. Focus on failures that you have already experienced, such as:
    - Abnormal shutdown of a database.
    - Process terminations.
    - Disk full conditions.

2. Work on common problem sources, such as:
    - Variable area extent growth.
    - High area space utilization.
    - High paging and swapping.
    - High disk space utilization.
    - Log file monitor for the database.
3. Work on housekeeping, such as:
    - Removing any unnecessary rules.
    - Adding additional rules as needed.

Job and report templates

Fathom lets you create job templates and use them as the basis for creating additional jobs. The templates allow you to create business rules for how you want specific tasks to be completed, and you can duplicate the process quickly while maintaining your business rules. Templates provide a quick mechanism to go from configuration to implementation when you are dealing with multiple resources in your environment.

Using job templates makes sense when you have multiple resources that are similar. If you are going to create only one job, it does not make sense to create a template. Also, if you are going to export your resource settings, the template you create might be more useful on other systems.

All reports use templates, so regardless of whether you are going to use the report once or several times, you will create a template for the report.

My Fathom

The operational model of the business determines the ideal configuration of one or more simple screens that are uncluttered and meaningful. The most common configuration is to have vital system resources and your databases on the **My Fathom** page. Other models might include:

- Profiles based on applications.
- Profiles based on servers.
- One server for your real time application, another for personnel and payroll, and a third for accounting and finance.
- Profiles based on Fathom users (database administrator versus system administrator).

Figure 4–4 shows a custom defined **My Fathom** page.

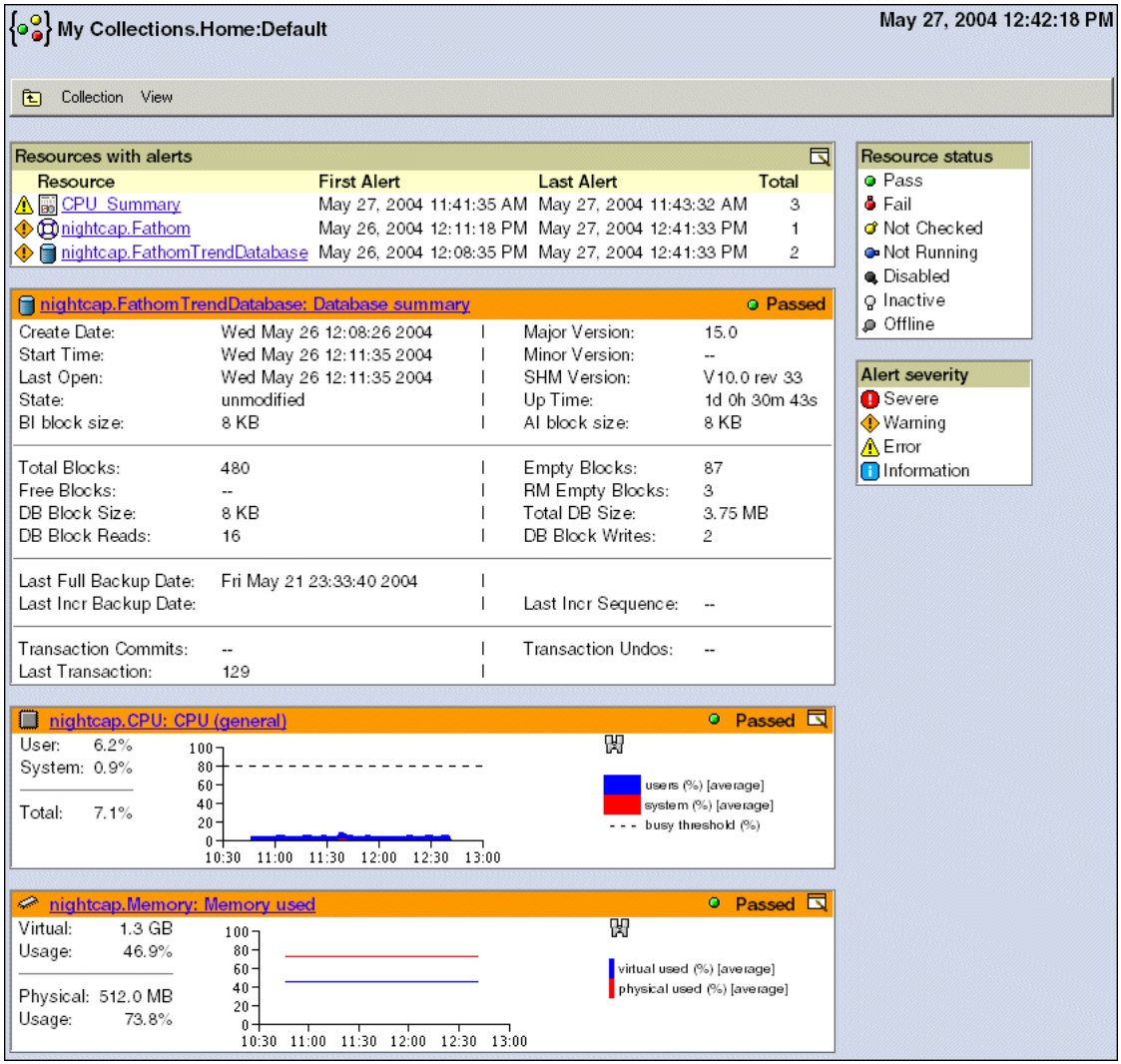


Figure 4–4: Customized My Fathom page

Export resource settings

Fathom lets you *export* your resource settings to provide a basis from which to build your next Fathom environment. This feature is especially helpful for VARs and other users who need to support the same application for a variety of systems. You can export the following items: job templates, actions, database rule sets, report templates, log file search criteria, log file rule sets, and shared views. These exported items can then be imported into a FathomTrendDatabase on a different system.

When you are importing resources from a resource settings file, you can choose how to handle resources that already exist in the current Fathom installation. Your choices are to display an error, keep the existing resource, or replace the existing resource.

Default database monitoring

By default, Fathom polls database resources every five minutes. If trending is enabled, Fathom also maintains trending information for each poll, with the exception of index and table statistics, which are trended once every 288 polls (once a day).

The default rule plan for a database resource contains a small number of commonly used rules. Any additional rules will need to be defined or imported.

Alert setup

Fathom alerts are effectively defined only after your Fathom actions have been defined and saved into the Fathom Management console. You should first define the default action. The initial setting of the default action is to send an e-mail to the address specified during the installation process. You must determine what the default action for an alert will be, and then modify the default action. In most cases, you can leave the default action as is and start creating additional Fathom actions, as needed.

In most cases, it makes sense to set up alert categories based on your business needs. For example, you might want alerts for system resources to trigger an action that sends an e-mail to the system administrator, while database alerts would trigger actions that send e-mail to the database administrator. Severe errors in either area could be directed to both system administrators and database administrators.

It is important to start with a few important alerts and then add others as needed. First take the default rule and add two additional rules, one for Variable Extent Grow for all of your database areas and another for Stopped Trending. These rules will ensure that Fathom is operational on your resources. Once these rules have been defined, you can add additional rules as needed. Make sure that the defined rules trigger an alert only when you want them to, or you will devalue the alert and run the risk of a valid alert being ignored.

Configuring Fathom for your environment

There are many different ways to configure OpenEdge-based applications, but most of these configurations fall into one of the following three types: single machine, multiple machine, or distributed multiple machine.

The simplest and most widely utilized configuration is a single-machine configuration. This is where there is one machine acting as the database engine. The term *database engine* refers to the machine that contains the OpenEdge databases and potentially the application code. However, this configuration also refers to the server in the classic client/server environment where you have many PCs connected to a central machine that serves data. In this configuration, you will keep the FathomTrendDatabase to store your performance trending information on the same machine as the application databases.

A multiple-machine environment is a little more complicated due to the increase in the number of machines and the amount of trending data that will be stored. To successfully configure a multiple machine environment, you need to know how you will use the information. If each machine has a separate purpose and separate administrative staff, it would be wise to separate the trending information as well. In most places, this is not the case. The machines work together and have only one or two administrators to manage them. In this case you would have a central machine act as the Fathom Management Machine (FMM), and the other machines would report the trending data to this machine. If one of the production machines has spare capacity (disk, CPU, memory, and network), you could use that machine as the FMM, but you could also have a dedicated machine to handle this function.

Note: Remember that the FMM machine must have an AdminServer license in addition to the Fathom license.

Having machines share the same FathomTrendDatabase will help reduce the number of places you need to look for performance information and allow the administrators to be more efficient in avoiding system issues. The downside is that there will be an increase in network traffic to accommodate trending data being passed from Fathom client machines to the FMM. You should avoid having more than ten Fathom client machines per FMM for performance reasons.

Determining the location and number of FathomTrendDatabases

Determining the number of FathomTrendDatabases is straightforward in most cases. A small percentage of installations will fall outside these guidelines, but most users should follow these general rules.

If you are in a **single machine** environment you should:

- Have one trending database.
- Trend only those databases essential to business functions.

Because the FathomTrendDatabase is sharing resources with your production application, take care to reduce the impact of monitoring on your other applications. By having only one FathomTrendDatabase and limiting the number of resources to those that are essential to your core business, you can implement Fathom and gain insight into your system's inner workings while limiting your use of shared system resources.

If you have **multiple machines in a local area network (LAN)**, you should:

- Have one trending database.
- Store the trending database on your most reliable machine, where possible.
- Store the trending database where you have spare capacity.
- Trend only those resources needed.

You will introduce additional network traffic by trending remotely, but the ability to trend and report from one database will provide significant benefits over time. If you place the database on a machine that has excess capacity, you can use that capacity to provide a service to your users.

If you have **multiple database machines distributed across a wide area network (WAN)**, you should:

- Have one trending database per site.
- Follow the rules for LAN machines, as described above.

For performance reasons, it is important to keep Fathom monitoring and trending traffic off of your WAN. There is no problem with accessing your trending information across the WAN through the Fathom Management console or through custom programs. However, you should treat each site in your organization like a separate local area network.

Isolating the FathomTrendDatabase

By isolating the FathomTrendDatabase you can limit the impact trending has on other resources. As stated before, the very act of monitoring and trending will have an impact on performance.

This section discusses possible ways to isolate the FathomTrendDatabase along with the costs and benefits associated with each suggestion.

Use a separate machine

If you isolate the FathomTrendDatabase on a separate machine, you can monitor this machine to better understand the impact of resource trending on the machine and the network. The major issue with this configuration is the increased network traffic to support monitoring. This might not be a problem in many cases. If it is a problem, you can overcome it by increasing the bandwidth of your network or implementing a private network. Given that the network is generally the slowest hardware resource, you need to understand that the additional load on this resource will not cause problems for your applications.

Use a private network

One way to eliminate problems in trending across the network is to set up a private network that will support trending. This is fairly inexpensive if the machines are in close proximity to each other. If this is not the case, the cost can increase significantly. Each monitored machine will need an additional network card and one additional router or switch to attach the machines. Once these machines are physically connected, you provide a separate subnet for the machines in the private network. The machines can then communicate over that network for Fathom without disturbing your production network.

Use both methods

By combining both of the previously described methods, you can collect data from a number of machines simultaneously and feed that data back to a central repository. The production machines would incur only the performance cost of collecting the data, while the reporting and storage costs would be on the FathomTrendDatabase server side. There would be additional setup and maintenance costs with this environment, but you need to weigh the value of the collected information with the cost of collecting it to determine if this solution makes business sense for your operation.

Remote monitoring

Fathom Management Version 2.1 introduced remote monitoring capabilities. Prior to that version, you were able to install Fathom on several machines and redirect trending information back to a single machine. This capability is still available; however, when Fathom Management is not installed on a secondary machine, it is still possible to use Fathom Management to manage resources on remote machines. The second machine would require only an OpenEdge license to have the functionality described in this section. The *Resource Monitoring Guide* has a complete description of this functionality. The goal of this section is to explain the capabilities and limitations of Fathom Management's remote monitoring features.

It is important to think about these remote machines just the same as you would think of any other machine: the same rules apply to remote machines as to the machine where the console resides. The only other consideration is the amount of data that you will be transferring between the machines for monitoring and trending, since all of the data will be transferred across the network to the FathomTrendDatabase. You might want to have remote machines on a less aggressive schedule if you think network bandwidth might be an issue.

What resources can be managed?

The answer to this question depends whether or not Fathom Management is installed on the remote machine. When Fathom Management is installed on a machine, it is possible to utilize all of the Fathom Management functionality on that machine, and retain the trending data locally, or send the data to a central system. In this case Fathom Management is not considered to be remote; rather, it is installed in two or more locations while being managed from a single console. If Fathom Management is not installed on a remote system, you must have an OpenEdge license on any machine on which you wish to support remote monitoring.

Limitations of remote monitoring

There are currently three limitations: remote log file monitoring cannot be performed, jobs cannot be run remotely, and scripted databases cannot be started or stopped. Otherwise, all other Fathom Management tasks can be managed from a central console.

Remote database monitoring

Databases can be monitored on remote machines, and trending and alerts for remote machines can be generated through Fathom Management. Remote databases can be started or stopped through the management console.

OpenEdge server support

The real benefit of remote monitoring is for OpenEdge servers. AppServers, WebSpeed agents, and the NameServer can all be remotely monitored through Fathom Management. All OpenEdge server administration functions (add, trim, start, and stop) are supported through the remote monitoring capabilities of Fathom Management.

System management support

All system monitoring and trending capabilities are available through the remote monitoring in Fathom Management.

Setup for remote monitoring

The Fathom Management Remote Configuration utility (`fmconfig`) is used on each machine to enable remote monitoring. This utility must first be run on the machine where the console resides and then it can be run on the remote hosts.

See the *Installation and Configuration Guide* for a further explanation of the initial setup of the remote machines.

Performance considerations

Progress has taken every step to reduce the impact of Fathom on your system. However, you need to think of Fathom as another application in your operation. All applications require system resources (disk, memory, CPU, and so on). Fathom is no different. There are cases where improper monitoring has had an adverse affect on performance. Here again, you need to look at the information you require and the information you desire. The information you require is the information that is critical to your operation. If you need to choose between Fathom monitoring and production performance, production performance will win. In most cases, you can monitor critical functions without any noticeable performance degradation. To determine how much additional monitoring you can perform without impacting your production performance, review these points:

- Consider the number of resources that you are monitoring on the machine and the frequency of the monitoring. If you are doing frequent samples and you have trending enabled for the resource, CPU and memory will be affected when gathering the data, and disk space will be affected when storing the data. It might be important to know the status of the database (if it is running or not) every two minutes, but you might only want to gather and trend storage information a few times a day.

- Consider how you plan to use the trending information for a resource. If you do not know how you will use the data, assume that hourly information is more than enough for most volatile resources and once a day is fine for more static resources. The reason that you want to decrease the frequency of information gathering and analysis is that each time one of these polls takes place there is a “burst of activity.” The more information you ask for, the greater the length of the activity burst. For information that is gathered daily, you can create a schedule that executes at a quiet period to reduce the impact on other applications.

Configuration Advisor

The following sections describes the Configuration Advisor.

What is the Configuration Advisor?

The Configuration Advisor is a tool to help determine the correct alert thresholds within Fathom Management by examining captured database and system statistics within the FathomTrendDatabase and providing suggested values for these alert thresholds. By looking at past data, the Configuration Advisor can determine how your system acts under normal conditions and then provide you with suggestions on the proper settings for each monitored area of your application.

Who is it for?

If you have a good understanding of how your system operates, what you want to monitor and how you want your alerts set, you might not want to use the Configuration Advisor. If you are unfamiliar with Fathom's capabilities, or have purchased new software, or are making significant changes to your system and are unsure as to the effect of these changes, you might want to use the Configuration Advisor to guide you.

How does it work?

After installing Fathom you will need to monitor and trend each resource where you believe the Configuration Advisor might be useful. Initially, it is advisable to monitor and trend all resources; you can later scale back the amount of collected data. Generally, you should gather trending data for a minimum of a week, and even up to a month if your activity varies greatly throughout the month. This might seem like a contradiction of the suggestion made in the "Performance considerations" section on page 4–22, which said to monitor only those things that are necessary and add others as necessary. However, with the Configuration Advisor you need to have accumulated the trend data for a resource in order to use the tool. Once you have used the Configuration Advisor to determine your alerts, you can disable monitoring and trending for resources where data is not necessary.

Once you have gathered the trending data you must define rules for each resource. Again, you can choose to define a rule for every resource or only the rules you think you will need. The Configuration Advisor will provide advice only on resources with rules defined. When you define each rule you should make the rule inactive until you have a setting that you believe will work for you.

Here is an example of the Configuration Advisor used for an AppServer. It is possible to set up multiple monitoring plans (such as weekdays and weekends). Figure 4–5 has multiple monitoring plans set up.

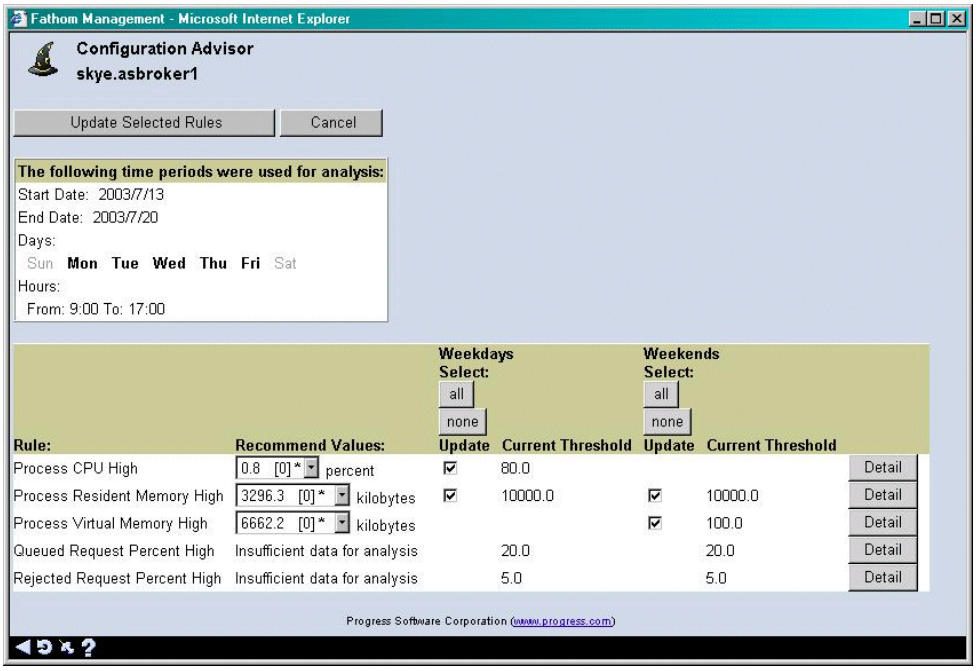


Figure 4–5: Configuration Advisor

The recommended values are based on the time periods used for analysis (in this case weekdays). These values might not be appropriate for the weekend monitoring plan, since activity levels can differ significantly from weekend to weekday. You can rerun the analysis for the weekend plan separately and then provide a threshold for that rule.

Let's take an example in the **Recommended Values** column. The **Process CPU High** shows a value of .8 percent. This is the value that the Configuration Advisor recommends you set as a threshold. Note the number in the parentheses to the right of the recommended value. This is the number of times an alert would have been raised during the sample period if the recommended value were set as a threshold. If you see a high number for this, you should either choose a different threshold or make a system modification and then re-establish the threshold value with new trending data prior to enabling the rule.

You might enable the **Process CPU High** alert. This alert will notify you if the CPU process uses more CPU than normal. This will speed detection of runaway processes because either you will be notified right away of the cause of the problem or you will have eliminated something from your troubleshooting.

This initial monitoring will cause a small additional load on the system during the monitoring period. The value of information gathered by Fathom for use by the Configuration Advisor, and the convenience in setting up proper thresholds warrants the trade-off. Remember to disable unnecessary monitoring after the Configuration Advisor has done its work.

If you have modified the default trending for any resources, it is possible that the Configuration Advisor will not provide an accurate threshold estimate. This happens because is Fathom will trend every poll by default for most resources and if you modify trending to trend every *n*th poll, instead you might have a situation where the polls that are not trended would raise an alert, but the trended poll would not. An example of this would be a situation in which you are polling every five minutes and you are trending every three polls. If two polls came in at 1000 reads per second and the third came in at 100 you would run the risk of only capturing the 100 reads per second poll in your trending. If you set an alert threshold at 200, which is twice your trend data, you would see two alerts if the situation happened again. This is not generally a problem for the vast majority of users, but you should realize that the Configuration Advisor is only as good as the data it uses for analysis.

The File Monitor

This feature replaces the file size option in earlier versions of Fathom. It provides numerous options for monitoring various statistics on files of all types. The file monitor now supports file aging, whether or not a file exists, growth rate, when a file was last modified, as well as current file size. The usefulness of these features extends far beyond monitoring your OpenEdge databases; they can be used for EDI functions, connectivity checking, application monitoring and debugging, and countless other day-to-day IS functions.

To use the File monitor, from the **Resources** screen select **New Resource Monitor**, then **File**. Figure 4–6 shows the **Create File Monitor** screen.

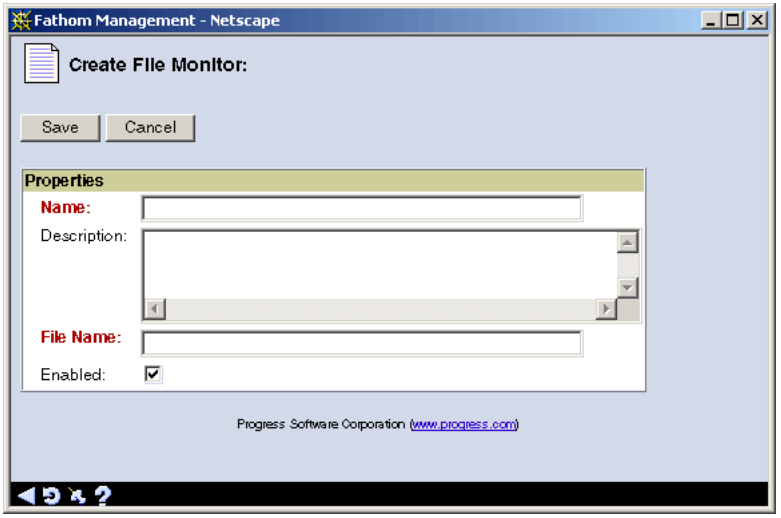


Figure 4–6: Create File Monitor

Enter the **Name** of the Fathom resource and the **File Name** that you want to monitor. It is always a good idea to put some notes in the **Description** so that others who use the Fathom Management console will understand why you are monitoring this file.

Once this information has been entered and you have saved the settings, you will be prompted to enter the monitoring schedule for this resource. You can now enter the rules for this particular file by clicking **Add Rule**. You will see the page in Figure 4–7:

Figure 4–7: Available File Monitor Rules

The following sections present a brief description of each rule and how you can choose to apply it in different situations.

File Age

This alert is raised if a file is unmodified or is older than a specified date-time. There are two ways to take advantage of this feature. One is to make sure that a file or files have been updated more recently than a certain date-time. The other is to detect files that have not been updated for a given period of time.

Using the most-recently-updated approach is extremely useful if you expect periodic refreshes (such as daily updates) of certain application files. For example, you might get an XML feed from a vendor with current pricing and stock information every day. You could avoid problems such as a dropped feed, and ensure a new file is in place every day by raising an alert when the file has not been updated.

The least-recently-updated feature is useful when files that have aged should be removed or archived. This is a powerful tool to help keep systems clean. Users are notorious for leaving output files around for long periods of time, often not knowing they are there, forgetting them, or not caring. Here, the file age rule will allow you to enforce the 30-day-rule and automatically remove output files over 30 days old with no user interaction.

File Exists

This alert is raised if a given file or file type exists or comes into existence. Fathom will look for the file and take action only if the file exists. This rule is excellent for dealing with applications that use operating system flag files to signify a problem. For example, OpenEdge core files can be automatically removed. You could define an action to remove the file simply when it comes into existence, or you can create a complex action to remove the file and send e-mail to the administrator. For example, if your system is shut down every night, you could check file aging for two-day old `LBI`, `SRT`, and other temporary files and remove them. Or, if you discover a `protrace` file, you could move it to a backup directory and send e-mail to the administrator.

File Growth Rate

This feature is equally good for monitoring database growth as well as non-DB files. For example, OpenEdge log files and other system and application files can be monitored with this option. If a file grows at a rate greater than some criterion you specify, an alert can be sent using standard Fathom methods. If you know the cause of the problem — for example, a runaway process — you could have the alert take corrective action on the first sign of the problem and send alerts on subsequent passes if the corrective action was ineffective.

File Modified

This alert will fire if a file is modified in any way. If you have static files that should only be modified by certain people, this would be an excellent way to keep track of when a file is modified. (This is a way to keep those meddling developers in check!) Many applications have files that should only be modified by the system administrator, such as the database parameter file. If anyone modifies the file an alert will be fired. This alert could also take corrective action by retrieving an original copy of the file from an online archive, in addition to sending a message.

File Size

Determining if a file has grown beyond a given size is a constant concern for most administrators. This alert allows you to define a maximum size threshold for a file and take action if that size is exceeded. This action could be a simple notification or a complex action that archived the file and then truncated it to allow for future growth.

Prior to Fathom Management Version 2.1, the administrator needed to either manually check on critical files every day, or write custom code to perform the checks. All of these checks can now be executed though a single, standard interface for any kind of file: OpenEdge, or non-OpenEdge. Alerts can be configured within Fathom, so that as the number of files being monitored changes, there is minimal additional maintenance required. For example you could define one action that notified multiple people when a file was modified. If the list of people notified changed, you would only need to modify one action instead of each individual file resource. This saves time and provides better consistency.

Creating custom reports using 4GL

The data stored within the FathomTrendDatabase is partially normalized. One thing that will help you to create effective reports with SQL is the creation of views to help with queries. Example 4–1 shows an example of a typical view.

Example 4–1: Typical view

```
CREATE VIEW my_view AS
  SELECT
    Pub.cf_Sample.sample_id,
    Pub.cf_Sample.sample_Len,
    Pub.cf_Sample.sample_date,
    Pub.db_areastatus_areaname areaname,
    Pub.db_areastatus_areanum areanum,
    Pub.db_areastatus_totblocks totblocks,
    Pub.db_areastatus_hiwater hiwater,
    Pub.db_areastatus_extents extents,
    Pub.db_areastatus_freenum freenum,
    Pub.db_areastatus_rmnum rmnum
  FROM pub.cf_sample
      LEFT OUTER JOIN pub.db_areastatus
      ON pub.db_areastatus.sample.id=pub.cf_sample.sample_id;
```

Fathom data is also accessible through the 4GL. You can define how you want to view the information. If you require all of the information for a resource, you can start with a simple query of the `Cf_Resrc` table, and then expand on the search criteria to narrow down the information that you really need. If you want to see what occurred on a specific day, start with the `Cf_Sample` table then move to the table that contains the information you require. There is more information regarding what data is stored in the tables within Fathom in the *FathomTrendDatabase Guide and Reference*.

Example 4–2 shows a 4GL program created to display database activity summary information based on a particular date.

Example 4–2: 4GL program to display database activity summary

```
FOR EACH Cf_Sample WHERE
  Cf_Sample.Sample_Date = 4/15/2002:
  FIND FIRST Cf_Resrc WHERE
    Cf_Resrc.Resrc.ID = Cf_Sample.Sample_ID NO-ERROR.
  IF AVAILABLE Cf_Resrc THEN
      DO:
        FIND FIRST Db_ActSum WHERE
            Db_ActSum.Sample_ID = Cf_Sample.Sample_ID NO-ERROR.
        IF AVAILABLE Db_ActSum THEN
        DO:
          DISPLAY
              Cf_Resrc.Resrc_Name FORMAT “x(15)”
              Cf_Resrc.Resrc_Loc FORMAT “x(15)”
                STRING(Cf_Sample.Sample_Time,”HH:MM:SS”) LABEL “Time”
              Db_ActSum.ActSum_Commits
              Db_ActSum.ActSum_DbAccesses
              Db_ActSum.ActSum_DbExtend
              Db_ActSum.ActSum_DbReads
              Db_ActSum.ActSum_DbWrites
              WITH FRAME frame_x 1 DOWN 1 COLUMN.
        END.
      END.
END.
```

Example 4–3 shows an example of the 4GL code required to display information about a specific resource.

Example 4–3: 4GL program to display resource information

```
FIND FIRST Cf_Resrc
    WHERE Cf_Resrc.Resrc_Name = “TrendDatabase” NO-ERROR.
IF AVAILABLE Cf_Resrc THEN
DO:
  FOR EACH Cf_Sample WHERE
      Cf_Sample.Resrc_ID = Cf_Resrc.Resrc_ID:
  FIND FIRST Db_ActSum WHERE
      Db_ActSum.Sample_ID = Cf_Sample.Sample_ID NO-ERROR.
      IF AVAILABLE Db_ActSum THEN
      DO:
         DISPLAY
             Cf_Sample.Sample_Date
             STRING (Cf_Sample.Sample_Time,”HH:MM:SS”) LABEL “Time”
             Db_ActSum.ActSum_Commits FORMAT “>,>>9”
             Db_ActSum.ActSum_DbAccesses FORMAT “>,>>9” LABEL “Accesses”
             Db_ActSum.ActSum_DbReads FORMAT “>,>>9”
             Db_ActSum.ActSum_DbWrites FORMAT “>,>>9”
                 WITH DOWN FRAME frame_x.
      END.
  END.
END.
```

Viewing archived data

Archived data is treated differently than actual trend data. You will always deal directly with the table that contains the data, regardless of whether you are looking at a specific resource or specifying a date range. Example 4–4 shows an example of how to display archived database summary information.

Example 4–4: Code to display archived database summary information

```
FIND FIRST Cf_Resrc WHERE Cf_Resrc.Resrc_Name = “TrendDatabase” NO-ERROR.
IF AVAILABLE Cf_Resrc THEN
DO:
    FOR EACH ar_actsum WHERE
        ar_actsum.Resrc_ID = Cf_Resrc.Resrc_ID:
      DISPLAY
        Ar_ActSum.ActSum_EndDate
        STRING (Ar_ActSum.ActSum_EndTime, “HH:MM:SS”) LABEL “Time”
        Ar_ActSum.ActSum_Commits FORMAT “>,>>9”
        Ar_ActSum.ActSum_DbAccesses FORMAT “>,>>9”
        Ar_ActSum.ActSum_DbReads FORMAT “>,>>9”
        Ar_ActSum.ActSum_DbWrites FORMAT “>,>>9”
            WITH DOWN FRAME X.
    END.
END.
```

Creating custom jobs

Progress has bundled Perl with Fathom. Perl is a programming language similar to the Korn shell in UNIX or Linux and command procedures in Windows. The main benefit of Perl is that it is supported on all of the platforms that OpenEdge supports. Any Perl programs you create in one environment should work in a different environment with little or no modification. This would not be the case if you used command procedures in Windows and decided to migrate your application to UNIX. In this case, you would need to completely rewrite all of your supporting programs. Fathom can integrate with Perl or any other programs you have on the system, so if you are already comfortable with the Korn shell, use that language to create your jobs.

Fathom allows you to perform different actions, depending on the return code of the program. The default success code for the korn shell is 0. You can have Fathom generate an alert for any return code that is generated by the program. For example, on a return of zero you can put a message in the log that your program completed successfully. On a non-zero return code, you could generate an alert to send an e-mail or page to the administration staff. Any alert that is available within Fathom is available to you at the command line.

To fire a Fathom alert from the operating system, use the following command:

```
Fathom -firealert [name_of_alert]
```

For more information about this command, see the *Alerts Guide and Reference*.

Each job defined within Fathom has its own set of return codes. Because of this, you must define the actions for each program individually. This can be cumbersome if you run many programs from within Fathom. A solution to this problem is to have your own return code analyzer. This is an external program that reads the return code and generates the proper Fathom action from the command line. The benefit to this mechanism over defining the alerts from within Fathom is maintenance. If you want to globally modify what action is performed on a particular return code, you need modify only one program rather than modify each job individually.

The following examples show you how to set up the logic for this type of processing. The bulk of this code could be set up as a subroutine that was called for each job. If a change were needed, you would have to only modify the subroutine.

Example 4–5 shows a Korn shell example.

Example 4–5: Korn shell example

```
#!/bin/ksh
FATHOM_BIN=${FATHOM_BIN-/usr/Fathom/bin}
Program_to_run
return_code=$?
case $return_code in
0)
 $Fathom_BIN/Fathom -firealert all_is_ok
;;
1)
 $Fathom_BIN/Fathom -firealert program_error
;;
*)
 $FATHOM_BIN/fathom -firealert warning_unknown_error
;;
esac
```

Example 4–6 shows a Perl example.

Example 4–6: Perl example

```
#!perl
$FATHOM_BIN = "/d14/fathom/bin";
$EXITCODE = system("program_name");
CASE: {
(($EXITCODE < 0) || ($EXITCODE > 1)) && do{
        system("$FATHOM_BIN/fathom firealert program_error");
          last CASE;
    };
($EXITCODE == 0) && do{
     system("$FATHOM_BIN/fathom firealert all_is_ok");
      last CASE;
    };
($EXITCODE == 1) && do{
     system("$FATHOM_BIN/fathom firealert program_error");
      last CASE;
    };
}
```

Fathom gives you the flexibility to choose the method that is right for your business. Some users prefer individual return codes per job, while other users prefer the return code mechanism, and still others use a combination of the two. You have the ability to customize your environment to meet your business needs.

Extending usefulness of existing Fathom functions

Although Fathom is rich with functionality, there are times when you might need to extend existing Fathom functions to meet your exact needs. The following example illustrates how to take an existing feature (the Log Monitor), and add some custom code to create a warm standby environment by applying after-image logs to a secondary server. This operation is rather difficult to automate and is an example of a task where Fathom can take a major burden off the hands of the administrator

After-image administration

The process of archiving an AI extent is fairly straightforward.

The basic steps are as follows:

1. Create an action that takes a “full” after-image file, archives it to a separate directory, and then marks the file as empty and available for reuse.
2. Create a rule on the log monitor to look for the string “After image extent switch.”
3. Set the action you created in Step 1 to the rule you created in Step 2.

There is a job template of this procedure that you can apply to your environment in the downloads section of the Progress Software Developers Network (PSDN) Web Site (http://www.psdn.com). The example on PSDN can be used as is or could be extended to apply the after image extent to a second database to create a warm standby environment. The general process is the same as the steps above; however, the action defined in the first step would be more complex.

Here are the general steps you would need to consider when doing asynchronous replication:

1. Check to make sure that the remote host or replication database or both is available.
2. Archive the oldest “full” after-image file.
3. Copy the after-image file to the remote host (if necessary).
4. Apply the after-image file to the replication database.
5. Mark the after-image file as empty.
6. Repeat if additional “full” after-image files are present.

Each of these steps must be completed successfully in order for the next step to begin. Thus, you would need to create a return code that would be sent to Fathom to indicate failure of any given step. Every command will return a status code upon completion. A status of 0 means success in most cases and a non-zero number indicates a possible problem. In UNIX, you can generate these return codes yourself by using the `exit` command in your scripts. The syntax of the `exit` command is: `exit #`, where `#` is the status that you want to return to the calling program. As the author of the job to run your shell program, you have the ability to modify what action is taken based on the completion code.

Figure 4–8 shows an example of completion actions and alerts.

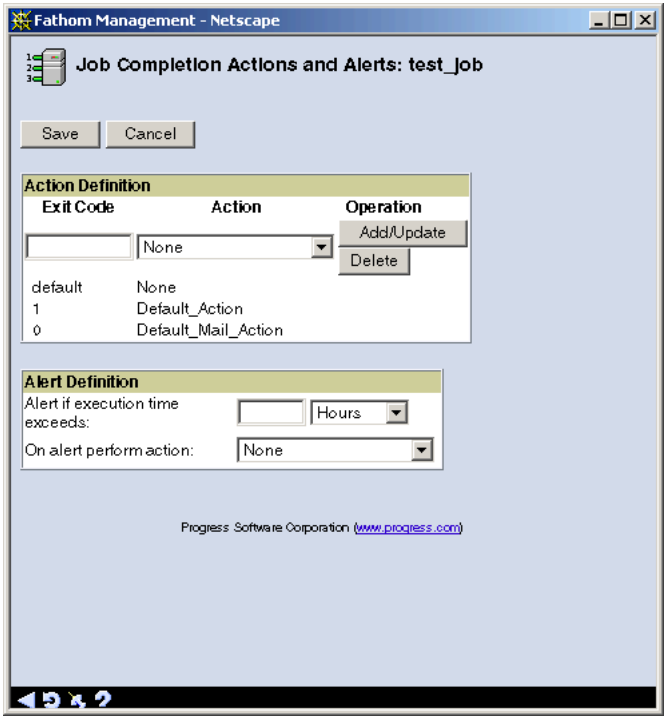


Figure 4–8: Job Completion Actions and Alerts

The UNIX shell script below tests for the existence of a file (actually any one of a group of files). If any file is found, the script will exit with a return code of 0, indicating success. Otherwise, it returns 1, which signals "no files are found." The `ls` command in this script will actually give its own return code. In this case the output of the command is not used, as both standard output and standard error are being sent to the bit bucket (`/dev/null`). The `$?` is the value of that return code. This code allows you to take all of the non-zero return codes and return a 1, which makes it easier to handle inside of Fathom or within other shell scripts:

```
#!/bin/ksh

ls *filename* 2>/dev/null >/dev/null

if [ $? = 0]
then
  exit 0
else
  exit 1
fi
```

This script is used to look for an input file in a directory. If the file is found, an action is defined to process the file.

If you need more robust replication than described above, or if you would like to set up a report server, Fathom Replication can provide real-time replication of data to a secondary database. In the case of a reporting DB, you can run extensive reports against the copy, freeing up resources on the primary (production) database. For more information, see the *Progress Fathom High Availability Replication User's Guide*.

Uses beyond monitoring

You can and should use Fathom Management to do other daily tasks beyond monitoring and manipulating OpenEdge-related files on your system. If you have an EDI system that has a staging area, you can check to make sure that area is emptied out on a regular basis by using file system thresholds or by viewing a log file related to the EDI process.

Any task you do on a daily basis is a good candidate for automating with Fathom Management. The trouble with manual processes is that they are only as good as the people running them. If the administrator forgets to do the task, or misses any step in the process, the task might not succeed. A well-managed system will eliminate human intervention whenever possible. Fathom Management is a way to automate complicated tasks using a common interface and well-established mechanisms for alerting an administrator if there is a problem. It is so easy for a busy administrator to forget to check an application log file or simply ignore the task for a few days. With Fathom Management, no task is ever overlooked or done improperly, and alerts can be delivered to several people at the same time to give greater visibility to an issue before it becomes critical.

If you have existing scripts that have been well tested you might want to integrate them into Fathom to allow for consistent reporting on issues of importance. Fathom provides a standard interface to all tasks, and global changes can be made to task steps from within the Fathom Management console should the need arise.

Fathom for system sizing

Fathom can also be extremely useful when it is time to upgrade a system. In the past, the steps for upgrading a system went something like this: “The system is slow; we need to get another one.” Now you can view historical Fathom Management data to determine what portion of the system is slow. In some cases, you will be able to simply modify an OpenEdge parameter, such as `-B`, to increase the efficiency of the database and avoid spending any money on hardware. In other cases where you have tuned the system well and you still have performance issues, you can use Fathom Management to determine where the problem lies rather than guessing at the source of the problem.

The key here is knowing what is growing and why. Perhaps management has said that you need to add 100 users in order to meet demand next month. What does this mean in terms of system utilization? Beyond the basic additional resources for 100 users, increasing `-n` and making sure you have enough memory to accommodate their sessions, what do you need? What will these sessions be doing? How are they distributed among job functions? What is the reason for growth? For example, is there going to be an increase in orders, and if so, how much of an increase? Management will be able to provide you with the answers to some of these questions but you will need to use the trending data from Fathom Management to determine the effect on the system from these business growth decisions.

Do you need to plan for peaks in activity? If you have a “regular” processing cycle but month-end has a much greater effect on the system, then you need to examine your peaks. On a regular day you might do 5000 orders and 1,000,000 record reads, which result in 5,000 operating system reads, so your buffer hit percentage is 99.5%. But during month-end you do 5,000,000 records and 1,000,000 operating system reads with a buffer hit ration of 80%. The effect of this is heavy reliance on your disks. If your disk subsystem can perform 200 I/O operations per second, you would be within any margin of safety. However, if you add the additional users and this doubles your record reads, you will need to increase disk throughput capacity to handle 400 I/O operations per second, or your system will not be able to handle the additional load.

If you have historical data, similar to Figure 4–9, showing how month-end has changed over time, you can determine what will happen as you increase the load on your system.

	Regular				Month End	
Month	Orders	Records Read	OS Read		Records Read	OS Read
Jan	1,000	200,000	1,000		40,000	8,000
Feb	2,000	400,000	2,000		490,000	98,000
Mar	3,000	600,000	3,000		1,440,000	288,000
Apr	4,000	800,000	4,000		2,890,000	578,000
May	5,000	1,000,000	5,000		5,000,000	1,000,000
Predicted:						
Jun	6,000	1,200,000	6,000		7,290,000	1,458,000
XXX	10,000	2,000,000	10,000		22,000,000	4,500,000

Figure 4–9: Historical data

Without trending and historical information you are blind. Things might look fine today and you might have a feeling for what it would take to add 100 users. But you have no ability to predict reliably. Many companies have spent a significant amount of money on a system upgrade just to find out that it does not solve the problem. With Fathom Management, you can pinpoint where the problems are and configure the system to correctly solve those problems. For example, if you are using 50% of your CPU on wait I/O time and your disks are working as hard as they can, you would not want to upgrade the CPU and memory capacity of the system until AFTER you increased the disk throughput capacity. By making the individual areas of the system more visible and trending each resource’s data over time, you should be able to make intelligent suggestions for improvements and save money by investing in the right areas.

For Application Resellers, Fathom Management can provide valuable information to use for new clients. If you are monitoring your existing clients and trending their resource data, you can use that data to provide projections for new clients when they inquire about how much system they will need to support their business. If a prospective client asks how much hardware will be needed for 100 users, you can look at your historical data for other 100-user systems and provide estimates based on similar use configurations. Along the same lines, if an existing 100-user customer is consuming 80 percent of the system capacity, and a new 100-user client is coming on line who might be adding new users in the mid-term time frame, you would want to specify greater capacity at the outset for the new client.

Another use for Application Resellers is in development or initial deployment. In development, it is difficult to predict how things will work in the production environment. If application resellers can access the trending data, particularly with an eye for nonlinear trends, then they can identify and resolve problems in the application long before the customer determines that there is a problem.

The rich, data-gathering capabilities of Fathom Management can provide information for the small or large individual shop, or the application partner with one or more customers.

Troubleshooting your Fathom installation

Use the following tips to troubleshoot your Fathom installation:

- Check the `admserv.log` file for error messages. Fathom places all information regarding problems in this file.
- If pages within Fathom are displaying errors such as #503, complaining about problems with compiling JSP pages, try cleaning out the `jspwork` directory. This will remove all cached files and force the Java Virtual Machine (JVM) to rebuild the files as screens are encountered. The `jspwork` directory is a subdirectory under your Fathom installation directory, by default, `C:\Progress\Fathom\jspwork` on Windows and `/usr/Fathom/jspwork` on UNIX and Linux. You can delete all of the files and directories under this directory, but be careful not to delete the directory itself or other files under the Fathom directory.

- The `logs` directory in your Fathom installation directory should be periodically cleaned up. This directory contains, among other things, request logs for the Web server. This allows you to see what has been requested from your system and when the request was made. There is a new log for each day. You could write a simple job to periodically clean out these files.
- To help in debugging reporting and job issues, check **Debug Log file**. By checking this box, you will get additional debugging information regarding the report or job.

Frequently asked questions and answers

This section provides solutions to the most common problems encountered during the setup and installation of Fathom.

Question 1

Why do I see an `admSQL.log` file in my Fathom home directory?

Answer

This log is created when tracing is turned to the verbose (high) setting.

Question 2

How do I debug strange behavior and page hangs when running the Fathom Management console?

Answer

You might need to disable the generation of charts. The default state of chart generation is "enabled." To disable all charting within Fathom, include the following line in `fathom.init.params`:

```
pscChartingDisabled=true
```

If you run Java utilities and Fathom at the same time, generating graphics within Java sometimes causes problems. By turning off chart generation from within Fathom you eliminate this variable and perhaps resolve the issue. Additionally, it is also good to verify that there are no errors being reported in the `admserv.log` file.

Question 3

How can I modify the amount of information that is written to the log files from Fathom?

Answer

To allow customers to reduce the amount of information written to the log files, specifically `admserv.log`, a switch called `LogLevelFathom` has been added that can be set to an integer value to control the amount of information that is written to the log file.

The valid values range is from 0 to 4. A lower number indicates you want only severe errors and a higher number indicates you want verbose output. The logging levels are as follows:

0 = Severe errors only

1 = Error

2 = Warning

3 = Informational

4 = Verbose

Question 4

How do I resolve the `Error initializing Fathom osmetrics library` error when starting Fathom or the `One or more properties changes are invalid` error when moving from page to page within Fathom?

Answer

Disabling the Remote Registry Service on Windows prevents `osmetrics.dll` from loading. To resolve the problem you need to start this service. You can access the service properties by selecting **Start→Settings→Control Panel→Administrative Tools →Services→ Remote Registry Service**.

Question 5

I have a license and installed Fathom. However, Fathom will not run and I receive a message that I am not licensed to run Fathom. How do I resolve this issue?

Answer

You might be looking at an incorrect configuration file. In the `Fathom.init.params` file there is an entry called `FathomLicenseFile` that allows you to specify the location of your configuration file (`progress.cfg`). The default entry looks like this:

```
SET fathomLicenseFile=/usr1/fathom/fathom.cfg
```

Question 6

I am having trouble using environment variables in Windows. Some variable references work, while others do not. What is the problem?

Answer

You can use variables in Windows that use UNIX syntax with the use of the `$VAR` syntax rather than the "`%VAR%`" syntax, as is required throughout Windows. However, the variable must be specified in the `${ENVVAR}` format if the embedded variable is in a path of longer string.

For example:

Value	Outcome
$TESTVAR	Works
${TESTVAR}	Works
$TESTVAR/dir1/foo	Fails
${TESTVAR}/dir1/foo	Works
/dir2/dir3/$TESTVAR/foo	Fails
/dir2/dir3/${TESTVAR}/foo	Works

Question 7

How do I resolve the following error?

```
FATAL ERROR in native method: JDWP "threadControl.c" (Apr 27 2000), line 878:
Unable to create thread table entry.
```

Answer

Increasing the Java limits might resolve the problem. The `-Xmm64m -Xss16` parameters in the `java_env` file control the Java limits.

For UNIX systems, you need to modify the correct `JVMARGS` entry. To determine which entry to modify, you need to know the type of operating system you are using. This can be done with a `uname` command with no options. This command will return the operating system type. Once that has been determined, you need to edit the `java_env` file and search for your operating system type. You need to edit the `JVMARGS` entry just below your operating system with the values that you want.

For Windows, use `regedit` to modify the values.

Question 8

I am getting a `Java out of memory` error on my system. How do I fix this?

Answer

On UNIX systems, change the values `-ms` and `-mx` in the `java_env` file.

On Windows systems, add the arguments to the registry key `JVMARGS` in:
`HKey_Local_Machine\Software\PSC\Progress\`*`OpenEdge_Release_Number`*`\JAVA`.

For UNIX systems, you need to modify the correct `JVMARGS` entry. To determine which entry to modify, you need to know the type of operating system you are on. This can be done with a `uname` command with no options. This command will return the operating system type. Once that has been determined, you need to edit the `java_env` file and search for your operating system type. You need to edit the `JVMARGS` entry just below your operating system with the values that you want.

For Windows, use `regedit` to modify the values. Values of 32 and 64 are acceptable.

Conclusion

Users view the application, database, and operating system as a single entity. Ultimately, user satisfaction is determined by the reliability, speed, and maintenance of the entire system. Generally, system and database administrators tend to focus on the details of individual resources, just as most tools tend to focus on a single aspect of the system. Fathom is the best tool to collect and organize all vital resource information. Also, Fathom compares interactions of each resource with other resources to help you make decisions regarding resource utilization and system architecture.

We have looked at the advantages of Fathom as a way to look into both your system and your OpenEdge resources through a common interface. The hardware portion of all systems requires you to look at the network, disk, memory, and CPU resources in an effort to move any potential bottlenecks from the slowest to the fastest resource.

With this said, we have seen that the best performance is important only if the system is available to the users. Availability can be improved by increasing system resiliency to eliminate outages or, in the worst case, reducing the length of an outage through advanced recovery planning. Regular maintenance can also improve both reliability and performance.

We have seen examples of how Fathom can provide a snapshot of how the system is operating at the present time. We have also seen how Fathom can be used to trend operating systems as well as OpenEdge resources over time to provide data points for capacity planning. Beyond that, Fathom can be used to notify you of resource usage issues prior to affecting users.

A well managed system will increase user productivity and confidence, and allow the system manager to sleep well at night.

Glossary

A

Action

A user-defined process set to automatically occur in response to the status, availability, or performance information of a monitored resource.

Administrator

A user who has access to all Fathom™ Management functionality without restrictions.

Alert

An indication of a real or potential problem in a monitored resource.

B

Bottlenecks

Processing conflicts.

Busy extent

An `AI` extent that is currently active.

C

Checkpoint

A synchronization point between memory and disk.

Chunk

See *Stripe*.

Closed extent

An AI extent that contains notes and cannot be written to until the extent is marked as empty and readied for reuse by the database administrator. Also known as a *Full extent*.

Console

See *Fathom Management console*.

CPU queue depth

The number of processes waiting to use the CPU.

Custom job

A user-defined task that is executed according to a user-defined schedule.

See also *Database maintenance job* and *Job*.

D

Database aware

The ability to scan each block to ensure it is the proper format during the backup process.

Database engine

The machine that contains the OpenEdge™ databases and potentially the application code.

Database maintenance job

A Fathom-supplied, specialized job template from which job instances can be created. The predefined database jobs address fundamental OpenEdge database maintenance activities.

See also *Custom job* and *Job*.

Database object

A table or index.

Database rule sets

A set of rules that you can define and then associate with one or more database resources.

Default rules

Fathom-provided default settings that define rules for resource monitors that you can use to quickly set up and/or update resource monitoring rules. Default rules are set at the resource type level. You can override them at the individual resource level.

Demand-page executables

Reserve text or static portions of an executable that is placed in memory and shared by every user of that executable. Also known as *Shared executables*.

Detail frame

The largest frame of the Fathom Management console that displays information and tools related to the selection made in the list frame.

E

Empty extent

An extent that is empty and ready for use.

Export

To place a copy of a Fathom component's definition into a file that you can then import to another machine and use.

Extents

Physical files that are stored at the operating system level.

F

Fathom Management console

A Web-based interface used to access all of Fathom's functionality.

Fathom System Architecture

Fathom Management is comprised of three components:

- A Web-based management console, which provides a central location for viewing and configuring resources that are monitored by Fathom.
- A database agent, which monitors and manages databases and gathers data from Virtual System Tables (VSTs) for reporting.
- A database, which stores all data collected by agents for use in reporting.

FathomTrendDatabase

A database that stores all data collected by Fathom agents.

Fragmenting the record

A record that has been divided between two or more blocks on disk.

Full extent

An extent that has been filled with data. In the case of an `AI` extent, it cannot be written to until the extent has been backed up and marked empty. If a data extent is full, it is still updatable. Also known as a *Closed extent*.

H

High-water Mark

A pointer to the last formatted block within the database storage area.

I

Idle time

A time period that means that the CPUs have capacity to support growth.

Import

To add a component definition from an import file to your project.

Index block split

The process of making additional space for an inserted index entry.

Indirection

The inability of a single inode table to address all of the physical addresses in a file.

Inode

A mapping of logical to physical addresses in a file.

Inode table

A table of contents on disk used to translate logical addresses to physical addresses.

Interleaved memory

Striped memory chunks. This is similar to striped disk blocks.

J

Job

General term used to identify a task executed on regularly scheduled intervals. Fathom Management supports two types of jobs: custom and database maintenance.

See also ***Custom job*** and *Database maintenance job*.

Job instance

An individual job derived from a job template. A job instance has schedules that define when Fathom runs these jobs.

See also *Job template*.

Job template

A template of predefined, common values from which individual jobs, called job instances, can be created and separately maintained.

See also *Job instance*.

L

Latch

One or more locks to ensure two updates cannot happen simultaneously.

List Frame

The vertical frame that displays the full length of the left side of the Fathom Management console and displays items related to the selection made in the menu bar.

M

Management console

See *Fathom Management console*.

Menu bar

A horizontal bar at the top of the Fathom Management console that lists the following options: My Fathom, Alerts, Resources, Library, Reports, Jobs, Options, and Help. The menu bar also displays the name of the machine and the username entered on the logon screen.

Metaschema

A set of definitions for the internal structure of the data.

Monitor

The combination of a resource, a schedule, and an alert that appears in the interface in response to rules being met.

Monitoring an object

To set up criteria by which you can keep track of an object's performance. You can adjust the criteria for performance output according to your expectations.

Monitoring plan

Defines a block of time that a resource is to be monitored and the processing rules that are to be checked during the defined time frame. The basic elements used by all resource monitoring plans are schedules, rules, alerts, and actions.

My Fathom Home Page

A default, private custom view that Fathom creates for each Fathom user.

O

Optimistic read

A data-retrieval technique where the disk that has its read/write heads positioned closer to the data will retrieve information.

Optimizing memory

Taking advantage of the memory that is available to you.

R

Report instance

The entity you schedule to run in order to produce report results. The report instance identifies the details to be reported on.

Report template

One of over 20 report templates provided by Fathom or created by you that you use to create a custom report of your own design.

Resiliency

The ability to recover in the case of a disaster.

Resource

A specific component of your system that is monitored by Fathom, such as database, files, database and log files, CPU, memory, disk, file system, TCP, UDP, and HTTP ports, and Ping (ICMP).

Rule definitions

The specific attributes of a resource to be monitored.

Rule set

Two or more individual rule definitions. Rule sets are defined and stored in the Fathom Component library and can be shared among resources that belong to the same resource type. Rule sets can be defined for log file monitors, database resources, and OpenEdge server resources.

Fathom provides default rule sets and also supports user-defined rule sets.

Rules

Criteria by which a resource's performance is measured.

S

Shared executables

See *Demand-page executables*.

SNMP trap action

A specific type of action that allows Fathom resource-related event notifications to be sent to your SNMP management console.

Social blocks

Blocks that contain records from different tables.

Stripe

Part of a disk included in a bigger file system structure. Also known as a chunk.

Swapping

The process of taking an entire process out of memory, placing it on a disk in the "swap area," and replacing it with another process from disk.

T

Tainted flags

A flag that tells OpenEdge there is a problem or abnormality with the database.

Total blocks

The total number of allocated blocks for the storage area.

Trend, Trending

The process of identifying and storing data in the FathomTrendDatabase.

Index

Numbers

A

B

C

D

E

F

G

I

J

L

M

N

O

P

R

S

T

V