In this reading sample, we'll explore how to define logical system devices for SAP ASE, which is a step that occurs after installation but before you can build database objects. In the selection from Chapter 15, you'll review how to work with your server for preventative maintenance.

Introduction to SAP ASE System Administration
"Defining Physical and SAP ASE-Mirrored Devices to the Server"
"Preventative Maintenance Regimen"

Contents
Index
The Author

Jeffrey Garbus
SAP ASE 16 / Sybase ASE Administration
720 Pages, 2014, $69.95/€69.95
ISBN 978-1-4932-1182-1

www.sap-press.com/3722
Introduction to SAP ASE System Administration

The system administrator (also referred to as SA, database administrator, or sometimes DBA) of an SAP ASE server is responsible for all aspects of creating, maintaining, and monitoring the server and database environment, including the following:

- Installing SAP ASE
- Establishing connectivity between clients and servers
- Installing and maintaining user databases
- Establishing security for SAP ASE
- Maintaining and troubleshooting SAP ASE
- Establishing disaster recovery procedures
- Monitoring SAP ASE
- Running utilities

This list isn’t comprehensive; there are many duties implied by this list that may fall to the DBA/SA. Let’s look at an example of an SA’s responsibilities, before we discuss them specifically.

Example

A stored procedure that previously ran in 30 seconds has recently been running for over an hour. As SA, you’ll need to find out why it’s taking longer now than before. If it’s a configuration problem, then it falls into the troubleshooting category for the SA to resolve. But if the SQL inside the procedure was poorly written, then tuning the SQL falls into a gray zone. In some shops, the programmer is responsible for the performance and tuning (P&T) of the procedures, and in others (our recommendation), the DBA reviews all procedures going in to the production system and approves the procedures based on his knowledge of the system and optimizer. Either way, it usually falls to the DBA/SA to resolve optimizer problems. In the end, the SA is responsible for the server. All aspects of SAP ASE are in the domain of the SA.

We’ll now take a closer look at the SA tasks mentioned in the preceding list and mention in which chapters you can find those topics discussed, starting in Chapter 2.
Administration Tasks
This book provides the details you’ll need to supplement the existing SAP ASE information that’s already available to you. First, however, we’ll provide an overview of the tasks you’ll need to accomplish, in typical order.

Installing SAP ASE
Discussed in detail in Chapter 2, the process of installing SAP ASE is the first step toward a productive database management system. Always read the Installation Guide that’s specific to your platform, as there are differing requirements and recommendations for different operating systems (OSs). In particular, walk through the checklist provided and make sure that you meet or exceed all OS requirements.

Analyze the user databases you intend to create on this new server, and size your disk space, memory, and other configurable resources to accommodate the current needs. Be sure to also estimate growth in the foreseeable future, and ensure that the resources you’ll need will be available. There are many calculations in the System Administration Guide to help you calculate resource needs, especially in the chapter on Configuring and Tuning the Server.

Establishing Connectivity between Clients and Servers
For clients and servers to communicate, they must be able to locate each other by creating an address listing, which is also discussed in Chapter 2. On UNIX machines, this address listing is called the interfaces file, and on Windows NT, it’s called the sql.ini file (the term “interfaces” is used to generically refer to any of these files, regardless of OS). Within these files, you’ll list all servers (SAP ASE servers, backup servers, monitor servers, XP servers) with their network addresses. Clients will use these files to find the server they want to query, and servers will use them to listen for client requests.

Within these interfaces files, there may be multiple server entries, each of which may contain multiple services. A service is a specific address used for a specific purpose. The master service (featuring the keyword master) is used for servers to track incoming requests; the query service (with the keyword query) is used by clients to locate a server when sending data queries or commands. Each service will contain a specific network location and information about supported network protocols. These entries look different depending on the OS.

Example
SYBASE_PROD
query tcp ether my_machine 4500
master tcp ether my_machine 4500

The easiest way to maintain these files is with the dsedit utility.

SAP ASE 12.5 introduced the Lightweight Directory Access Protocol (LDAP), which will replace the interfaces file if it’s used. LDAP is an industry standard for accessing directory services. Directory services allow components to look up information by a distinguished name (DN) from an LDAP server that stores and manages server, user, and software information that is used throughout the enterprise or over a network.

The LDAP server can be located on a different platform/machine from the one on which SAP ASE or the clients are running. LDAP defines the communication protocol and the content of messages exchanged between clients and servers. Messages are operators, such as client requests for read, write, and query, and server responses, including data format information.

The LDAP server stores and retrieves information about the following:

- SAP ASE, such as IP address, port number, and network protocol
- Security mechanisms and filters
- High availability companion server name

The LDAP server can be configured with these access restrictions:

- Anonymous authentication: All data is visible to any user.
- User name and password authentication: SAP ASE uses the default user name and password.

Installing and Maintaining User Databases
Before a user database can be created, the DBA needs to set up mirrored devices (Chapter 3) and make storage available to the server. This is performed using the disk init command, illustrated in Listing 1.

disk init
name = "device_name",
physname = "physical_name"
size = {number_of_blocks | {size K|M|G} }
[, vstart = virtual_address ,
ctrltype = controller_number ]
[, dsync = {true|false}]  
Listing 1 disk init Syntax

Detailed syntax and usage for disk init and other device-related commands are found in Chapter 4.

After the devices have been initialized, the create database command can be issued.

Note
Prior to SAP ASE 12.5, you needed to provide a virtual device number (vdevno), but with release 12.5, that is automatically assigned.

Listing 2 provides a quick example.

create database database_name
[on {default | database_device} [= size]
[, {default | database_device} [= size]}...
][log on {default | database_device} [= size]
[, {default | database_device} [= size]}...
[with {override | default_location = "pathname"}]  
[for {load | proxy_update}]  
Listing 2 create database Syntax

After the database is created, you can add tables, indexes, users, and permissions. Creation of databases is covered in more detail in Chapter 4, and recovery of the database is discussed in Chapter 5.

Establishing Security for SAP ASE

Security is critical for the protection of SAP ASE and the data contained within and is the subject of Chapter 6. Sensitive data must be protected from improper access, whether by reading or modification.

There are several levels of security within SAP ASE:

- At the server level, a login allows a connection to be established.
- At the database level, a user has access to the storage structures of data.
- At the object level, permissions allow the use of data creation and manipulation syntax.

It takes a combination of these security measures to successfully create a secure SAP ASE environment.

After you’ve put your chosen security scheme in place, the SAP ASE auditing feature may be enabled at the SA’s discretion, discussed in Chapter 7. Auditing allows you to monitor users, commands, and administrative tasks (but adds overhead to the server). Auditing must be installed and requires its own database (sybase) in which it retains auditing information. Reports can be generated from the auditing tables to track trends and help identify attempted security breaches.

Kerberos/Distributed Computing Environment (DCE)

In a distributed client-server computing environment, intruders can view or tamper with confidential data. SAP ASE works with third-party providers to give you security services that do the following:

- **Authenticate users, clients, and servers**
  Make sure they are who they say they are
- **Provide data confidentiality with encryption**
  Ensure that data can’t be read by an intruder
- **Provide data integrity**
  Prevent data tampering and detect when it has occurred

SSL

SAP ASE security services now support Secure Sockets Layer (SSL) session-based security. SSL is the standard for securing the transmission of sensitive information, such as credit card numbers, stock trades, and banking transactions, over the Internet.

The implementation of SAP ASE SSL features assumes that there is a knowledgeable system security officer who is familiar with the security policies and needs of your site, and who has a general understanding of SSL and public-key cryptography.

Security is covered completely in Chapter 6 and Chapter 15.

Controlling Data Flow

The DBA also is responsible for how data flows throughout a system, whether it’s standalone or part of a larger network. There are several tools and applications that will help you to accomplish various tasks:
- Set user limits, discussed in Chapter 9
- Prioritize how system resources are used, discussed in Chapter 10
- Partitioning data, discussed in Chapter 13
- Work with a remote server, discussed in Chapter 14
- Decide how queries are best received, discussed in Chapter 18

**System Performance**

What good is getting your system up and running if it doesn't work properly? The greatest part of the SA’s role may just be making sure that the system is running optimally. This book provides some guidelines on how to configure your memory optimally in Chapter 11, and how to cache your data in Chapter 12.

**Maintaining and Troubleshooting SAP ASE**

When the application databases are in place and users are allowed into the SAP ASE database, you still need to keep the server running. Logs will fill, data will outgrow its allotted space, indexes will become inefficient, and data can become corrupt. Many of these things can be detected early or avoided by general maintenance being performed on a scheduled basis. You should consider implementing the following maintenance processes:

- Run the database consistency checker (DBCC), which is a series of commands that detects structural problems within your system. Some of these are highly resource intensive and should only be run during slow times.
- Periodically rebuild indexes to improve performance. This is more important on highly volatile data (lots of inserts, updates, or deletes) than on static data. Maintaining statistics to improve query optimization.
- Add thresholds to warn when free space approaches a minimum you've defined.

These topics are discussed in Chapter 14 and Appendix B.

**Establishing Disaster Recovery Procedures**

Disasters happen. Whatever the nature of your disaster—environmental or otherwise—you must be prepared to move the complete business (or part of the business) operation to a new location and be up and ready in a short amount of time. A well-thought-out and thoroughly practiced plan must be available to deal with any contingency. Learn how to devise a good preventative maintenance plan in Chapter 15.

Some things to consider include the following:

- Spare machines with sufficient resources to run the business will be required.
- Networking needs must be addressed.
- Connectivity issues must be addressed and tested.
- Good backups of the databases will need to be accessible from off-site.
- Installation disks for all components (clients, servers, applications, etc.) must be accessible from off-site.
- Timetables for acceptable recovery time must be established.
- A step-by-step disaster recovery plan must be created.
- Personnel must be designated for disaster recovery.
- A failover site with warm or hot backups should be evaluated (determine whether it’s an option or mandatory for business survival).
- The plan must be tested. No matter how impressive it is on paper, it still may fall apart. Test it repeatedly.

There are books on disaster recovery and how to prepare for it. Each site will have different requirements and possibly different approaches to this subject matter. Just remember that it can happen to you, so be prepared. System and database backups and restoration is discussed in Chapter 8, and disaster recovery and high availability is discussed in Chapter 16.

**Monitoring SAP ASE**

There are many ways to monitor SAP ASE, including the following:

- **System stored procedures** (Chapter 7 and Chapter 16)
  These procedures present information about the state of the server.
- **Monitor and historical servers** (Appendix A)
  These servers obtain and track performance statistics.
- **Third-party applications** (Chapter 7, Section 7.5)
  These applications can be run with SAP ASE.
Whatever the tool or technique, monitoring is essential to the health of SAP ASE; knowing which resources are overused (or available in excess) allows you to maximize SAP ASE performance. Most mature shops have their own set of monitoring tools, which have been developed over time.

Use of Utilities
SAP ASE provides a set of utilities specifically for you, the administrator, to perform tasks for which you have sole responsibility and sole permission/rights. There are many tools discussed in this book; you can find a Sybase utilities are used at the OS level to perform special services against SAP ASE (discussed in Chapter 20):

- **srvbuild**
  A UNIX-based program to create the SAP ASE after it’s unloaded from CD.

- **srvconfig**
  A Windows NT-based program to create SAP ASE after it’s unloaded from CD.

- **dsedit**
  Edits the server entries in the `interfaces` file and LDAP servers.

- **bcp**
  The bulk copy program allows mass loading (or unloading) of data files into (or out of) tables.

- **defncopy**
  Creates script files containing the data definition language for specific database objects.

- **optdiag**
  Displays (and, in some cases, modifies) statistics pertaining to tables and the optimizer.

Databases
Within a server, a database provides the context for data. There are four types of databases stored in SAP ASE. The following system databases are essential to SAP ASE operation:

- **master database**
  Contains information needed to start the server, as well as all the system tables needed to govern SAP ASE. Anything that has to do with SAP ASE as a whole is stored in `master` (including permitted logins, existing databases, space allocation, and other broad-scope items). `master` is the first database to be brought online when you power up SAP ASE; therefore, it should be kept small and without user tables for faster access and startup of your SAP ASE.

- **model database**
  Used as a template when new databases are created, the contents of `model` are copied to each newly created database. Therefore, all changes that are made to `model` are reflected in new databases.

- **tempdb** is the temporary working area. Temporary tables are automatically created to hold intermediate results in some queries (especially when sorting is involved), and users may create temporary tables for their own purposes. Remember that the contents of `tempdb` aren’t permanent. `tempdb` is cleared and recreated each time SAP ASE is started, and some temporary tables will delete themselves even without a server restart. `tempdb` is heavily used by SAP ASE. If you experience performance bottlenecks, you can create more temporary databases.

- **sybsystemprocs** contains the stored procedures that are installed by the server (known as system procedures), which are used to query and change the contents of system tables.

- **sybsystemdb** is used for the data change processes known as distributed transactions (most transactions take place in a single database, but distributed transactions span multiple databases and sometimes multiple servers).

Special functionality:

- **dbccdb** is used to store output from the `dbcc checkstorage` command.

- **sybsecurity** is the auditing database, which must be installed prior to turning on the auditing options.

- **sybdiag** database is installed by Sybase Technical Support to help debug problems.

Sample databases
`pubs2` and `pubs3` are optional databases. They are the source of syntax examples in the documentation set and are used in many Sybase classes for labs. The installation scripts are provided with SAP ASE.

User databases
These are databases created specifically for applications.

The rest of this book is detail surrounding these tasks. Good luck!
Defining devices is a necessary step that comes before building tables and other database objects.

3 Defining Physical and SAP ASE-Mirrored Devices to the Server

Now that the server is installed, you need to create your user databases so that you can begin building your tables and other database objects. Before you can do that, you must first define the storage areas that the databases will require, which is described in this chapter.

Unlike some other database management systems, SAP ASE uses logical rather than physical devices to manage database space. A logical device is simply a name that’s mapped to a physical area on disk, which is identified by a file name. In the case of UNIX OSs, the file name can be either a raw device or operating system (OS) file (Windows NT also allows for raw devices, but NTFS is preferred). The differences between those two types of devices will be discussed later in this chapter in Section 3.3.1. Regardless of the type of physical device you choose, the syntax for creating and managing the logical device is the same.

Using logical devices provides you with more manageable and flexible control over your database and object placement because your database isn’t tied directly to a file. A single database can span multiple devices, and multiple databases can share the same device. Each database is made up of one or more fragments (contiguous allocations of physical storage), with each fragment mapped to part or all of a logical device.

Where you choose to place your databases and their individual fragments can play a major role in how they’ll perform. Make the most out of your available disk drives so that you don’t overwork one drive while another sits mostly idle. For that reason, you may want to create multiple logical devices across many physical drives so that when you create your database, it won’t reside entirely on one physical disk. Additionally, the type of device you choose to create can
Defining Physical and SAP ASE-Mirrored Devices to the Server

impact performance. Keep these facts in mind as you begin defining your devices. We’ll look at load balancing and device types in more detail in Section 3.3.2. We’ll also look at methods of protecting the databases from device failure and optimizing the performance of those database devices.

3.1 Creating and Dropping Devices

Managing your devices is handled through a series of commands and stored procedures. The device information and database mapping is stored in a set of system tables in the master database, which can be queried directly or through predefined stored procedures. Creating and modifying your devices should never be done by direct modification of the system tables; SAP ASE provides all the commands and procedures necessary to do this safely.

Don’t wait until you need database space to create your devices. You should have a set of devices predefined so that if your database runs out of space, you can quickly allocate more space to the database without having to scramble to find available disk space and define new logical devices. Additionally, choosing a standard set of sizes for your devices simplifies administration, record-keeping, and emergency repairs.

The master device and the device used to house the sybsystemprocs database (typically, syssprocsdev) are automatically created through the server installation process. The devices used to house the sybsecurity database are created through the auditinit utility. All other database devices are created using the disk init command. Each device created will be stored as a single row in the sysdevices system table in the master database, regardless of how the device was created.

Device Limitations and Restrictions

Following are the limitations and restrictions associated with database devices:

- **Logical device names**
  Must be unique (case sensitive), up to 255 characters (no spaces or punctuation).

- **Device sizes**
  System dependent, but generally 4 Terabytes (TB) without making changes to the OS.

- **Number of devices**
  This is capped at 2,147,483,648.

Vdevno
This is 1 through 2,147,483,647, with 0 reserved for the master device.

Dsync
Ignored for raw partitions and PC file systems; always on for the master device.

Location
Physical file must be local to the machine where SAP ASE resides (can’t be a network or remote mounted drive). You should fully qualify the path name, which must be 255 characters or fewer in length.

Tip
Remember to dump the master database after any allocation of system resources.

In the following sections, we’ll discuss the creation of devices, how to set up defaults, and how to drop devices.

3.1.1 Master Device Creation

The master device is the first device created at installation time. It used to be a strong recommendation to put the master device on a raw partition rather than an OS file, but with dsync, this is no longer a concern because writes are ensured.

After installation is complete, you’ll have four databases occupying space on the master device. Besides the master database, the other three are the model database, the initial fragment of your default tempdb, and sybsystemdb.

Note
If you’re upgrading from a pre-12.0 server, sybsystemdb is probably on a non-master device; if you’re installing 12.0 or 12.5, sybsystemdb resides on the master. You should keep it that way. To keep space free to allow for master database expansion, don’t add user databases to the master device. With that said, be sure to define the master device with enough space to allow for expansion of the master database. Typically, the master database gets larger with each new release of SAP ASE.

To build the master device, look at Listing 3.1.

```bash
dataserver -d master_device_name
(-e, -c, -T, -I, etc.)
[-b master_device_size [k|K|m|M|g|G|t|T]]
[-z logical_page_size [k|K]]
```

Listing 3.1 Building a Master Device
The –d option specifies the physical location and name of the master device to create.

The –b option specifies the number of virtual pages to allocate for the device. Virtual pages are always 2K in size, so this is similar to the `disk init` command where you multiply megabytes by 512 to determine the number of pages. Note that you can optionally specify the device size by specifying the size in kilobytes, megabytes, or gigabytes. You don’t have to use whole numbers when using megabytes or gigabytes; for instance, 210.5MB and 1.5GB are both valid sizes.

### 3.1.2 Raw Devices versus File System

We've mentioned raw devices and regular files but only from an operational standpoint. So, what about from a functional standpoint? The main difference between a raw device and an OS file isn’t in how you reference them or how you remove them, it's in how they are actually written to, and what it can mean to the safety of your database.

An OS file uses buffered writes. In general, this essentially means that as you're performing data modifications in your databases, SAP ASE writes these changes to the database devices on disk. Or, at least, it thinks it’s doing that. In reality, the OS tells SAP ASE that the writes have been completed, when in fact they have been buffered in cache and are only truly written to disk when a certain amount of write operations have occurred. If the server were to suddenly shut down in a less than graceful manner, you could end up with corruption in your index or page linkage, which can cause partial or full data loss in the affected table. This is not a good thing, obviously.

Raw devices on the other hand, don’t buffer their writes. Instead, when SAP ASE sends a write, the information is actually written to disk before SAP ASE receives confirmation of the write. As you can see, this is a much safer type of device to use, and it’s highly recommended for your system databases.

Safety isn’t the only difference between raw devices and regular files. Performance is also a consideration. Because regular files use buffered writes, and raw devices must actually write to disk, regular files tend to outperform raw devices. Depending on how heavily you’re writing to your database and how fast your disk drives are, this can turn out to be a significant difference in performance. However, this performance difference may vary widely (or even be negligible), depending on the hardware and OS of your host machine.

Manageability is another key difference. Managing raw devices tends to take a lot more effort and planning than using regular files and is more restrictive. If you use regular files, create your directories, or need a new device, just define one by pointing to that directory and assigning a new file name. With raw devices, you’re limited to how many slices you can have per disk drive, unless you’re using some volume management software, and you must plan ahead to determine how big each raw device should be.

It’s up to you to decide which is more important for each database and server. As with almost everything, there are trade-offs. Some databases are less critical than others. If corruption occurs, maybe restoring from yesterday's backup is acceptable, while slow performance isn’t. In that case, you might consider using regular files. You might choose to go with raw devices for production servers, while choosing the manageability of regular files for your development environments.
3.1.3 Create Devices

Now, we will discuss devices that should be created after the master. You'll use the code detailed in Listing 3.2 to create a device.

```plaintext
disk init
name = "device_name",
physname = { 'physical_name' | 'cache_name'},
[skip_alloc=true | false],
vdevno = virtual_device_number,
[. type = 'inmemory']
[. vstart = virtual_address
, cntrltype = controller_number]
[. , directio = {true | false}]
[, instance = "*instance_name"]
```

Listing 3.2 disk init Syntax

We’ll go over the main disk init parameters in the following subsections.

Name

Name is the logical name of the device, and it will be the name referred to when allocating database space to this physical device. It’s also the name you’ll reference when mirroring or assigning default devices. The name can be anything you like, but it must be unique to this server and can’t contain any spaces or punctuation. As is true with objects and databases, the name is case sensitive and must be 30 characters or less.

It’s a good idea to name devices to identify their use. For instance, creating devices called dev1, dev2, and dev3 is far less informative than sales_data1, products_data1, and sales_index1. If you were to run out of index space in the sales database, you can quickly identify which logical device to use to expand the database and extend your index segment.

physname

physname is the physical file that the logical name maps to and is the actual storage area that will house your database fragment. Depending on whether you’re using a raw device or OS file for the new device, the Sybase account must have read/write permissions on either the raw device itself or the directory that will contain the regular file. Without proper permissions, the disk init command will fail.

You may want to use symbolic links for physical device placement (on UNIX). This can be very useful for relocating physical files (to balance the load) when you add more disks later. In this example, we’ll only change the symbolic link, so nothing needs to be done within SAP ASE.

size

The third parameter is the size. By default, this is specified in number of virtual pages (the page size for the server), not megabytes. Note that size allocation isn’t an area that SAP ASE provides consistency; if you don’t remember the units when you’re using a command, be sure to look it up. A simple calculation to determine the number of 2K pages is to multiply the number of megabytes by 512.

Since release 12.5, you may instead choose to specify the units for disk init: ‘k’ or ‘K’ indicates kilobytes, ‘m’ or ‘M’ indicates megabytes, and ‘g’ or ‘G’ indicates gigabytes. Since release 15, you can also use ‘T’ or ‘t’ for terabytes. Although it’s optional, SAP recommends that you always include the unit specifier in both the disk init and create database commands to avoid confusion in the actual number of pages allocated. You must enclose the unit specifier in single or double quotes. Don’t use float; while you can specify a float, documentation differs as to whether any numbers after the decimal are truncated (e.g., if you specify size as ‘4.5G’, it’s initialized as ‘4G’), or are converted to the lower multiple of 2K pages. We recommend that you simply avoid the problem.

This chapter assumes 2K pages, which is the default (and the only size available prior to release 12.5). If you’re using larger page sizes, make sure you alter your
calculations accordingly. Devices have only one page size: 2K. You don’t begin to use the alternate sizes until you allocate the devices to databases.

You must be certain that the raw device isn’t smaller than the specified size, or in the case of regular files, that the directory has enough free space to create the new file.

**Tip**

Unlike regular files, where you can define many files under one directory, when using raw devices to define your logical devices, use the entire raw device size. Only one logical device can map to any given raw device, so if you don’t use it all, you’re wasting space. You may have a 2GB raw device, use the entire 2GB. Take this into account when working with your UNIX admin to carve up your physical disks into multiple raw device slices. If you want to define your devices as 500MB, then make sure your UNIX admin defines your raw devices as 500MB each.

**skip_alloc**

This is a Boolean parameter that’s supported for devices created on non-Windows file systems and for Windows raw systems. When **skip_alloc** is set to true, users avoid initialization of pages with zeros. The default of **skip_alloc** is false.

**vdevno**

This is the virtual device number, which is a unique identifier assigned to each device. It must be a number that isn’t already used by another device, and it must be less than the configured number of devices. If the number is 25, then **vdevno** can be no higher than 24. If you’ve used up all available device numbers, you must reconfigure for more devices using **sp_configure**. As of release 12.5, you can skip this parameter, and the server will automatically assign the next available virtual device number. This is recommended, as keeping track of available virtual device numbers can be tedious.

**Tip**

To determine the configured number of devices, run **sp_configure 'number of devices'**, or look for ‘number of devices’ in your configuration file (ServerName.cfg).

To determine which device numbers are currently in use, run **sp_helpdevice** or the following query.

```sql
select low/16777216
from sysdevices where cntrltype=0
order by low
```

On large servers where you have more than 127 devices, try this instead:

```sql
select low/power(2,24) + ((low/power(2,24)) & 128) * 2
from sysdevices where cntrltype=0
order by 1
```

**vstart**

Defining your virtual starting address and the controller number is optional and generally not used. The value defaults to 0. Don’t specify this unless directed to by SAP Technical Support.

**directio**

This parameter defines whether the **dsync** option is on or off for this device. This will be covered in detail in Chapter 16, Section 16.3.

The **directio** parameter for **disk init**, **disk reinit**, and **sp_deviceattr** allows you to configure SAP ASE to transfer data directly to disk, bypassing the OS buffer cache. **directio** performs IO in the same manner as raw devices and provides the same performance benefit as raw devices, but it has the ease of use and manageability of file system devices. **directio** is a static parameter that requires a restart of SAP ASE to take effect.

By default, the **directio** option is set to false (off) for all platforms except Linux, for which it’s set to true (on) by default.

The **directio** and **dsync** parameters are mutually exclusive. If a device has **dsync** set to true, you can’t set **directio** to true for this device. To enable **directio** for a device, you must first reset **dsync** to false.

**instance**

**instance = "instance_name"** is used only in clusters; it specifies the device as private and sets its owning instance to **instance_name**.
3.1.4 Create Devices: Examples

Let's take a look at some examples of logical device creation. Listing 3.3 shows how to create a logical device called sales_data1 on a raw device in a UNIX environment. Remember that the physical device '/dev/rdsk/c0t1d0s2' must have read/write access for the Sybase account, and the device size should be exactly the same as the size specified so that no space is wasted. In this case, the size is 100MB, or 51,200 2K pages. Also note the logical name sales_data1 must be unique.

```plaintext
disk init
name = sales_data1,
physname = '/dev/rdsk/c0t1d0s2',
size = 51200
```
Listing 3.3 Example 1

The second example shows how to create a logical device called tempdb_dev1 as an OS file with a size of "300M" (see Listing 3.4). In this example, we've used the '/tmp' directory to improve performance. This is mapped to the swap area, which is essentially the same as mapping it to memory so we get better performance than we would from a slow disk drive. Because '/tmp' is cleared out when the UNIX server is recycled, we'll want to make sure that we modify our Sybase startup script to check for the existence of the tempdb_devices directory and the tempdb_dev1.dat file prior to starting the server. If they don't exist, the script should create them, or the server won't start properly. If the tempdb database isn't available after startup, the server is unusable.

```plaintext
disk init
name = tempdb_dev1,
physname = '/tmp/tempdb_devices/tempdb_dev1.dat',
size = "300M"
```
Listing 3.4 Example 2

The last example (in Listing 3.5) shows how to create a logical device called sales_index1 on an OS file in a PC-based environment. Note that all syntax is the same, with the only difference being how you reference the physical file name.

```plaintext
disk init
name = sales_index1,
physname = 'D:\Database_Devices\Sales\sales_index1.dat',
size = "150M"
```
Listing 3.5 Example 3

3.1.5 Default Devices

After server installation, the master device is the only default device. This should be changed immediately so that you don't accidentally allocate user database space to the master device. Let's take a look at the syntax:

```plaintext
eexec sp_diskdefault logical_device_name, 'defaultoff | defaulton'
```

The master device should only contain system databases, not user databases. If the master database becomes full, and no space is left on the master device, the master database can only be allocated to the master device, which can be a big problem.

The example in Listing 3.6 illustrates how to take the master device out of the default device pool and add other devices to the pool.

```plaintext
eexec sp_diskdefault 'master', 'defaultoff'
eexec sp_diskdefault 'data_dev1', 'defaulton'
eexec sp_diskdefault 'data_dev2', 'defaulton'
```
Listing 3.6 sp_diskdefault Examples

Execution of this stored procedure will update the status column of the sysdevices table to identify whether the device is part of the default device pool. If you create or extend a database without specifying which device to use, space is allocated to one or more of the default devices, using them up in alphabetical order (by logical name).

Example

If you are creating a 500MB database, but data_dev1 only has 300MB free, it uses up that 300MB and then uses 200MB of the data_dev2 device, assuming space is available.
3.1.6 Dropping Database Devices

So, why would you want to remove a device? You may have made a mistake when you created it, such as defining the wrong size or location or name. Or perhaps you want to reallocate your device space to achieve better load balancing. Maybe you’ve reached the number of devices limit, so you need to combine smaller devices into larger ones. You may have had a disk drive fail, and after installing a new drive, you need to reallocate the device. Replacing a slower drive with a faster one is another good reason. Whatever your reason happens to be, SAP ASE provides a means for dropping database devices.

As mentioned previously, the device information is stored in the `sysdevices` system table, which is located in the master database. It’s okay to query this table, but just like you should never directly modify this table to add devices, the same holds true for removing devices. Instead, a stored procedure named `sp_dropdevice` should be used:

```sql
exec sp_dropdevice 'logical_device_name'
```

To remove the device, you specify the logical device name, not the physical name. When you execute this stored procedure, it removes the entry from the `sysdevices` table. It doesn’t remove the physical file, so you’ll need to do that manually to free up the directory space or to reuse that physical file name. Of course, you should only remove the physical file if it’s an OS file, not if it’s a raw device.

3.1.7 dsync Option

A nice feature that was added in version 12.0 was the `dsync` option. With this feature, you can combine the ease of use of regular files with the safety of raw devices. Basically, it eliminates the buffered behavior of regular files by telling UNIX to open the database device with the `dsync` flag. In other words, all writes are guaranteed, so the safety of your database is maintained. This isn’t to say you still can’t have corruption occur for other reasons, but the chances of it occurring due to missed writes is greatly reduced. On the other hand, it’s slower than using files with OS-supported buffering. There can be a real, very noticeable difference between OS files with `dsync` on or off. Note that this depends heavily on the storage type; flash storage, for example, will give you near-memory performance in storage.

Another benefit of using the `dsync` option is that although your writes will be slower, your reads will be faster than raw devices. This is because regular files typically use buffered reads in addition to buffered writes. So, in effect, you have a hybrid device that has the safety and slowness of raw device writes, but the read speed and flexibility of regular files.

### Note

Because the `master` database is so crucial to your server, `dsync` is always set to `true` if you define your master device to an OS file. You can’t change this; if you attempt to, you’ll receive a warning message.

Besides setting the `dsync` option at disk initialization, you can also change it after the device has already been created. This is done via the `sp_deviceattr` stored procedure shown here. Before the change takes effect, you must restart SAP ASE with this command:

```sql
exec sp_deviceattr 'logical_name', 'option_name', 'option_value'
```

**Example**

Use this changing DSYNC characteristic on a logical file:

```sql
exec sp_deviceattr 'sales_data1', 'dsync', 'true'
```

3.2 SAP ASE Mirroring

What happens when one of your disk drives fails? Unless you’ve done something to protect yourself, you’ll end up having to replace the failed drive, reinitialize database devices that were mapped to the drive, re-create one or more databases, and then restore the databases from backups. Depending on how you’re handling backups and how you’ve allocated your databases and transaction logs, you may end up losing data. Even if you’ve set things up so you don’t lose any data, you’re definitely going to have some downtime, which isn’t going to please your users or your management. It’s a good way to find yourself quickly hunting for a new job!

Thankfully, SAP ASE provides you with a method of protecting your database devices (and your job!). That method is called mirroring, and that’s what we’ll be exploring in the following sections. Mirroring basically means that each write to a primary device is subsequently written to a secondary device, so that you have a complete copy of your data. Of course, the secondary device should be defined on a completely separate physical disk drive, or it doesn’t do you much good. An
even better approach is to place the secondary device on a drive that uses a different controller, to further protect you in the event of a controller failure.

Note

Disk mirroring on your system can only be managed by your SAN, so this section may not be relevant for you.

3.2.1 Disk Mirror Syntax

The disk mirror syntax is similar to the disk init syntax because you must specify a logical and a physical device name. That's where the similarities end. Because it's a mirror of the database device, it doesn't require a size or vdevno value. It doesn't create a new entry in the sysdevices table, but instead modifies information about the logical device that you're mirroring.

Example

```bash
disk mirror
  name = 'logical_device_name',
  mirror = 'physical_mirror_name'
  [, writes = {serial | noserial}]
  [clear = {TRUE | FALSE}]
```

Let's examine the different main parameters in the syntax.

The first parameter is `name`, which represents the logical name of the device you want to mirror. It must refer to a database device that already exists.

The second parameter is `mirror`, which is the physical location of the secondary, or mirror, device. This follows the same rules as the `physname` parameter of the disk init command. The Sybase account must have read/write access to the location, it should be fully qualified with the path (unless you're using relative paths), the file can't already exist (unless it's a raw device), and there must be enough free space to create the file. The size of the mirror device will be exactly the same as the database device you're mirroring. If the device or directory is too small, the command will fail. Use the entire raw partition because any unused portion of that device will be unable to be referenced by any other program.

The third parameter is `writes`, which tells SAP ASE whether to use synchronous or asynchronous writes for the mirror device. This is similar to the difference between using regular files and raw devices. If you choose `noserial`, then as soon as the write to the primary side completes, processing continues, even if the mirror side hasn't completed its write. If the server were to go down suddenly, there's no guarantee that the mirror side is in synch with the primary side, which is a big problem. If you choose `serial writes`, then both sides must complete their writes before processing continues. This guarantees that the mirror and primary side are always in synch. As is the case with raw devices, there is a performance penalty to pay because you're now doubling your writes, so keep that in mind as you weigh your options.

The final parameter, `clear`, initializes the mirror device with zeros to guarantee that the underlying file system has reserved space for the mirror device. The default value, `FALSE`, doesn't clear the mirror, and executing a write to the device might fail due to lack of space on the file system. If you specify `TRUE`, the mirror is cleared, forcing the file system to reserve space for the device. `TRUE` will take longer, but it's safer.

Example

```bash
disk mirror
  name = 'data_dev1',
  mirror = '/sybase/mirror_devices/data_dev1_mir.dat',
  writes = noserial
```

When you run the disk mirror command, it may take a while to complete because it must copy all used pages from the primary database device to the mirror device. Be sure to run this during periods of little or no activity so the users aren't affected. After the pages are copied, it will update the row in the `sysdevices` table for the corresponding database device. This update will identify the physical location of the mirror device and update the `Status` column to identify it as mirrored and to note whether writes are serial or not.

Note

Reads are only done from the primary side.

3.2.2 Deciding What to Mirror

Keep in mind that because you're mirroring devices and not databases, you should make sure that if your database spans multiple database devices, you mirror them all; otherwise, you're only partially protecting your database. Depending on your goals and available space, you have several options as far as a mirroring strategy.
All Database Devices
If you want to ensure full nonstop access to the database devices, mirror all database devices. This means you'll be protecting your user databases and your system databases, and you'll be protecting your data and your transaction logs. If any of your disk drives fail, your server and all databases will still be accessible, and the users won't even know anything has happened. This safety does have a cost associated with it. You'll require twice as much disk space, but thankfully disk prices continue to drop. Perhaps the larger cost is the performance hit you'll take because, again, you’re doubling the number of writes.

Critical Databases
If you want to reduce the costs but still protect your data, you might consider only mirroring the truly critical databases. This depends on whether you can afford any downtime and how much effort will be involved in recovering your databases. You may have a database that is read-only. In this case, mirroring doesn’t cost you anything as far as write speed is concerned, but it does cost you disk space. Because the database is read-only, and maybe you can afford for that database to be unavailable for a while, you can leave it unmirrored and then just restore from your nightly backup if the device takes a hit. This leaves disk space available to mirror only your more mission-critical databases.

Transaction Log Devices
Another option is to only mirror the transaction log devices. As you’ll discover in later chapters, the transaction log is the only part of a database that is really needed to achieve up-to-the-minute recoverability. By using this mirroring strategy, you minimize your space requirements and performance hit by not having to mirror your data devices, while still ensuring no loss of data. The downside is that you’ll have some downtime while you replace your data devices and perform other recovery tasks.

Master Device
Of course, probably the most important database device to mirror is your master device. Without it, the entire server goes down, not just some of your user databases. In addition to mirroring the master device with the disk mirror command, another step is also required. Because SAP ASE reads the sysdevices table during startup to determine where all its mirror devices are, if the master device becomes damaged while the server is down, it can’t come back up because it can’t read the sysdevices table. To get around this problem, you must update the command line in the run_server file to identify the location of the mirror device, as shown here:

dataserver -d/sybase/devices/master.dat -r/sybase/mirrors/master_mir.dat -e.....etc.

If the primary side is damaged, SAP ASE will attempt to start from the mirror device. In theory, the server will figure this out anyway, but in practice it’s a good idea.

As you can see, there are a lot of decisions to be made, and every system has its own needs, so you’ll need to work out a mirroring strategy that works best for you.

3.2.3 Disable Mirroring
Mirroring may be disabled automatically, in case a device-level error is detected, or manually, in case you want to do something with the devices separately from the mirrors.

Automatic Disabling
If SAP ASE determines that there is an I/O error on one of the devices in a mirrored set, mirroring is disabled. At that point, all reads and writes are redirected to the device that is still functioning. This happens very quickly, and users likely won’t even notice. A message is written to the error log, and the Status column in the sysdevices table is modified to set the status to Disabled. You should manually inspect the error log frequently so that you can spot this error and correct it before your remaining device fails. Optionally, you can automate this process by writing a script that will scan the error log and notify you via email or page when mirroring is disabled.

Another option is to use a mirrexit process via the waitfor command, illustrated in the example below.

```sql
Example
waitfor mirrexit
begin
exec sp_sendmail 'dba_group','Mirroring error detected, please see errorlog for details'
end
```
Note that the `sp_sendmail` stored procedure isn't a system procedure; it's just an example of a process that you might write to handle sending email. You can have the process do anything you like, but generally, you want it to perform some action that will cause you to be notified of the failed device. The only trick to this method is that it's not persistent. In other words, you must execute this command each time SAP ASE is restarted.

**Manual Disabling**

At certain times, you might want to intentionally disable mirroring of a particular database device. This can be accomplished with the `disk unmirror` command, which has the same effect as automatic disabling (see Listing 3.7). All reads and writes will now be against the remaining device in the mirror set, and `sysdevices` will be updated to reflect the change. It does run much faster than the disk mirror command because all it's really doing is updating a single row in `sysdevices`.

```
disk unmirror
  name = 'logical_device_name'
  [ ,side = {'primary' | 'secondary'}]
  [ ,mode = {'retain' | 'remove'}]
```

Listing 3.7 Disk Unmirror Syntax

The first parameter is `name`, which identifies the logical database device you want to unmirror. It's the only required parameter. The second parameter is `side`, which specifies whether you want to stop writing to the secondary (mirror) side, or the primary side. The default value is `secondary`.

The last parameter is `mode`. The `retain` value causes a temporary disabling, meaning you can re-enable mirroring with the `disk remirror` command, which we'll cover later. It resets the status bits in `sysdevices` to indicate that mirroring is disabled, but it retains the physical name of the secondary device. When you use `remove`, it updates the status bits to show no mirroring and removes the physical name of the secondary device. To remirror, you need to use `disk mirror`, not `disk remirror` because there is no record of any previous mirroring on that device.

<table>
<thead>
<tr>
<th>Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>You might find a need to relocate a logical device to a faster disk or to improve load balancing. One way to do this is to drop all databases that reside on the affected devices, drop the devices, re-create the devices on different physical disks, and then re-create and reload the databases from backup. This requires a lot of work and a lot of downtime. Instead, you can simply use mirroring commands as in the following example:</td>
</tr>
</tbody>
</table>

```
disk mirror
  name = 'data_dev1',
  mirror = '/sybase/mirrors/data_dev1_mir.dat'

disk unmirror
  name = 'data_dev1',
  side = 'primary',
  mode = 'remove'
```

First, `data_dev1` is copied to a mirror location. The primary side is then dropped, making `/sybase/mirrors/data_dev1_mir.dat` be the new physical location of `data_dev1`, and then removing any record of mirroring. It's just copying and repointing, and it can be done while users are on the system (with some impact, of course).

As mentioned earlier, after you've unmirrored a device with the `retain` option, you can go back at a later point and remirror. This essentially synchs up the mirror side with the primary side, but it performs a complete copy of the primary device to the mirror side, not just changes that occurred because mirroring was disabled. The main advantage remirror has over mirror is speed; it's that you don't have to specify a physical name or any other options.

The disk remirror syntax is the following:

```
Disk remirror
  Name = 'unmirrored_logical_device_name'
```

### 3.2.4 Software- and Hardware-Level Mirroring

While SAP ASE provides very effective mirroring capabilities, it does involve a lot of planning and setup, and it can be slow compared to storage-level hardware mirroring, which almost always offers better performance. In this section, we'll discuss hardware- and software-level mirroring, as well as the redundant array of inexpensive/independent drives (RAID).

<table>
<thead>
<tr>
<th>Tip</th>
</tr>
</thead>
<tbody>
<tr>
<td>If you have a choice, we recommend hardware mirroring for performance.</td>
</tr>
</tbody>
</table>

#### Hardware-Level Mirroring

Because hardware mirroring is handled by the controller, it reduces the CPU cost that SAP ASE and the OS would otherwise incur. In other words, the CPUs can
spend their time doing real work instead of managing the disk, which means it’s typically faster than software mirroring. It also protects the areas of the disk that SAP ASE mirroring can’t, such as SAP ASE’s software and other file systems. This type of mirroring tends to be very reliable and better performing, but it’s also more expensive.

**Software-Level Mirroring**

SAP ASE’s mirroring is software-level mirroring. Windows NT provides disk striping and mirroring capabilities, which is also software-level mirroring. There are many types, but the one thing they all have in common is that software is managing the disk mirroring, which can be expensive in terms of CPU cycles. In its favor, software mirroring allows you to fragment your physical disks so that you can pick and choose which parts of the disk to mirror. This provides you with more control but also more administration by having to plan, set up, monitor, and adjust as needed.

### 3.2.5 RAID

Redundant Array of Inexpensive Drives, or RAID (some say independent drives, which is also true), has many different configurations (levels), but the most common are 0, 1, 5, and 10 (not available on all platforms, but it’s our current favorite). RAID incorporates mirroring in some cases, striping in other cases, and in still others, both mirroring and striping. RAID uses hash information stored on one or more disks to “mirror” the information stored as regular data. If one of the devices fails, the data is “re-created” from the hash information. This should be transparent to users.

Without going into detail on RAID, mirroring protects your data by keeping a “copy” of it in another location, and striping helps balance your I/O load across available disks to improve performance. With striping, blocks of data are written to successive devices instead of to just one. So, if you’re writing 64KB of data to disk, it might write 16KB to each of four disks if you were using a 16KB stripe size. This can greatly improve read and write performance. You may want to use different RAID levels for your data devices than you do for your log devices to achieve protection where you want it and performance where you need it. The bottom line is, investigate the various RAID levels, and then choose the one that works best for you.

### 3.3 Volume Management

Volume management is the practice of using software to enhance the usability of your devices. It can help improve load balancing, reduce maintenance, and provide you with much more usable devices. Without going into great detail, this section is intended to introduce you to the concept. Talk to your OS administrator to determine what’s available on your system.

#### 3.3.1 Maintenance and Ease of Use

In many UNIX environments, you’re limited in how many raw device partitions (typically six) you can have on a single disk drive. With limited disk drives, you can only have so many SAP ASE logical devices defined. It’s much simpler with file system devices because you can create many logical devices under one directory.

Volume management allows you to define logical raw partitions, greatly increasing the number of raw partitions you can define, which means more flexibility in creating your SAP ASE logical devices. If you have 4 drives, you’re no longer limited to having 24 partitions; you may have 50, 100, or more. Unused raw partitions can be recarved to better suit your needs as they evolve.

#### 3.3.2 Load Balancing

Volume management software typically includes not only a way of “carving” your physical disks with more control than without, it also tends to include striping capabilities. With it, you can create a single logical raw partition that might span many physical raw partitions across many disks. The logical raw partition provides you with the same safety of the underlying physical raw partitions; that is, the writes aren’t buffered. Striping them across many physical disks is one way of offsetting the performance cost associated with choosing raw instead of file system devices, while maintaining peace of mind.

#### System Information

The following are some stored procedures that give you useful information about your physical environment:
Defining Physical and SAP ASE-Mirrored Devices to the Server

- `sp_helpdevice ['logical_device_name']`
  Returns a listing of all database devices and their attributes, or a single specified device.

- `sp_dropdevice 'logical_device_name'`
  Removes the device entry from `sysdevices`. The device can't be occupied by any database, or the procedure will fail.

- `sp_diskdefault 'logical_device_name’, {'defaulton' | 'defaultoff'}`
  Adds or removes a logical device from the default device pool.

- `sp_devicemattr 'logical_device_name', 'option', 'setting'`
  Changes an attribute of a logical device.

- `sp_configure 'number of devices'`
  Displays or changes the configuration value for the number of logical devices allowed.

- `sp_helpdb 'database_name'`
  Displays basic database information, which fragments reside on which logical devices, and each fragment's use.

Let's look at a few examples.

In Listing 3.8, we have output from `sp_helpdevice` on `test_dev1`, showing the physical name and options selected.

<table>
<thead>
<tr>
<th>device_name</th>
<th>physical_name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_dev1</td>
<td>/sybase/devices/test_dev1.dat</td>
<td>special, MIRROR DISABLED, mirror = '', nonserial writes, physical disk, 2048.00</td>
</tr>
</tbody>
</table>

Listing 3.8 `sp_helpdevice 'test_dev1'`

We have output from `sp_dropdevice` on `test_dev1` with Device dropped.

You can change the default on the device to "off" with return status = 0. Note that you don't have a lot of output here.

In Listing 3.9, we have output from `sp_configure`, showing the number of configured devices.

<table>
<thead>
<tr>
<th>Parameter Name</th>
<th>Default</th>
<th>Memory Used</th>
<th>Config Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of devices</td>
<td>10</td>
<td>#10</td>
<td>10</td>
</tr>
</tbody>
</table>

(1 row affected, return status = 0)

Listing 3.9 `sp_configure 'number of devices'`

In Listing 3.10, we have output from `sp_helpdb`. This shows the database attributes, as well as the storage it's using on individual devices.

<table>
<thead>
<tr>
<th>name</th>
<th>db_size</th>
<th>owner</th>
<th>dbid</th>
</tr>
</thead>
<tbody>
<tr>
<td>db1</td>
<td>48.0 MB</td>
<td>sa</td>
<td>4</td>
</tr>
</tbody>
</table>

created 4 Sep 14, 2005
status no options set

<table>
<thead>
<tr>
<th>device_fragments</th>
<th>size</th>
<th>usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_dev</td>
<td>28.0 MB</td>
<td>data only</td>
</tr>
<tr>
<td>log_dev</td>
<td>20.0 MB</td>
<td>log only</td>
</tr>
</tbody>
</table>

created 14 2005 11:06AM 19008
release status = 18896

Listing 3.10 `sp_helpdb db1`

System Tables

You have three options for system tables:
Defining Physical and SAP ASE-Mirrored Devices to the Server

### Modifying System Tables

Unless absolutely necessary, you should never directly modify system tables. The results could be disastrous. If you want to add a new device, use `disk init`; don’t insert a row manually into `sysdevices`. If you want to drop a device, run `sp_dropdevice`; don’t manually delete the row. If you want to enable mirroring, don’t update the row; run `disk mirror` or `disk remirror`. In other words, you have all the commands and procedures you need to manage your disk resources without having to directly modify the system tables.

There are occasions where you may need to update system tables. But as far as the `sysdevices` table is concerned, there is no reason to modify it directly, unless directed to by SAP Technical Support.

### Disaster Recovery

In the event of a disaster, such as the loss of an entire machine (or building!), you’ll want to get the system back up and running as quickly as possible. Some important system information will be required to make it quicker and easier to accomplish. Although not required, you’ll probably want to create your devices and databases with the same or similar layout on your physical disks. After you’ve identified a suitable replacement for your machine, you’ll want to quickly install SAP ASE and begin creating your devices and databases so you can restore from backup. Therefore, it’s very important that you keep a record of or better yet, an extraction of, your system table information.

An easy way to do this is to `bcp` out your `sysdevices`, `sysusages`, and `sysdatabases` tables, and then copy the `bcp` files to tape. (See a discussion of `bcp` in Chapter 20.) This should be done routinely, depending on how often the system information changes. In addition, running the `sp_helpdb` and `sp_helpdevices` procedures and saving off the reports can help speed and verify the recovery process. The reports can be run via `iSQL` and the output saved to disk, so they too can be copied off to tape.

### For Geeks Like Us

The more you work with SAP ASE, the better acquainted you’ll become with system tables. System tables contain metadata (information about how the information is stored). You’ve seen a few queries that identify specific information from the system tables; the following sections provide a more in-depth study of what’s happening inside.

#### Sysdevices

The definition of a device is stored in the system table `sysdevices`. The columns in this table record how a device was created, and how it’s configured. Several of the columns in `sysdevices` can be directly interpreted based on the `disk init` command that created the device; in this example, the `name` and `physname` columns correspond to the `name` and `physname` arguments of the creation script (see Table 3.1).

**Note**

Unlike the ‘disk init’ argument, the column is spelled p-h-y-n-a-m-e (no “s”).
jumped. You were looking at 28 (on the right) and 27 (on the left), and all of a sudden, you find yourself between 100 (right) and 101 (left).

To avoid confusion when different builders added buildings to Fifth Avenue, each block was pre-assigned a range of street numbers to use: the first block gets to use 1–99 and 2–98; the second block is assigned 101–199 and 100–198, and so forth. We could put up as many buildings as we want on either side of the street, as long as the numbers increase as we go north, and we don’t run outside of our range; if we don’t use all of the assigned numbers (as in this case), the unused range is wasted. At the same time, we’re limited to how many buildings we can put up on one side of the block by the number of available addresses (although it’s unlikely that we could fit more than 50 buildings on one side of the block).

But the alternative is trying to coordinate all of the builders up and down Fifth Avenue, hoping for perfect communication, and praying that no two of them decide to use the same street address. In this case, wasting some potentially usable street numbers seems better than the risk of putting up two buildings with the same number.

SAP ASE does the same thing with devices, only the preallocated ranges aren’t in ranges of 100; SAP ASE uses a 3-byte address to locate pages on a device, which allows up to 2^24 = 16,777,216 pages per device. Each device is pre-assigned a unique range of 2^24 page numbers; any numbers not actually used (because most devices won’t be large enough to require the full range of page numbers) are wasted.

This information is stored in the columns low and high, which represent (respectively) the first and last page numbers assigned to the device. The value of low is calculated by multiplying the vdevno used to create the device by 2^24; because each device is only allowed a maximum of 2^24 pages, this allows for non-overlapping ranges of page numbers for each device.

The value of high for any device is derived from low and the size argument of ‘disk init’; each device has size pages, starting from low and ending on high.

Because the page range is inclusive, note the following:

\[
\text{high} = \text{low} + \text{size} - 1
\]

\[
\text{size} = \text{high} - \text{low} + 1
\]
For the sample `sysdevices` given previously, we can derive the following information:

```plaintext
master: low = 0  high = 25599
vdevno = low/2^24  size = high - low + 1
  = 0 / 2^24  = 25599 - 0 + 1
  = 0  = 25600 virtual pages
sysprocsdev: low = 16777216  high = 16828415
vdevno = low/2^24  size = high - low + 1
  = 16777216/2^24  = 16828415 - 16777216 + 1
  = 1  = 51200 virtual pages
```

While most users will find these calculations sufficient, those with large numbers of devices will find one other behavior in the calculation of `low`. At large values of `vdevno` (128 – 255 inclusive), `low` and `high` suddenly jump to very large negative numbers. This happens because SAP ASE has no way to represent unsigned numbers. Table 3.2 shows some sample values from `sysdevices`.

<table>
<thead>
<tr>
<th>low</th>
<th>high</th>
<th>status</th>
<th>cntrltype</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2147483648</td>
<td>-2147483137</td>
<td>16386</td>
<td>0</td>
<td>testdev1</td>
</tr>
<tr>
<td>-2130706432</td>
<td>-2130705921</td>
<td>16386</td>
<td>0</td>
<td>testdev2</td>
</tr>
<tr>
<td>2113929216</td>
<td>-2113928705</td>
<td>16386</td>
<td>0</td>
<td>testdev3</td>
</tr>
</tbody>
</table>

Table 3.2 sysdevices Values

Given these numbers and the previously shown formulas, we can calculate the following:

```plaintext
testdev1  vdevno = -128  size = 512
testdev2 vdevno = -127  size = 512
testdev3 vdevno = -126  size = 512
```

The `low` and `high` columns use an integer datatype to store their values, which has a maximum value of 2,147,483,647, which is enough to store all of the possible pages for devices with `vdevno` values 0 to 127, but no higher. But the integer datatype is signed and also allows storage of negative numbers down to −2,147,483,648, which is enough space to store another 128 devices. Because SAP ASE will allow up to 256 devices per server to be configured, this requires that negative values be used in `low` and `high`.

For `vdevno` values in the range 128 – 255 (inclusive):

```plaintext
low = (vdevno - 256) * 2^24
vdevno = (low / 2^24) + 256
```

For our three devices:

```plaintext
testdev1: low = -2147483648  high = -2147483137
vdevno = (low/2^24) + 256  size = high - low + 1
  = 128  = 512 pages

testdev2: low = -2130706432  high = -2130705921
vdevno = (low/2^24) + 256  size = high - low + 1
  = 129  = 512 pages

testdev3: low = -2113929216  high = -2113928705
vdevno = (low/2^24) + 256  size = high - low + 1
  = 130  = 512 pages
```

The value of `vdevno` for any device can be calculated with the following:

```sql
select convert(int,(low/power(2,24))) & 255
```

from `sysdevices` order by 1

**Status, Cntrltype, and Mirrorname**

The `status` column is an integer number that describes the presence or absence of other attributes of a device. About 15 different characteristics, each true or false, could be used in combination to describe your device. For example:

- **This is a default device (1)**
- **This device is mirrored (64)**
- **This device has the dsync flag enabled (16384)**

If the characteristic is true for a device, then the binary value in parentheses is added to the `status` column; if false, a zero is added instead. If all of the possible characteristics for a device are false, then the `status` value would be 0. If only the ‘default device’ characteristic were true, then the `status` value would be 1. If the device were dsync enabled, mirrored, and a default device, then the `status` value would be 16449 (1 + 64 + 16384). This `bitmap` value allows for a large number of combinations of characteristics to be stored in very little space. The `status` column can be translated into verbal descriptions with the `sp_helpdevice` procedure, and a full list of `status` values can be found in the reference manual.
The \textit{cntrltype} column is used to distinguish between \textit{database devices} (every device discussed in this chapter) and \textit{dump devices} (used solely for storing backups and are discussed in Chapter 7). For reference, all dump devices will have \textit{cntrltype} values greater than 0, and \textit{status} values of 16. Dump devices will also have values of low and high that don't follow the rules given previously and can't be mirrored.

The \textit{mirrorname} column is used to store the physical name of the device's mirror; if no mirror is defined for the device, \textit{mirrorname} is \texttt{NULL}.

\textbf{Creating Devices for In-Memory and Relaxed-Durability Databases}

Later in this book (see Chapter 19), we'll be talking about in-memory databases (IMDB). Relaxed-durability databases are a subset of IMDB.

For completeness in this chapter, we're including information on this specific database type. There are a variety of considerations to take into account when creating devices for in-memory and relaxed durability databases:

- The logical name of the in-memory device can't be the same as the name of the cache on which a device is created. Caches will also be discussed later in Chapter 12. In general, it's the memory you're setting aside for reducing I/O, which is what IMDB is all about in the first place.
- In-memory devices are reserved for the first IMDB you assign to it.
- The physical device name must be the logical name of the in-memory device.
- The logical name of the in-memory device must be unique across all devices.

You can't do the following:

- Use physical device types of parameters when you create an in-memory device: \texttt{vstart, cntrltype, dsync, directio, and skip_alloc}.
- Create an in-memory device that is larger than the cache on which it is created.
- Create an in-memory device from a regular named cache, including the default data cache. Instead, create it from a newly defined cache (drop the other one if you need the space).
- Increase the size of an in-memory device after it's created. However, you can use \texttt{sp_cacheconfig} to increase the size of the in-memory cache, and you can use \texttt{disk init} to create new in-memory devices.
- Use these commands against in-memory devices:

\begin{itemize}
  \item \texttt{disk resize}
  \item \texttt{disk mirror}
  \item \texttt{diskremirror}
  \item \texttt{disk unmirror}
  \item \texttt{disk refit}
  \item \texttt{disk reinit}
\end{itemize}

Use the \texttt{sp_deviceattr} and \texttt{sp_diskdefault} system procedures on disk devices.

\section*{3.4 Summary}

How you choose to implement your device strategy is an important step in setting up and modifying your database environment. You need to choose which types of devices to use for each database, whether to use default disks, what to mirror, and how you want to implement mirroring. You can decide to implement load balancing with disk striping or with user-defined segments, and you can choose whether or not to use volume management software. Regardless of what you choose, we've learned that it's always better to do the planning up front than to have to rework it all later.
Maintaining your SAP ASE system for optimal performance can stop trouble before it starts.

15 Preventative Maintenance Regimen

Like a piece of machinery, SAP ASE must be properly maintained to stay in good working order. In this chapter, we’ll discuss some common tasks that are used to maintain the server and ensure that it stays in working order.

The different types of maintenance will be described in levels based on scope. Some tasks take place at the server level, meaning that your tasks involve resources that affect the entire server; for example, memory is a shared resource among all the tasks running on SAP ASE. Other tasks take place at the database level. The database is the server’s unit of backup and recovery, so you perform database backups (making a copy of the database) and other types of care and maintenance one database at a time. Finally, at the table level (the table is what contains your data), you do things to help ensure database performance; that is, fast access to the data.

In this chapter, we’ll break down the different types of maintenance into three categories: server level, database level, and table level.

15.1 Server-Level Maintenance

The word “server” is used in a variety of ways in IT literature, from a file server (which manages disks) to server in SAP ASE (which interfaces with each client to manage your data). Here, we focus on SAP ASE as a single unit and consider issues and monitors for the server as a whole.

Part of systems administration is disaster preparedness. Under some circumstances, you may need to reinstall your entire server. Given this, it becomes necessary to include scripts that can be used to rebuild the entire server as part of the server-level maintenance routine. This need not be a daily issue, but there are plenty of shops
that extract this information from system tables on a nightly basis (via a scripted scheduled job such as a cron job) against just such a potential calamity.

SAP ASE comes with various tools to monitor the activity on the server. Regular monitoring of the server can prevent situations where the system starts to perform badly because of misconfiguration, or the monitoring can be used to identify any bottlenecks in the server. The act of monitoring the server’s performance on a repeated basis throughout a day can also be used to identify peak times for the server’s use.

We’ll discuss the commands you need and the procedures you’ll follow in the following sections.

15.1.1 System Use Information

sp_sysmon produces low-level system utilization information. This data is often more useful for identifying bottlenecks or periods of intensive server use. The sp_sysmon stored procedure does take into account the existence of multiple processors. However, when the utilization percentages are combined, it’s possible to see overall utilizations of greater than 100%.

Use of sp_sysmon contributes some overhead, but it’s difficult to measure. The procedure should be run before and after changing system parameters, or adding or removing caches, disks, or engines within the server. This will allow the administrator to gather a before and after picture of any changes to the server and their effect on performance. Running the procedure before and after the installation of new applications to the server can help identify the impact the new application has on the overall server. And, of course, this should be run periodically during peak times to determine the load on your systems. If you have a lot of processing you want to understand, and the processing runs 30 minutes, go ahead and run sp_sysmon for 30 minutes to track the performance.

Note

SAP instructs that sp_sysmon can’t be executed on a server with a default tempdb size and recommends that the tempdb be at least 2MB larger than the default.

The procedure actually returns a lot of information that is organized into various sections, as described in Table 15.1. The name of the section can be used to force the procedure to return information only about the section called.

<table>
<thead>
<tr>
<th>Section</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Management</td>
<td>appmgmt</td>
<td>Reports on user tasks</td>
</tr>
<tr>
<td>Cache Wizard</td>
<td>cache wizard</td>
<td>Reports on data caches</td>
</tr>
<tr>
<td>Data Cache Management</td>
<td>dcache</td>
<td>Reports on all data caches</td>
</tr>
<tr>
<td>Disk I/O Management</td>
<td>diskio</td>
<td>Reports on all disk I/O</td>
</tr>
<tr>
<td>ESP Management</td>
<td>esp</td>
<td>Reports on extended stored procedures</td>
</tr>
<tr>
<td>Housekeeper Task Activity</td>
<td>housekeeper</td>
<td>Reports on the housekeeper process</td>
</tr>
<tr>
<td>Index Management</td>
<td>indexmgmt</td>
<td>Reports on index management</td>
</tr>
<tr>
<td>Kernel Utilization</td>
<td>kernel</td>
<td>Reports how busy the kernel was</td>
</tr>
<tr>
<td>Lock Management</td>
<td>locks</td>
<td>Reports on locks, deadlocks, and lock promotions</td>
</tr>
<tr>
<td>Memory Management</td>
<td>memory</td>
<td>Reports on pages allocated and deallocated</td>
</tr>
<tr>
<td>Metadata Cache Management</td>
<td>mdcache</td>
<td>Reports on the metadata cache</td>
</tr>
<tr>
<td>Monitor Access to Executing SQL</td>
<td>monaccess</td>
<td>Reports on show plans and monitors server access to query plans</td>
</tr>
<tr>
<td>Network I/O Management</td>
<td>netio</td>
<td>Reports on network I/O</td>
</tr>
<tr>
<td>Parallel Query Management</td>
<td>parallel</td>
<td>Reports on parallel queries</td>
</tr>
<tr>
<td>Procedure Cache Management</td>
<td>pcache</td>
<td>Reports on the procedure cache</td>
</tr>
<tr>
<td>Recovery Management</td>
<td>recovery</td>
<td>Reports on the number and length of checkpoints</td>
</tr>
<tr>
<td>RepAgent</td>
<td>repagent</td>
<td>Reports on the replication</td>
</tr>
<tr>
<td>Task Management</td>
<td>taskmgmt</td>
<td>Reports on open connections</td>
</tr>
<tr>
<td>Transaction Management</td>
<td>xactmgmt</td>
<td>Reports on transaction management activities</td>
</tr>
<tr>
<td>Transaction Profile</td>
<td>xactsum</td>
<td>Reports on transaction-related activities</td>
</tr>
<tr>
<td>Worker Process Management</td>
<td>wpm</td>
<td>Reports on the activity of worker processes</td>
</tr>
</tbody>
</table>

Table 15.1  sp_sysmon Returned Information
An example of \texttt{sp\_sysmon} syntax is given here:
\begin{verbatim}
exec sp_sysmon "HH:MM:SS" [, section_name]
\end{verbatim}

Let's look at a sample of the \texttt{sp\_sysmon} output in Listing 15.1.

\begin{verbatim}
Server Version: Adaptive Server Enterprise/16.0 GA PL01/
EBF 22544 SMP/P/
x86_64/Enterprise Linux/ase160sp00pl01/3523/64-
bit/FBD/T
we Apr 15 13:24:31 2014
Run Date: Oct 03, 2014
Sampling Started at: Oct 03, 2014 17:17:25
Sampling Ended at: Oct 03, 2014 17:17:42
Sample Interval: 00:00:17
Sample Mode: No Clear
Counters Last Cleared: Oct 02, 2014 22:06:16
Server Name: sap_ase_1

Kernel Utilization

Engine Utilization (Tick %) User Busy System Busy I/O Busy Idle
------------------------- ------------ ------------ ---------- --
ThreadPool : syb_default_pool
Engine 0 14.8 % 38.6 % 17.6 %
29.0 %
\end{verbatim}

\textbf{Listing 15.1 Partial sp\_sysmon Output}

The output of \texttt{sp\_sysmon} is extensive, and learning to read and interpret all of its results requires a great deal of practice. Administration can be as much art as science. There may be many sets of correct values for the server configuration parameters. Issues can be found by looking at the percentages to see where ASE is spending its time and effort. For example, looking at the Kernel Utilization section can determine the extent of CPU Utilization (Engines). The percentages can provide insight into configuration settings or cache settings that may be over- or under-utilized.

The goal is to schedule \texttt{sp\_sysmon} regularly throughout the day. When an issue occurs, the hope is to have \texttt{sp\_sysmon} collecting performance data while the issue happens. Afterwards, the output from \texttt{sp\_sysmon} can be used to compare against other \texttt{sp\_sysmon} data to determine what may be causing the issue.

\section*{What to Monitor}

When using the \texttt{sp\_sysmon} output, keep an archive of the outputs, and try to at least gather information about Engine Busy Utilization (in the Kernel Utilization section), and Disk I/O Management. Try to gauge maximum, minimum, and average values for these readings during peak and off-peak times. This will provide a frame of reference for what is “normal” on the server and will make abnormalities easier to identify. Also, try to gather information on User/Process connections (Worker Process Management and Task Management sections). Metrics such as the total number of connections at a particular time of day, average connections per user, and the average length of time blocked by other users (some of which you’ll have to calculate yourself) can help gauge the normal work load attributed by users to the server.

As an alternative, SAP ASE also comes with a monitor server that collects real-time data on the servers and can store that data to assist in discovering bottlenecks. You may write your own client tools to display and interpret these data, or you may use the monitor clients of SAP Control Center (SCC).

There are also many good third-party tools that can be used to monitor SAP ASE.

\subsection*{15.1.2 MDA Tables}

SAP ASE also has memory-only tables that can be used to monitor performance, users, and a variety of other tasks. There are over 90 of these MDA tables that are easily identifiable tables as they all begin with “mon” \textit{Memory only} means that the tables don’t actually take any actual disk storage. When the server is shut down, SAP ASE does not retain any information of the data stored in these tables. In fact, many of these tables are so volatile, the act of executing a query against the table will clear its data. If the same query were executed immediately afterwards, the data would be different. Due to the highly active nature of these tables, it’s recommended that the MDA tables be inserted into physical tables so that the data being examined is not lost.
Many DBAs will develop their own processes to extract and store data from these tables. Many others will use third party performance monitoring tools which are designed to query these tables and record the data to repository databases.

A few examples of MDA tables are:
- monCIPC
- monCIPCEndpoints
- monCIPCLinks
- monCIPCMesh
- monCLMObjectActivity
- monCMSFailover
- monCachePool

### 15.1.3 Locking Contention Monitoring

SAP ASE should also be monitored on a regular basis for locking contention, which occurs when multiple users try to lock the same data as other users. This is considered a form of bottleneck, and while unavoidable, it can be minimized through efficient query writing and proper configuration of the locking scheme for tables.

**sp_lock**

The `sp_lock` stored procedure will list the various locks at a point in time on the SAP ASE. The output will include the type of lock and the SPID (server process ID of the process) of the user that is being blocked by the lock (if any).

The `sp_lock` output shows what process (spid) has a lock on what object ID, and it will always list at least one lock (sp_lock itself locks data while running).

`sp_lock` is often used in conjunction with `sp_who` to diagnose locking contention; `sp_who` is used to identify the running processes on the server. For each connection (and certain background system processes), it will provide information that includes the login name, any blocking locks, and what kind of task was being performed when the procedure was run.

An example of `sp_lock` syntax is given here:
Now let’s switch our attention to sp_who, which has the following syntax:

```
sp_who [loginname | "spid"]
```

- **page** is the page being locked if the lock type is a page lock.
- **row** is the row being locked if the lock type is a row lock.
- **dbname** is the name of the database in which the item being locked resides.
- The **class** column indicates whether a lock is associated with a cursor. It displays one of the following:
  - **Non Cursor Lock** indicates that the lock isn’t associated with a cursor.
  - **Cursor Id number** indicates that the lock is associated with the cursor ID number for that SAP ASE process ID.
- A cursor name indicates that the lock is associated with the cursor **cursor_name** that is owned by the current user executing sp_lock.
- The **context** column identifies the context of the lock. Worker processes in the same family have the same context value. Legal values for **context** are as follows:
  - **NULL** means that the task holding this lock is either a query executing serially or is a query executing in parallel in transaction isolation level 1.
  - Sync pt duration request means that the lock will hold the lock until the query is complete. A lock’s context may be Sync pt duration request if the lock is a table lock held as part of a parallel query, if the lock is held by a worker process at transaction isolation level 3, or if the lock is held by a worker process in a parallel query and must be held for the duration of the transaction.
- **Ind pg** indicates locks on index pages (all pages locked [APL] tables only).
- **Inf key** indicates an infinity key lock (for certain range queries at transaction isolation level 3 on data only locked [DOL] tables).
- **Range** indicates a range lock (for range queries at transaction isolation level 3 on DOL tables). These new values may appear in combination with Fam dur (which replaces Sync pt duration) and with each other, as applicable.

### Listing 15.2 sp_lock Output

| fid | spid | dbname | object_name | context | sync_point_duration | non_cursor_lock | cursor_id_number | cursor_name | locked_page | locked_row | locked_class | locked_table | locked_page | locked_row | locked_class | locked_table |
|-----|------|--------|-------------|---------|---------------------|-----------------|-----------------|-------------|-------------|------------|--------------|--------------|--------------|-------------|-------------|--------------|--------------|
| 0   | 16   | NULL   | NULL        | 32      | NULL                | Non Cursor Lock | NULL            | NULL        | NULL        | NULL       | NULL         | NULL         | NULL         | NULL        | NULL        | NULL         | NULL         |

The `sp_lock` output is ordered by **fid** (family ID) and then **spid**. The **fid** column identifies the family (including the coordinating process and its worker processes) to which a lock belongs. Values for **fid** are given here:

- A zero value indicates that the task represented by the SPID is executed serially. It isn’t participating in parallel execution.
- A nonzero value indicates that the task (spid) holding the lock is a member of a family of processes (identified by **fid**) executing a statement in parallel. If the value is equal to the SPID, it indicates that the task is the coordinating process in a family executing a query in parallel.

**SPID** is the server process ID of the process and has the following parameters:

- The **oid** (lock owner ID) column identifies the unique lock owner ID of the blocking transaction. Even **oid** values indicate that a local transaction owns the lock. Odd numbered values indicate that an external transaction owns the lock. Use the object_name system function to derive a table’s name from its ID number.
- The **locktype** column indicates whether the lock is a shared lock (Sh prefix), an exclusive lock (Ex prefix), or an update lock, and whether the lock is held on a table (table or intent), on a page (page), or on a row (row). A **blk** suffix in the **locktype** column indicates that this process is blocking another process that needs to acquire a lock. As soon as this process completes, the other processes moves forward. A **demand** suffix in the **locktype** column indicates that the process is attempting to acquire an exclusive lock.
- You can use the object_name() built-in function to see the name of the object for the ID displayed, or, you can modify the sp_lock stored procedure to display the name. Better yet, you can write your own version of the procedure.
The loginname or spid value (which must be quoted) will retrieve information about a single process; otherwise, the procedure will list all current processes.

Now let’s look at an example of the output in Listing 15.3.

```sql
1> sp_who
2> go

<table>
<thead>
<tr>
<th>fid</th>
<th>spid</th>
<th>status</th>
<th>loginame</th>
<th>origname</th>
<th>hostname</th>
<th>blk_xloid</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>sleeping</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Listing 15.3 Output of sp_lock

The references to `loid` and `xloid` (unique lock owner IDs) only apply to distributed transactions between databases or between servers. If you aren’t running that sort of processing, the loid/xloid values should all be zero. Likewise, the references to `fid` are only meaningful when running parallel query processing; in such cases, a user connection and internal parallel query worker processes are working together on one query when they share a value of `fid`.

### 15.1.4 System Parameter Monitoring

You can use the `sp_monitorconfig` stored procedure to keep track of the utilization of many of the system parameters that you’ve tuned, as you can see in Listing 15.4.

```sql
1> sp_monitorconfig 'all'
2> go

Usage information at date and time: Oct 3 2014 11:03PM.

<table>
<thead>
<tr>
<th>Name</th>
<th>Num_free</th>
<th>Num_active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pct_act</td>
<td>0</td>
<td>157</td>
</tr>
<tr>
<td>Max_Used</td>
<td>100.00</td>
<td>1576592</td>
</tr>
<tr>
<td>Reuse_cnt</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Instance_Name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>additional network memory</td>
<td>0</td>
<td>157</td>
</tr>
<tr>
<td>audit queue size</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Preventative Maintenance Regimen

Server-Level Maintenance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value 1</th>
<th>Value 2</th>
<th>Percent</th>
<th>Value 3</th>
<th>Value 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression Info Pool Size</td>
<td>4096</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>Disk I/O Structures</td>
<td>1024</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Heap Memory Per User</td>
<td>12266</td>
<td>NULL</td>
<td>0</td>
<td>0.18</td>
<td>22</td>
</tr>
<tr>
<td>Kernel Resource Memory</td>
<td>3961</td>
<td>NULL</td>
<td>0</td>
<td>38.07</td>
<td>2435</td>
</tr>
<tr>
<td>Max CIS Remote Connection</td>
<td>100</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>Max Memory</td>
<td>2269462</td>
<td>80</td>
<td>0</td>
<td>26.12</td>
<td>80253B</td>
</tr>
<tr>
<td>Max Number Network Listen</td>
<td>4</td>
<td>NULL</td>
<td>0</td>
<td>20.00</td>
<td>1</td>
</tr>
<tr>
<td>Max Online Engines</td>
<td>0</td>
<td>NULL</td>
<td>0</td>
<td>100.00</td>
<td>1</td>
</tr>
<tr>
<td>Memory Per Worker Process</td>
<td>748</td>
<td>NULL</td>
<td>0</td>
<td>26.95</td>
<td>276</td>
</tr>
<tr>
<td>Number of Alarms</td>
<td>27</td>
<td>NULL</td>
<td>0</td>
<td>32.50</td>
<td>14</td>
</tr>
<tr>
<td>Number of Aux Scan Descriptors</td>
<td>256</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>Number of Devices</td>
<td>94</td>
<td>NULL</td>
<td>0</td>
<td>6.00</td>
<td>6</td>
</tr>
<tr>
<td>Number of DTX Participants</td>
<td>500</td>
<td>NULL</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of Java Sockets</td>
<td>40</td>
<td>NULL</td>
<td>0</td>
<td>4.76</td>
<td>2</td>
</tr>
<tr>
<td>Number of Large I/O Buffers</td>
<td>50</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>2</td>
</tr>
<tr>
<td>Number of Locks</td>
<td>9898</td>
<td>NULL</td>
<td>0</td>
<td>1.02</td>
<td>1524</td>
</tr>
<tr>
<td>Number of Mailboxes</td>
<td>18</td>
<td>NULL</td>
<td>0</td>
<td>40.00</td>
<td>12</td>
</tr>
<tr>
<td>Number of Messages</td>
<td>64</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>Number of Open Databases</td>
<td>12</td>
<td>NULL</td>
<td>0</td>
<td>40.00</td>
<td>8</td>
</tr>
<tr>
<td>Number of Open Indexes</td>
<td>529</td>
<td>NULL</td>
<td>0</td>
<td>3.82</td>
<td>36</td>
</tr>
<tr>
<td>Number of Open Objects</td>
<td>481</td>
<td>NULL</td>
<td>0</td>
<td>12.55</td>
<td>84</td>
</tr>
<tr>
<td>Number of Open Partitions</td>
<td>529</td>
<td>NULL</td>
<td>0</td>
<td>3.82</td>
<td>36</td>
</tr>
<tr>
<td>Number of Remote Connects</td>
<td>20</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>1</td>
</tr>
<tr>
<td>Number of Remote Logins</td>
<td>20</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>Number of Remote Sites</td>
<td>10</td>
<td>NULL</td>
<td>0</td>
<td>0.00</td>
<td>1</td>
</tr>
</tbody>
</table>
This output allows a look at configurable parameters for the server, what the configured values are, what the current values are, and whether or not the limits were reached since the server was started.

### 15.1.5 Monitoring the System Error Log

By default, SAP ASE will place an error log file in the directory in which you started the server (usually the install directory). In addition to boot time messages identifying which resources came up, their success, and the amount of elapsed time, any major errors or anomalies from the server will be recorded in the error log; this can include anything from abnormal disconnections from the server to major system halting errors.

Some third-party tools will notify you of additions to the error log. Many shops write tasks that monitor the error log on a continual (say, once per minute) basis. Note that SQL error messages will list a value of severity that will indicate the relative importance of the error (the higher, the more severe). Severity levels 11–18 are classed as user errors, and severities of 19+ are classed as system errors. Severity level 10 is reserved for informational messages.

#### What to Look For

Monitor the server for valid startup. Any errors that occur during startup can be an indication of a misconfiguration or a hardware error. Any errors that occur during the start of the server may prevent a database from coming online or, in a worst case, prevent the server from starting.

Look for any database threshold errors. SAP ASE monitors thresholds for every database that doesn’t set the no free space acctng dboption (which should probably never be set on a production database). If a threshold is crossed (either the last-change threshold in the log, or any other user-defined threshold), the associated threshold procedure will be executed. A notification about crossing the threshold and the output of any print messages from within the threshold process will be added to the error log. (For more on thresholds and threshold procedures, see Chapter 5.)

Listing 15.5 provides an error log example, and we’ve also provided some note boxes for commentary.
Preventative Maintenance Regimen

Note

Here we can see what version the server is running.

00:00:00.0000:0000:0000:2014/10/03 23:08:24.45 kernel Confidental property of SAP AG or an SAP affiliate company. Use, duplication or disclosure of the software and
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Documentation by the U.S. Government is subject to restrictions set forth
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel In a license agreement between the Goverment and SAP AG or an SAP affiliate
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel or other written agreement specifying the Government's rights to
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel use the software and any applicable FAR
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel provisions, for example, FAR 52.227-19.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel SAP AG or an SAP affiliate company
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Using /opt/sap as the 'SYBASE' environment variable, found during startup.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Using OCS-16_0 as the 'SYBASE_OCS' environment variable, found during startup.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Using ASE-16_0 as the 'SYBASE_ASE' environment variable, found during startup.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel ASE booted on host 'atcrhel' running Linux release 2.6.32-431.23.3.el6.x86_64 version #1 SMP Wed Jul 16 06:12:23 EDT 2014.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Using '/opt/sap/ASE-16_0/sap_ase_1.cfg' for configuration information.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Logging ASE messages in file '/opt/sap/ASE-16_0/install/sap_ase_1.log'.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Disk Controller Manager is online.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Network Controller Manager is online.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Ct-1b Controller Manager is online.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Platform TCP network support seems IPv6-aware. IPv4/IPv6 are both enabled.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel ASE booted with TCP_NODELAY enabled.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Dynamic Pluggable Component Interface is disabled.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Network and device connection limit is 4081.
00:00:00.0000:0000:2014/10/03 23:08:24.45 kernel Adaptive Server is running as process id 7249.
Server-Level Maintenance

15.1

Preventative Maintenance Regimen

00:00:00.0000:0000:2014/10/03 23:08:24.59 server Number of blocks left for proc headers: 268704.
00:00:00.0000:0000:2014/10/03 23:08:24.59 server Proc header memory allocated 134350 pages for each engine cache.
00:00:00.0000:0000:2014/10/03 23:08:24.83 server Begin processing to generate RSA keypair.
00:00:00.0000:0000:2014/10/03 23:08:24.90 server Completed processing to generate RSA keypair.
00:00:00.0000:0000:2014/10/03 23:08:24.96 server Loaded encryption provider CSI-2.9M10-linuxamd64-2014/04/08 20:42:03 PDT-sybcsi_openssl-OpenSSL 1.0.1g-fips 7 Apr 2014.

Note

This is a useful message if, for example, you've turned on trace flags to detect deadlocks.
Preventative Maintenance Regimen

00:00:06:0000:00001:2014/10/03 23:08:25.21 server Started ANALYSIS pass for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.21 server Completed ANALYSIS pass for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.21 server Log contains all committed transactions until 2014/10/03 23:07:32.89 for database master.
00:00:06:0000:00001:2014/10/03 23:08:25.21 server Started REDO pass for database 'master'. The total number of log records to process is 1.
00:00:06:0000:00001:2014/10/03 23:08:25.21 server Completed REDO pass for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.42 server Timestamp for database 'master' is (0x0000, 0x000127f0).
00:00:06:0000:00001:2014/10/03 23:08:25.42 server Recovery of database 'master' will undo incomplete nested top actions.
00:00:06:0000:00001:2014/10/03 23:08:25.42 server Started recovery checkpoint for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.44 server Completed recovery checkpoint for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.47 server Started filling free space info for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.47 server Completed filling free space info for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.47 server Started cleaning up the default data cache for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.47 server Completed cleaning up the default data cache for database 'master'.
00:00:06:0000:00001:2014/10/03 23:08:25.47 server Checking external objects.
00:00:06:0000:00001:2014/10/03 23:08:25.50 server Database 'master' is now online.
00:00:06:0000:00001:2014/10/03 23:08:25.50 server Database 'master' is now online.
00:00:06:0000:00001:2014/10/03 23:08:25.57 server The transaction log in the database 'master' will use I/O size of 8 KB.
00:00:06:0000:00001:2014/10/03 23:08:25.59 server server name is 'sap_ase_1'

Note
The server brought up the master database and the physical devices. Next, we'll recover each of the other databases. Remember, you can choose the order in which the databases recover using the sp_dbrecovery_order stored procedure.
03 23:08:26.08 server Completed estimating recovery log boundaries for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.08 server Started ANALYSIS pass for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.08 server Completed ANALYSIS pass for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
00:0006:00000:00001:2014/10/
03 23:08:26.08 server Started REDO pass for database 'sybsystemdb'.
The total number of log records to process is 1.
00:0006:00000:00001:2014/10/
03 23:08:26.09 server Completed REDO pass for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.18 server Timestamp for database 'sybsystemdb' is (0x0000, 0x000015b1).
00:0006:00000:00001:2014/10/
03 23:08:26.18 server Recovery of database 'sybsystemdb' will undo incomplete nested top actions.
00:0006:00000:00001:2014/10/
03 23:08:26.19 server Started recovery checkpoint for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.20 server Completed recovery checkpoint for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.20 server Started filling free space info for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.20 server Completed filling free space info for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.20 server Started cleaning up the default data cache for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.20 server Completed cleaning up the default data cache for database 'sybsystemdb'.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Checking external objects.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server The transaction log in the database 'sybsystemdb' will use I/O size of 8 Kb.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Transaction coordinator initialized.
00:0006:00000:00001:2014/10/
03 23:08:26.25 server Resident Node id: f8f8d0e936d1
00:0006:00000:00001:2014/10/
03 23:08:26.27 server Recovering database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.27 server Started estimating recovery log boundaries for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Database 'model', checkpoint=(931, 5), first=(931, 5), last=(931, 5).
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Completed estimating recovery log boundaries for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Started ANALYSIS pass for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Completed ANALYSIS pass for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Log contains all committed transactions until 2014/08/01 23:50:15.24 for database model.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Started REDO pass for database 'model'.
The total number of log records to process is 1.
00:0006:00000:00001:2014/10/
03 23:08:26.28 server Completed REDO pass for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.31 server Timestamp for database 'model' is (0x0000, 0x000014f6).
00:0006:00000:00001:2014/10/
03 23:08:26.31 server Recovery of database 'model' will undo incomplete nested top actions.
00:0006:00000:00001:2014/10/
03 23:08:26.31 server Started recovery checkpoint for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.32 server Completed recovery checkpoint for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.32 server Started filling free space info for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.32 server Completed filling free space info for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.32 server Started cleaning up the default data cache for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.32 server Completed cleaning up the default data cache for database 'model'.
00:0006:00000:00001:2014/10/
03 23:08:26.33 server Checking external objects.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.33 server The transaction log in the database 'model' will use I/O size of 8 Kb.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.33 server Database 'model' is now online.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.33 server The logical pagesize of the server is 4 Kb.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.33 server 0 dump conditions detected at boot time
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.33 server Clearing temporary database 'tempdb'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.66 server Tempdb 2 is added to the list of local temporary databases
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.71 server Timestamp for database 'tempdb' is (0x0000, 0x000014f6).
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.71 server The transaction log in the database 'tempdb' will use I/O size of 4 Kb.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.71 server Database 'tempdb' is now online.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.73 server Recovering database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.73 server Started estimating recovery log boundaries for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.73 server Timestamp for database 'sybsystemprocs' is (0x0000, 0x0005161d).
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.73 server Recovery of database 'sybsystemprocs' will undo incomplete nested top actions.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.77 server Started recovery checkpoint for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.77 server Completed recovery checkpoint for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.77 server Started filling free space info for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:26.79 server Completed filling free space info for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.75 server Timestamp for database 'sybsystemprocs' is (0x0000, 0x00040146).
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.75 server Recovery of database 'sybsystemprocs' will undo incomplete nested top actions.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.77 server Started recovery checkpoint for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.77 server Completed recovery checkpoint for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.77 server Started filling free space info for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.79 server Completed filling free space info for database 'sybsystemprocs'.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.80 server Checking external objects.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.81 server The transaction log in the database 'sybsystemprocs' will use I/O size of 8 Kb.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.81 server Database 'sybsystemprocs' is now online.
00:00:06:0000:0000:0001:2014/10/
03 23:08:28.82 server The wash size of the 4K buffer pool in cache default data cache has been changed from 61440 Kb to 49764 Kb due to a change in the size of the pool.
00:00:06:0000:0000:0001:2014/10/
03 23:08:29.11 server The wash size of the 4K buffer pool in cache default data cache has been changed from 61440 Kb to 49764 Kb due to a change in the size of the pool.
00:00:06:0000:0000:0001:2014/10/
03 23:08:29.11 server The wash size of the 4K buffer pool in cache default data cache has been changed from 61440 Kb to 49764 Kb due to a change in the size of the pool.
00:00:06:0000:0000:0001:2014/10/
03 23:08:29.12 server Recovery has tuned the size of '32K' pool in 'default data cache' to benefit recovery performance. The original configuration will be restored at the end of recovery.
00:00:06:0000:0000:0001:2014/10/
03 23:08:29.12 server Recovery has tuned the size of '4K' pool in 'default data cache' to benefit recovery performance. The original configuration will be restored at the end of recovery.
00:00:06:0000:0000:0001:2014/10/
03 23:08:29.12 server Recovery has tuned the '32K' pool in 'default data cache' by changing its 'local async prefetch limit' from 10 to 80. The original configuration will be restored at the end of recovery.
Preventative Maintenance Regimen

Server-Level Maintenance 15.1

00:00:06:0000:00001:2014/10/03 23:08:29.12 server Recovery has tuned the '4K' pool in 'default data cache' by changing its 'local async prefetch limit' from 10 to 80. The original configuration will be restored at the end of recovery.

00:00:06:0000:00001:2014/10/03 23:08:29.12 server The server will recover databases serially.

00:00:06:0000:00001:2014/10/03 23:08:29.14 server Recovering database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.14 server Started estimating recovery log boundaries for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Database 'pubs2', checkpoint=(1332, 32), first=(1332, 32), last=(1332, 32).

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Completed estimating recovery log boundaries for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Started ANALYSIS pass for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Completed ANALYSIS pass for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Log contains all committed transactions until 2014/08/01 23:56:18.80 for database pubs2.

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Started REDO pass for database 'pubs2'. The total number of log records to process is 1.

00:00:06:0000:00001:2014/10/03 23:08:29.16 server Completed REDO pass for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.21 server Timestamp for database 'pubs2' is (0x0000, 0x00002389).

00:00:06:0000:00001:2014/10/03 23:08:29.21 server Recovery of database 'pubs2' will undo incomplete nested top actions.

00:00:06:0000:00001:2014/10/03 23:08:29.21 server Started recovery checkpoint for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.21 server Started filling free space info for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.22 server Completed filling free space info for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.22 server Started cleaning up the default data cache for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.23 server Completed cleaning up the default data cache for database 'pubs2'.

00:00:06:0000:00001:2014/10/03 23:08:29.23 serverChecking external objects.

00:00:06:0000:00001:2014/10/03 23:08:29.24 server The transaction log in the database 'pubs2' will use I/O size of 8 Kb.

00:00:06:0000:00001:2014/10/03 23:08:29.24 server Database 'pubs2' is now online.

00:00:06:0000:00001:2014/10/03 23:08:29.26 server Recovering database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.26 server Started estimating recovery log boundaries for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.26 server Started estimating recovery log boundaries for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.26 server Started estimating recovery log boundaries for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Database 'pubs3', checkpoint=(1263, 12), first=(1263, 12), last=(1263, 12).

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Completed estimating recovery log boundaries for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Started ANALYSIS pass for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Completed ANALYSIS pass for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Log contains all committed transactions until 2014/08/01 23:56:18.82 for database pubs3.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Started REDO pass for database 'pubs3'. The total number of log records to process is 1.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Completed REDO pass for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Timestamp for database 'pubs3' is (0x0000, 0x000023a6).

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Recovery of database 'pubs3' will undo incomplete nested top actions.

00:00:06:0000:00001:2014/10/03 23:08:29.27 server Started recovery checkpoint for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.28 server Completed recovery checkpoint for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.30 server Timestamp for database 'pubs3' is (0x0000, 0x000023a6).

00:00:06:0000:00001:2014/10/03 23:08:29.30 server Recovery of database 'pubs3' will undo incomplete nested top actions.

00:00:06:0000:00001:2014/10/03 23:08:29.31 server Started recovery checkpoint for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.31 server Completed recovery checkpoint for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.31 server Started filling free space info for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.32 server Started filling free space info for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.32 server Started filling free space info for database 'pubs3'.

00:00:06:0000:00001:2014/10/03 23:08:29.32 server Started filling free space info for database 'pubs3'.
Preventative Maintenance Regimen

Server-Level Maintenance

Listing 15.5 Error Log Example, with Commentary

03:23:08:29.37 server Completed recovery checkpoint for database 'payroll'.
00:00:06:0000:00001:2014/10/
03:23:08:29.38 server Started filling free space info for database 'payroll'.
00:00:06:0000:00001:2014/10/
03:23:08:29.39 server Completed filling free space info for database 'payroll'.
00:00:06:0000:00001:2014/10/
03:23:08:29.39 server Started cleaning up the default data cache for database 'payroll'.
00:00:06:0000:00001:2014/10/
03:23:08:29.39 server Completed cleaning up the default data cache for database 'payroll'.
00:00:06:0000:00001:2014/10/
03:23:08:29.39 server Checking external objects.
00:00:06:0000:00001:2014/10/
03:23:08:29.39 server The transaction log in the database 'payroll' will use I/O size of 8 Kb.
00:00:06:0000:00001:2014/10/
03:23:08:29.41 server Database 'payroll' is now online.
00:00:06:0000:00001:2014/10/
03:23:08:29.41 server Recovery has restored the value of 'local async prefetch limit' for '32K' pool in 'default data cache' from '80' to 'DEFAULT'.
00:00:06:0000:00001:2014/10/
03:23:08:29.41 server Recovery has restored the value of 'local async prefetch limit' for '4K' pool in 'default data cache' from '80' to 'DEFAULT'.
00:00:06:0000:00001:2014/10/
03:23:08:29.42 server Recovery has restored the original size for '32K' pool and '4K' pool in 'default data cache'.
00:00:06:0000:00001:2014/10/
03:23:08:29.43 server ASE's default unicode sort order is 'binary'.
00:00:06:0000:00001:2014/10/
03:23:08:29.43 server ASE's default sort order is: 'bin_iso_1' (ID = 50)
00:00:06:0000:00001:2014/10/
03:23:08:29.43 server on top of default character set: 'iso_1' (ID = 1).
00:00:06:0000:00001:2014/10/
03:23:08:29.43 server Master device size: 65 megabytes, or 33280 virtual pages. (A virtual page is 2048 bytes.)
When you see this message, the server is up.

Only the startup messages are shown in this example. Following the copyright information, we find messages concerning the following:

- Allocation of memory
- Opening of master device and recovery of master database
- Opening of the remaining devices
- Recovery of system databases sybsystemdb and model
- Drop and re-creation of tempdb
- Recovery of sybsystemprocs
- Recovery of user databases
- Confirmation of server sort order and character set

Any errors in this section should be considered serious; errors after the startup section concern conditions during runtime and may or may not be serious.

Managing the System Error Log

The error log starts and stops with the server; if the server is up for a long period of time, the error log can grow quite large. If left unmanaged, the error log can grow large enough to fill up an entire file system. On some systems, if the error log can’t grow any more, SAP ASE will crash.

If your OS will permit it (some don’t), simply move the error log periodically (weekly, bi-weekly, etc.) to another location. If not, for example, on Windows NT 4, it’s recommended that administrators periodically (weekly, bi-weekly) shut down SAP ASE and move the error log to another location.

15.1.6 Resource Verification

Another good practice for administrators is to verify that all resources in the server are allocated properly and that all configurable parameters are optimally configured. SAP ASE allocates memory for most resources and parameters at startup, and most of the server options require some amount of memory to maintain. If any resources are over-allocated, then memory that might be better used in cache by the SAP ASE is essentially wasted. If any options or parameters are under-allocated, then SAP ASE may not perform as well. Check resources allocated versus resources in use. Are any configured parameters drastically out of range? For example:

- Are the concurrent connections at 100, with 25 the most used at any one time?
- Are the device connections at 50, with only 5 devices defined?
- Is remote access defined but not in use?
- Are all of your resources online?
- Did all of the disks start up?
- Are all of the engines online?

15.1.7 Software Maintenance

SAP ASE maintains its software with releases and point releases called software rollups (SWRs; previously called EBFs on older releases of the server). These software updates are available to supported users via download.

A common approach to dealing with SWRs is the following:

1. Wait a couple of weeks or months before installing the upgrade. On very rare occasions, a fix will come out immediately after the SWR release.
2. Install it on the development server, and test it there for a few weeks. This is often common sense in most shops. Before anything is implemented in production, the changes should be implemented in a development environment and rigorously tested to make sure the new upgrade fixes the identified problems and also to make sure the changes don’t expose any new problems.
3. Finally, after testing on your development server, install it on your production server. Be prepared to back off your upgrade at a moment’s notice. It’s always advisable to hope for the best and prepare for the worst when dealing with software upgrades. Should the installation of a software update go badly, the worst-case scenario is that the database server may need to be rebuilt. Be prepared to fall back to the last-known working version of the server should the upgrade process fail.

The preceding approach is considered appropriately conservative. Many shops generally adhere to balancing the following two axioms regarding software implementation:
If it ain’t broke, don’t fix it.
The various SWR releases usually come with a list of problems that have been repaired. Often, the majority of the list may not apply to the applications running on your servers. Introducing the latest SWR may cause the applications to behave differently enough that major modifications may be called for.

If you put off upgrades long enough, eventually SAP won’t support you.
While patience is considered a virtue, waiting too long may invalidate your service contracts. The goal is to not wait too long.

15.1.8 Recording Runtime Data
The \texttt{sp\_configure} stored procedure will list all system configuration variables that are easily set by administrators. These configuration settings are also stored in a file under the \texttt{SYBASE} directory called \texttt{<servername>.cfg}; if multiple servers are running, there will be multiple configuration files. SAP ASE also archives the file when changes are made; these backup files are named \texttt{<servername>.bak} (current version) and a series of numbered older files \texttt{<servername>.###} (historical configurations). Always keep off-site copies (hard and soft) of the configuration file. The copy can be used to start a restored server without having to manually reconfigure the server a parameter at a time. It’s also a good idea to archive a good copy of the results from a run of \texttt{sp\_configure}. If there are any problems with the configuration file, this allows the appropriate settings to be entered manually.

15.2 Database-Level Maintenance
SAP ASE has a series of routines called \texttt{dbcc} or \texttt{Database Consistency Checker}. These were originally unpublished routines used to verify the supporting structures for the data stored in the database. These structures, if corrupted, can prevent the database from functioning, and if left unrepaired, they can potentially lead to a fatal corruption of the database. If the corruption is copied into a database backup, the backup and future restorations from that backup will also be corrupt.

We use these routines as periodic maintenance to confirm that the database’s internal integrity is consistent. It’s very unusual to find an error here, but if you do find one, you should correct it quickly.
Contents

Acknowledgments ........................................................................................................... 23
Preface ............................................................................................................................ 25
Introduction to SAP ASE System Administration ................................................... 27

1 Introduction to SAP ASE 16 .............................................................................. 37
  1.1 Placement within the SAP Landscape ............................................................ 38
  1.2 Architecture Overview ............................................................................ 38
  1.3 SAP ASE 16: Key Features ...................................................................... 41
    1.3.1 Increased Speed and Scalability ...................................................... 41
    1.3.2 Security and Auditing ..................................................................... 45
    1.3.3 Simplicity .......................................................................................... 46
  1.4 SAP ASE Key Capabilities ........................................................................ 50
    1.4.1 SAP ASE Cluster Edition ................................................................ 50
    1.4.2 Partitioning ...................................................................................... 51
    1.4.3 Compression .................................................................................... 52
    1.4.4 Data Federation (via Component Integration Services) .......... 52
    1.4.5 Graphical Monitoring and Administration ..................................... 53
    1.4.6 Encryption ...................................................................................... 53
    1.4.7 Replication ....................................................................................... 54
    1.4.8 ASE In-Memory Database ............................................................... 55
  1.5 Summary ......................................................................................................... 55

2 Installation and Connectivity .............................................................................. 57
  2.1 Preparation .................................................................................................... 58
    2.1.1 Logical Page Size ............................................................................. 59
    2.1.2 Physical Devices ............................................................................. 60
    2.1.3 Server Names .................................................................................... 61
    2.1.4 Networking Information .................................................................. 61
    2.1.5 Sybase Software Asset Manager ................................................... 61
  2.2 File Transfer ................................................................................................ 62
  2.3 Running Installation ..................................................................................... 62
    2.3.1 Device Installation .......................................................................... 63
    2.3.2 Database Configuration ................................................................... 70
    2.3.3 Server Startup ................................................................................ 70
  2.4 Installation Files ............................................................................................ 73
    2.4.1 Interfaces File .................................................................................. 73
6.1.3 Database-Level Security ................................................... 176
6.2 Standard Roles ................................................................. 181
6.2.1 The System Administrator Role ...................................... 182
6.2.2 The sysusers Table ......................................................... 183
6.2.3 The Operator Role ........................................................... 185
6.2.4 The System Security Officer Role .................................... 186
6.3 Server Roles ................................................................. 188
6.3.1 System Role Definitions ................................................... 189
6.3.2 System Role Functions .................................................... 189
6.3.3 syslogins, sysloginroles, and syssrvroles ............................ 190
6.3.4 Turning System Roles Off ................................................. 190
6.4 User-Defined Roles .......................................................... 191
6.4.1 Creating Roles ............................................................... 191
6.4.2 Activating Roles ............................................................. 192
6.4.3 Role Hierarchy ............................................................... 192
6.4.4 Assigning Roles to Users ................................................... 192
6.4.5 Dropping Roles .............................................................. 193
6.4.6 Mutually Exclusive Roles ............................................... 193
6.4.7 Assigning Passwords to Roles .......................................... 193
6.4.8 Default Roles for Logins ................................................. 194
6.4.9 Granting Permissions to Roles .......................................... 195
6.4.10 Displaying Information about Roles ................................. 195
6.4.11 Show Active Roles ......................................................... 196
6.4.12 Display Permissions ....................................................... 197
6.4.13 Groups ........................................................................... 197
6.5 Login Activities ............................................................... 199
6.5.1 Dropping Logins ............................................................ 199
6.5.2 The syslogins Table ........................................................ 200
6.5.3 Show Login Information .................................................. 200
6.5.4 Change Existing Login Information ................................. 201
6.6 Commands ................................................................. 202
6.6.1 Display Server Connections .............................................. 202
6.6.2 Granting the Set Proxy Command .................................... 203
6.6.3 The Kill Command .......................................................... 204
6.6.4 Command Security ......................................................... 205
6.6.5 System Built-In Functions ............................................... 205
6.7 Object Permissions .......................................................... 207
6.7.1 With Grant Option .......................................................... 208
6.7.2 Revoking Object Access .................................................. 209
6.7.3 Granting Vast Permissions ............................................... 209
6.7.4 Displaying Permissions ................................................... 210
6.7.5 Ownership Chains ......................................................... 210
6.7.6 Test/Change Permissions ................................................ 213
6.8 Access Rules ................................................................. 213
6.8.1 Access Rules Using Java Function and Application Contexts ................................................... 213
6.8.2 Syntax for Access Rules .................................................. 214
6.8.3 Disable Access Rules ....................................................... 218
6.8.4 Access Rules and bcp ...................................................... 218
6.8.5 Fine-Grained Access Control (FGAC) ................................ 219
6.9 Column Encryption .......................................................... 219
6.9.1 Enabling Encryption ........................................................ 221
6.9.2 The sysencryptkeys Table ............................................... 221
6.9.3 Creating New Tables with Encryption ................................. 223
6.9.4 Altering Existing Tables ................................................... 224
6.9.5 Select Into Syntax with Encryption ................................... 224
6.9.6 Cipher Text Randomization ............................................. 225
6.9.7 Column Encryption Performance Considerations .............. 225
6.9.8 Decryption Permissions ................................................... 226
6.10 Summary ................................................................. 227
7 Auditing ..................................................................................... 229
7.1 Installation Overview ......................................................... 230
7.2 The syssecurity Database and Tables ..................................... 232
7.2.1 syssecurity Transaction Log ............................................ 235
7.2.2 syssecurity Stored Procedures ......................................... 236
7.2.3 Set Auditing Options ....................................................... 236
7.2.4 Display Enabled Auditing Options ................................. 238
7.2.5 Write User-Defined Comment to Audit Trail .................... 239
7.2.6 Add Audit Table to Audit Trail ........................................ 240
7.3 Audit Storage ................................................................. 241
7.3.1 Queue ................................................................. 241
7.3.2 Audit Database Storage Requirements ............................... 242
7.3.3 Archiving Audit Records ............................................... 244
7.3.4 Creating Audit Tables .................................................... 245
7.4 Querying the Audit Trail ..................................................... 246
7.5 External Applications and Third-Party Software ........................ 249
7.6 Additional Tips on How to Effectively Audit ............................ 250
7.7 Summary ........................................................................... 250
Index

@@trancount, 154
$SYBASE directory, 73

A
Access method costs, 580
Access rule, 213
bind to column, 214
bulk copy utility (bcp), 218
disable, 218
extended syntax, 215
Java function and application contexts, 213
row-level restriction, 219
syntax, 214
Active roles, 196
Active-active architecture, 535
Address listing, 28
Administration tools, 613
Alias, 178, 690
assign to other user, 186
dbo, 178
Allocated shared memory, 325
Allow remote access, 328
Allow resource limits, 292
Alter database, 135
ANSI standards, 152
ANYENGINE, 307, 313
Asymmetric mode, 534
Asynchronous initialization, 130
Asynchronous write, 509
Atomic, Consistent, Isolated, Durable (ACID), 599
Audit
archive records, 244
commands, 233
create table, 245
database storage requirements, 242
write comment to trail, 239
Audit queue
and requirements for storage, 241
size configuration, 241
Audit trail
add audit table, 240
add message, 240
Audit trail (Cont.)
archive, 244
query, 246
Auditing, 31, 229
display options enabled, 238
enable, 231
global and database options, 234
installation, 230
limitations, 237
multi-table, 230
set options, 236
specific, 238
stored procedures, 236
third-party interactions, 249
Auditinit, 230
Automatic system boot, 86
Availability
database layers, 504
network, 505
Backup
plan, 251
restore to existing database, 266
schedule, 254, 458
schedule recurring, 49
storage save, 264
types, 254
versus restore, 524
Backup server, 256
advantages, 256
dump compression, 264
identify name, 257
media change during dump/load, 258
remote, 257, 263
shut down, 258
start, 258
Base priority, 307
bcp, 614
command examples, 617
mode, 616
parallel, 618
run on partitions, 394
transfer data, 614
begin tran statements, 153
Binary
installation, 68
order, 58
Binding, 376
defaults, 690
Blocked process, 696
Bookmark, 152
Boot phase, 94
Boot sequence, 516
Bottleneck, 426
Bootstrap, 307
Buffer pool, 371
cache, 372
cache/reorder, 373
create, 372
size, 372
Buffers
fetch and discard, 383
Bulk copy program—see bcp
Bulk copy utility (bcp), 218
Business process continuity, 540

Cache, 156
binding commands, 370
bottleneck, 365
configure, 361
create/configure, 370
dbcc checkstorage, 477
drop bind, 377
hit rate, 383
IMDB, 605
memory guidelines, 362
memory needed, 362
optimization plan, 364
partition, 383
partitioning, 385
relaxed strategy, 44
replacement strategies, 586
report on configuration, 368
room for addition increase, 325
size, 361
tune, 379
type, 322

Cachelets, 385
Capital markets, 600
Cascade option, 209
Catalog, 123
Centralized authentication mechanism, 171
Chained mode, 150
Checkpoint, 158, 276
configure multiple tasks, 160
free, 160
Cipher text, 225
columns, 226
CIS, 47, 331, 413
add remote server, 414
column size, 421
file access, 421
Client-server architecture, 73
Client-server paradigm, 407
Cluster
management layer, 535
ratio, 584
redundant server requirement, 505
Clustering, 532
Cold standby, 504, 525
Column encryption, 53, 219, 224
performance, 225
Command Parser, 623
Commands list, 687
Commit, 156
Commit tran, 152
Component Integration Services—see CIS
Compression assistant utility, 48
Concurrency management, 41
Concurrency clients, 38
Configuration
data, 68
summary, 72
Configuration file, 327, 353
corrupted, 354
format, 354
location, 94
Configuration system table, 327
Configuration variable
set and read, 327
Consistency, 558
Consistency checker, 268
Contention on objects, 648
Corrupted database, 267

Cost-based optimization, 569
CPU
cycle, 642
spinning, 43
time, 307
Create database
events, 133
Create script, 269
Creation script, 620
cron, 633
Cross-database constraints, 290
Cross-database transactions, 522
Cumulative dump, 254, 282
relative frequencies, 282

D
Data
archive/delete, 394
change metric, 588
compression, 52
consistency, 394
copy to OS file, 614
fragment, 271
loss prevention, 498
maintenance, 391
mining, 669
replication, 539
retention, 499
server, 547
to disk, 158
transfer directly to disk, 99
Data cache, 322, 365
defaults, 371
increase size, 322
reduce heavy use, 385
size, 367
sizing, 379
spinlock, 385
tempdb, 385
Data Federation, 52
Data page
cache, 157
Data partitioning, 51, 391
semantic methods, 396
types, 396
when to use, 397

Database
active to standby switch, 563
attributes, 113
backup, 251
cache, 604
can’t be accessed, 697
change default behaviour, 141
change owner of, 174
change/expand, 135
changes list, 132
clear pages, 134
common characteristics, 123
complete description, 137
default, 123
decision, 70
drop, 140
drop damaged, 470
dump, 259
dump and load (cold standby), 526
dump and load permissions, 185
election, 53
in-memory database, 129
in-memory, temporary, 606
kill user-connection, 655
load, 266
load from database dump, 130
load summary, 268
master, 125
mirror, 106
model, 126
mount/unmount, 290
online, 268
options, 141
ownership, 132
permission to dump/load, 252
recoveryability, 497
recovery order of, 289
reduce cost of maintenance, 392
redundant copy, 525
remove, 470
restore after disk crash, 255
Database (Cont.)
schedule maintenance, 457
segment, 270
separate log from data, 131
sizing, 129
storage area, 91
structure, 123
syssecurity, 127
sysystemprocs, 127
system, 125
tempdb, 126
template for creation, 126
template for IMDB, 608
threshold error, 439
types, 34
unique users, 648
Database backup, 520
cross-database transaction, 522
incremental/transaction, 521
native, 522
Database consistency checker → see dbcc
Database device utilization report, 658
Database devices
limitations, 92
database dump, 251, 254
incremental, 254
Database layer, 576
backup, 522
Database log
create on separate drive, 132
prevent from filling, 167
Database server
active/inactive connections, 663
Database table
column encryption, 219
Data-intensive business application, 600
DB user name, 649
dbcc, 128, 456
crypt, 481
crypt user name, 649
crypt user names, 649
dbcc checkallocc, 464
dbcc checkcatalog, 468
dbcc checkdb, 463
dbcc checksum, 460
column, 460
commands, 485
consistency checks, 484
country, 479
generate reports, 485
maintain, 482
plan, 479
recommend size for, 470
statistics, 487
update configuration, 483
dbo
definition, 185
dbo privileges, 178
dbschema.pl, 644
DDL
commands, 689
reverse-engineering tool, 645
dlgen, 620
command examples, 622
Deadlock, 334
trace flag, 443
Deadlocking, 635
Decryption permissions, 226
default, 226
default compression, 130
defalcopy, 619
command examples, 620
command tips, 620
parallel, 620
Delayed commits, 290
Destination database, 267
Development and review process, 642
dbcc
dbc checkallocc, 464
dbc checkcatalog, 468
dbcc checkdb, 463
dbcc checksum, 460
compare results, 491
configuration parameters, 476
parallelism, 476
report, configuration information, 487
summary of operations, 485
system preparation, 462
dbcc checktable, 462
dbcc dbrepair, 470
dbcc indexalloc, 465
dbcc tablealloc, 466
dbccdb
add table/stored procedure, 480
clean, 483
consistency checks, 484
country, 479
generate reports, 485
maintain, 482
plan database, 479
recommend size for, 470
statistics, 487
update configuration, 483
dbo
definition, 185
DBo privileges, 178
dbschema.pl, 644
DDL
commands, 689
reverse-engineering tool, 645
dlgen, 620
command examples, 622
Deadlock, 334
trace flag, 443
Deadlocking, 635
Decryption permissions, 226
default, 226
default compression, 130
defalcopy, 619
command examples, 620
command tips, 620
parallel, 620
Delayed commits, 290
Destination database, 267
Development and review process, 642
dbcc checkallocc, 464
dbccheckcatalog, 468
dbcc checkdb, 463
dbcc checksum, 460
column, 460
country, 479
generate reports, 485
maintain, 482
plan, 479
recommend size for, 470
statistics, 487
update configuration, 483
dbo
definition, 185
DBo privileges, 178
dbschema.pl, 644
DDL
commands, 689
reverse-engineering tool, 645
dlgen, 620
command examples, 622
Deadlock, 334
trace flag, 443
Deadlocking, 635
Decryption permissions, 226
default, 226
default compression, 130
defalcopy, 619
command examples, 620
command tips, 620
parallel, 620
Delayed commits, 290
Destination database, 267
Development and review process, 642

Device
create, 96, 100
default, 101
enhance usability with software, 111
information storage, 92
in-memory and relaxed-durability database, 120
installation, 63
locate pages on, 117
logical name, 96
mirroring, 103
remove, 102
specify as private, 99
Device file
backup, 522
directio, 510
Directory services, 75
Dirty page
flush cache to disk, 160
Disaster recovery, 115, 499
Disaster recovery, 115, 499
approach, 501
database, 123
plan, 493, 499
Disk
crash, 255
file backup, 259
mirror, 104
partition, 546
replication, 523
disk init, 29
disk refk, 281
disk reinit, 280
disk transfer, 261
disk tran dbname, 257
Dumps and loads access media on separate
hardware, 257
Duplicate components, 502
Durability, 129, 603
level, 604

Dump (Cont.)
multiple databases to a single tape, 263
tape devices or disk files, 259
test, 525
time required, 267
tape, 524
tape devices, 261
tape transfer, 261
tape to tape, 524
tape tran dbname, 257
Dumps and loads access media on separate
hardware, 257
Duplication components, 502
Durability, 129, 603
level, 604

E
Embedded Replication Server System Data-
base (ERSSD), 546
Enable DTM, 331
Encrypted column
index, 225
joins, 226
Encryption
columns, 224
columns, 224
create with, 223
crypt, 221
key storage, 221
syntax, 222
Enforcement type, 298
Engine
affinity, 307
busy utilization, 429
lock transfer, 43
number, 313
Engine group, 306, 313
assignment, 307
create own, 313
create group/engine, 314

708
Lock enhancements, 44
cascaded, 635
Manager, 334
promotion PCT, 335
promotions, 335
shared lock, 432
Locking contention, 430
Locks
number of, 335
Log cache, 366
Log shipping, 526
Log suspended, 167
Logical connection, 557, 560
Logical data space, 331, 413
Logical device, 91
creation, 100
Logical name, 96
physical file, 96
Logical page size, 59
Logical Process Manager, 305
conflicts and precedence, 317
execution classes, 307
stored procedures, 308
Logical separation of SAP ASE devices, 512
Login, 30
change existing information, 201
disable without removing, 187
remove, 199
resource limits, 300
show information, 200
Login ID and password combination, 175
Login names
mapping, 419
oid, 432
Low and high columns, 117
Low Water Mark, 335

M
Maintenance, 425
Manifest file, 523
Master database, 34, 93, 125
dump, 93
restore, 279
Master device
build, 93
bytes, 94
creation, 93
initialization, 63
location, 94
mirror, 106
Master system tables, 126, 143
Match login and user IDs, 685
Max memory, 323
Maximum allowed number of devices, 332
MDA table, 372, 379
Mean time between failures, 499
Media changes during dumps and loads, 258
Media failure, 275
Memory, 365
configuration variables, 326
configure, 321
pre-SAP ASE12.5, 321
SAP ASE 12.5 and later, 322
shared, 325
shared, unallocated, 325
use, 335
use only as specified, 365
Memory configuration boundaries, 323
Memory-related variables, 359
Metadata, 130
cache, 43, 340
Minimally Logged DML, 610
Minimum-size database, 131
Mirror
all database devices, 106
decide what to mirror, 105
Mirror device
initialize, 105
size, 104
synchronous/asynchronous writes, 104
Mirroring, 103, 507
disable, 107
disable manually, 108
software/hardware, 109
Model database, 35, 93, 126
Monitor server, 429, 644
Monitoring, 341
Monitoring server
stored procedure, 645
Most recently used → see MRU
Mount point, 511
MRU, 595
cache replacement strategy, 387
replacement strategy, 383
Multiple processors, 339
Multiple SAP ASE instances on the same server, 68
Multiple-trigger support, 47
Multi-processor environment, 305
Multi-Site Availability (MSA), 529, 544
Multistage processing, 126
Multi-table auditing, 230
Multithreaded architecture, 38

N
Named cache, 365
bind to, 377
bound object, 376
collection guidelines, 369
define, 367
IMDB, 599
transaction log, 369
transaction performance, 367
tuning tips, 384
when to use, 366
NC index, 634
Network
communication, 342
packet size, 342
timeout, 411
type, 57
Networking information, 61
Object
assign to cache, 376
bind to defined execution class, 310
bound to cache, 365
cache, 365
copy definitions → see defncopy
define, bound to execution classes, 305
Object (Cont.)
execution class, 306
faults report, 489
no access, 698
permissions, 207
revoke access, 209
unbind from cache, 377
Omni, 419
Online database, 267, 268, 277
Open Client, 82
Open Client libraries (CT-Lib), 82
Open Client/Server (OCS), 73
Open Database Connectivity (ODBC), 83
Open databases, 340
oper_role, 182, 252
Operating system clustering, 532
Operating system file, 95
Operator role (OPER), 185
Operator/console message, 78
optdiag, 519, 589, 625
histogram output, 593
index statistics, 592
Optimistic concurrency control, 557
Optimizer engine, 575
Optimizer statistics, 495
view, 589
Override, 129
Ownership chain, 210

P
Page chain, 374
load from disk, 365
make physical copy, 261
size, 421
Parallel process, 343, 585
Parameter
default network packet size, 342
maximum failed logins, 351
maximum network packet size, 342
print deadlock information, 335
upgrade version, 349
Partition
configure, 403
data cache, 365
Index

Partition (Cont.)
delete information without locking, 405
information, 403
spinlock ratio, 403
stable storage, 546
statistics, 405
target, 404
Partition-level locking, 44
Partition-specific maintenance, 391
Password
assign to roles, 193
count, 187
crypt, 194
crypt, 221
generate new SSO, 187
require digit in, 179, 181
server-wide configuration parameters, 187
set number of days between changes, 187
Patches, 513
Pathname method, 415
Performance and tuning analysis, 642
Performance documentation, 644
Performance improvement, 599
Performance targets, 498
Performance tuning, 581, 641
Physical device
options, 60
Physical file
qualify name, 97
Physical I/O, 573
Physical memory, 343
Physical separation of SAP ASE devices, 512
Physname, 96
Plugable Authentication Module (PAM), 171
Point-in-time analysis questions, 230
Point-in-time restoration, 277
Pool
create/configure, 370
Port
number, 61, 79
override start, 515
Precedence rule, 317
Prefetch strategy, 383
Primary key constraint, 395
Priorities, 306
Prioritization of work, 305
Procedure cache, 322, 362
size, 340
sizing, 379
Production databases
lead to development environment, 641
Production system
backups, 253
Proxy, 413
create database, 417
database, 130
table, 414
Proxy tables
SAP HANA, 47

Query

determine cost of, 578
develop in graphical environment, 626
estimate cost of, 573
evaluation and execution, 569
limit rows for return, 297
limit second a batch can run, 298
parallel process, 343
parallel processing, 585
plan, 571
plans used for, 571
tree, 215

Query processor, 51, 374, 569
search argument, 579
Queue, 241
priority, 306
Queue database, 283, 522

R

RAID, 507
Range partitioning, 396, 398
Raw device, 94
management, 95
size, 98
Raw partition, 508
Read-mostly data, 600
Real-time monitoring, 644
Recoverability
database layers, 504
separation of devices, 512
Recovery, 160
basic unit of, 251
cost, 503
interval, 161
interval in minutes, 328
more than one database, 517
parallel, 289
planning, 503
point in time, 278
set time per database, 161
up to the minute, 278
Redundancy, 502
active vs. passive, 502
power, 505
storage layer, 507
Redundant
server, 505
site, 505
Redundant Array of Inexpensive Drives (RAID), 110
Regular file, 94
Regular replication, 544
Relational database, 37
Relaxed query limits, 572
Relaxed-durability database
create, 611
Remote access, 410
setup, 411
Remote backup, 263
server, 257
Remote login, 411
remove, 412
Remote object
local storage location, 415
Remote Procedure Call (RPC), 80, 257, 407
Remote server
access like local server, 413
add for CDS, 414
login/password, 419
management, 407
one-to-one mapping, 420
proxy database, 417
reorg utility, 294
Replication, 531
agent, 547
definition, 546
Replication Server System Database (RSSD), 546
Residual data removal, 46
Resource Governor, 291
Resource limit
create, 296
enable, 292
enforcement action, 299
enforcement type, 298
hierarchies, 300
system tables, 302
time periods, 292
type, 297
user defined, 303
Resource verification, 454
Response time, 602
Restore
corrupted database, 267
scenario, 255
Role
activate, 192
assign passwords, 193
assign to users, 192
create, 191
default for login, 194
disable, 190
display, 196
display information, 195
drop, 193
enable (password), 191
expand down, 196

Index
SAP ASE 16 (Cont.)
uncach mode default, 151
version 11.5, 306
SAP ASE Cluster Edition, 50, 535
SAP ASE HA
failover, 534
SAP ASE IMDB, 599
create, 606
durability, 603
operations, 605
versus traditional database, 602
SAP ASE Relaxed-Durability Database (RDDDB), 603
SAP ASE to HANA data access, 47
SAP Control Center (SCC), 48
backup scheduling, 49
SAP HANA, 38
SAP Replication Server, 54, 539
add standby database, 562
components, 545
installation, 550
installation planning, 547
software installation, 549
system tables, 546
warm standby, 528
Savetopoint, 155
Scope, 299
parameter, 311
Script maintenance, 494
Search engine, 579
Secondary environment, 276
Secure Sockets Layer (SSL), 31, 74
protocol, 413
Security, 79
database-level, 176
for commands, 205
integration, 171
level, 172
object, 208
operating system-Level, 174
server-level, 175
Segment number, 270
Segment, 271, 391
free-space threshold, 273
space requirements, 685
Self-healing, 499
Semantic data partitioning, 396
Separation of mount points, 511
Server
address, 79
authentication, 80
class, 414
configure for multiple processors, 339
control access to, 186
display connections, 202
entries, 28
get information, 205
identification, 257
layer, 39
limits resources, 291
look up address of remote server, 409
memory configuration, 363
name, 61
optimization goal, 68
process ID, 164
process IDs (SPIDs), 51
real-time information, 429
re-create, 494
role, 188
security, 187
shut down, 88
startup, 70
startup settings, 76
storage, 29
users can’t gain access, 696
valid startup, 439
verify startup, 86
Server configuration file
servername.cfg, 78
Server start, 83
UNIX, 84
UNIX, manual, 85
Windows, 84
Server-level maintenance, 425
servername.cfg file, 78
Server-to-server communications, 410
Server-wide audit options, 233
Service, 28
Session-based security, 74
set proxy command, 203
setuser command, 186
Shared disk scenario, 532
Shared memory allocation, 323
Shared-disk cluster, 50
Shopping cart type application, 601
Showplan, 571, 574
Shutdown
graceful, 89
maintenance, 89
Sizing, 134
Software update, 455
Software-level mirroring, 110
Sort order, 58
Stop
add standby database, 239
addauditstored procedure, 240, 245
addserver, 409
addumpdevice, 259
auditcmd, 249
audit stored procedure, 236
changedowner, 174
configure
configure info for specific user, 353
output, 328
run recovery, 517
speed boot time, 517
sp.displayaudit stored procedure, 238
sp.displaylogin, 197
sp.helprotect, 197
sp.heluser, 179
sp.listener, 515
sp.lock, 430
sp.objectstats, 648
sp.plan_DbCdb, 470
sp.role, 189
sp.spaceused, 243
sp.syntax, 128
sp.sysmon, 426
Data Cache Management, 379
output, 428
output for transaction log, 370
performance monitoring, 648
sp.threshholdaction, 167, 274
sp.volchanced, 258, 279
sp.who, 433
Space management, 494
spid, 184
Spinlock, 385
reduce contention in cache, 378
SAP ASE IMDB, 599
SAP ASE HA
SAP ASE 16
SAP ASE to HANA data access
SAP Control Center (SCC)
spt_values, 327
SQL standards, 642
transaction behavior, 150
SQL Server Administration, 346
SSL, 79
SSL Plus™ library, 79
SSO security-related options, 350
stable, 186
Standby system, 503
Star join, 570
Standby system configuration, 48
Storage bytes, 58
Storage layer
replicate change made to disk, 523
StoredProcedure, 522
StoredProcedure procedure, 212
find source code, 636
physical environment information, 111
run on remote server, 408
user can’t run, 698
Storing statistics, 583
Striped output device, 262
Sun Solaris 2.4+ system, 633
SYB_BACKUP entry, 257
Sybase login, 62
Sybase Software Asset Manager (SySAM), 61
Sybsecurity database, 232
System role
definitions, 189
functions, 189
turn off, 190
System security officer role, 186
System stored procedure, 127
System table, 113, 123
allow updates, 349
information, 625
modify, 114
query, 114
recovery, 115
sysdevices, 115
System utilization information, 426
systemusages, 303
sysusages, 144
Table
check integrity, 466
corruption, 253
corruption prevention, 253
corruption recovery, 253
corruption prevention and recovery, 253
corruption prevention and recovery procedures, 253
corruption prevention and recovery system, 253
create from system files, 418
crypt columns, 224
MDA, 429
partitioned, 393
dump, 254, 274
range partitioning, 398
Space allocation, 466
symcryptionkeys, 221
syslogins, 200
sysresourcelimits, 303
Table (Cont.)
sysvprobes, 190
sysusers, 183
table-level maintenance, 494
Tabular Data Stream (TDS), 82, 407
tape backup, 259
tape stripping, 493
task
high priority, 318
initial priority, 307
TCP/IP, 61
tempdb, 70, 93
assign to own data cache, 385
move off master device, 637
multiple, 516
dumped cached, 366
purpose, 126
size, 127
work table, 126
Template database, 130, 608
temporary
database, 129, 606
table, 126
Terminate a process, 204
test dump, 525
Thread pool, 313
Threaded kernel mode, 39
Threshold, 235, 272
free space, 273
last-chance, 272
Threshold procedure, 168
Time range, 293, 303
add, 294
chart, 294
modify, 295
remove, 296
Timeslice, 307
total data cache size, 329
total logical memory, 324
total memory configuration parameter, 321
total physical memory, 324
Transaction
abort, 141
abort/suspend, 274
control statements, 152
definition, 149
dump, 254, 274
Transaction (Cont.)
dump activities, 276
durable, 603
load, 277
manage across multiple DB, 331
mark start of, 152
modes of behavior, 150
nesting, 154
rollback, 160
rollback pending work, 155
uncovered, 161
write to disk, 129
Transaction dump
strategy, 521
Transaction log, 149, 369
2-page pool, 370
automatically truncate the log on checkpoint, 162
backup recommendation, 253
data reconciliation, 160
dumps from primary server, 276
IMDB, 599
inserted/deleted tables, 370
is full, 162
monitor, 271
save, 155
size cache/pool, 369
suspend, 167
sybsecurity, 235
sybs, 156
truncate, 162
truncate without checkpoint, 275
truncate without disk edit, 275
transaction without dump, 275
Transaction log device
mirror, 106
Transaction mode, 150
change, 151
Triggers, 370
Troubleshooting, 695
Truncation point, 162
determination, 162
trusted roles file, 80
T-SQL commands, 623
Unallocated shared memory, 325
Unavailability cause, 499
Unchained mode, 150
Unicode, 351
Unit specifier, 97
UNIX
  create logical device, 100
  manual startup, 85
  start at boot time, 86
  startup, 84
  system backup, 633
  system process table, 686
  verification options, 86
Unrecovered log record, 161
Uptime, 497
User, 30
  add to database, 177
  can’t access server, 696
  connections, 340
  display information, 179
  interface, 82
  kill connection to database, 655
  listener, 515
  log cache, 157, 604
  log cache queue, 42
  unique, 648
User Database
  install/maintain, 29
Virtual page, 94
  size, 97
Virtual starting address, 99
Volume management, 111
Warm standby, 504, 528, 544, 556
  enable database replication for, 561
  quiesce, 285
  set up replication, 559
  setup, 276
Wash area, 374
  bottleneck, 375
  size, 375
Windows
  start at boot time, 86
  startup, 84
  Task Manager, 87
  verification options, 87
  with grant option, 208
  with standby access, 527
Worker process, 476
Workspace size, 475
Workspaces
  concurrent, 475
  segment, 479
Write-ahead log, 42, 156
Write-ahead transaction logging, 604
Zero-disk footprint database, 603
Jeffrey Garbus is a database consultant and CEO of Soaring Eagle Consulting, an RDBMS consulting and training firm. A consultant and trainer for more than 20 years, he currently focuses his work on very large databases, data warehousing, training, and remote database administration. Jeff is a frequent speaker at ISUG and industry conferences and is the author of several other books about Sybase.