# 

1.1.1        [What is SQL Server and ASE anyway?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.1)  
1.1.2        [How do I start/stop ASE when the CPU reboots?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.2)  
1.1.3        [How do I move *tempdb* off of the master device?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.3)  
1.1.4        [How do I correct *timeslice -201*?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.4)  
1.1.5        [The how's and why's on becoming a *Certified Sybase Professional DBA (CSPDBA)*?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.5)  
1.1.6        [RAID and Sybase](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.6)  
1.1.7        [How to swap a db device with another](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.7)  
1.1.8        [Server naming and renaming](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.8)  
1.1.9        [How do I interpret the tli strings in the interface file?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.9)  
1.1.10      [How can I tell the datetime my Server started?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.10)  
1.1.11      [Raw partitions or regular files?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.11)  
1.1.12      [Is Sybase Y2K (Y2000) compliant?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.12)  
1.1.13      [How Can I Run the ASE Upgrade Manually?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.13)  
1.1.14      [We have lost the sa password, what can we do?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.14)  
1.1.15      [How do I set a password to be null?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.15)  
1.1.16      [Does Sybase support Row Level Locking?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.16)  
1.1.17      [What platforms does ASE run on?](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#1.1.17)

# 1.1.1: What is SQL Server and ASE?

## Overview

Before Sybase System 10 (as they call it) we had Sybase 4.x. Sybase System 10 has some significant improvements over Sybase 4.x product line. Namely:

* the ability to allocate more memory to the dataserver without degrading its performance.
* the ability to have more than one database engine to take advantage of multi-processor cpu machines.
* a minimally intrusive process to perform database and transaction dumps.

## Background and More Terminology

A ASE (SQL Server) is simply a Unix process. It is also known as the database engine. It has multiple threads to handle asynchronous I/O and other tasks. The number of threads spawned is the number of engines (more on this in a second) times five. This is the current implementation of Sybase System 10, 10.0.1 and 10.0.2 on IRIX 5.3.

Each ASE allocates the following resources from a host machine:

* memory and
* raw partition space.

Each ASE can have up to 255 databases. In most implementations the number of databases is limited to what seems reasonable based on the load on the ASE. That is, it would be impractical to house all of a large company's databases under one ASE because the ASE (a Unix process) will become overloaded.

That's where the DBAs experience comes in with interrogation of the user community to determine how much activity is going to result on a given database or databases and from that we determine whether to create a new ASE or to house the new database under an existing ASE. We do make mistakes (and businesses grow) and have to move databases from one ASE to another. At times ASEs need to move from one CPU server to another.

With Sybase System 10, each ASE can be configured to have more than one engine (each engine is again a Unix process). There's one primary engine that is the master engine and the rest of the engines are subordinates. They are assigned tasks by the master.

Interprocess communication among all these engines is accomplished with shared memory.

*Some times when a DBA issues a Unix* ***kill*** *command to extinguish a maverick ASE, the subordinate engines are forgotten. This leaves the shared memory allocated and eventually we may get in to situations where swapping occurs because this memory is locked. To find engines that belong to no master ASE, simple look for engines owned by /etc/init (process id 1). These engines can be killed -- this is just FYI and is a DBA duty.*

Before presenting an example of a ASE, some other topics should be covered.

### Connections

An ASE has connections to it. A connection can be viewed as a user login but it's not necessarily so. That is, a client (a user) can spark up multiple instances of their application and each client establishes its own connection to the ASE. Some clients may require two or more per invocation. So typically DBA's are only concerned with the number of connections because the number of users typically does not provide sufficient information for us to do our job.

Connections take up ASE resources, namely memory, leaving less memory for the ASEs' available cache.

### ASE Buffer Cache

In Sybase 4.0.1 there was a limit to the amount of memory that could be allocated to a ASE. It was around 80MB, with 40MB being the typical max. This was due to internal implementations of Sybase's data structures.

With Sybase System 10 there really was no limit. For instance, we had an ASE cranked up to 300MB under 10. With System 11 and 12 this has been further extended. ASE's with 4G bytes of memory are not uncommon. I have not heard of an 11.9.3 or a 12 server with *more* that 4G bytes, but I am sure that they are not far away.

The memory in an ASE is primarily used to cache data pages from disk. Consider that the ASE is a light weight Operating System: handling user (connections), allocating memory to users, keeping track of which data pages need to be flushed to disk and the sort. Very sophisticated and complex. Obviously if a data page is found in memory it's much faster to retrieve than going out to disk.

Each connection takes away a little bit from the available memory that is used to cache disk pages. Upon startup, the ASE pre-allocates the memory that is needed for each connection so it's not prudent to configure 500 connections when only 300 are needed. We'd waste 200 connections and the memory associated with that. On the other hand, it is also imprudent to under configure the number of connections; users have a way of soaking up a resource (like an ASE) and if users have all the connections a DBA cannot get into the server to allocate more connections.

One of the neat things about an ASE is that it reaches (just like a Unix process) a working set. That is, upon startup it'll do a lot of physical I/O's to seed its cache, to get lookup information for typical transactions and the like. So initially, the first users have heavy hits because their requests have to be performed as a physical I/O. Subsequent transactions have less physical I/O and more logical I/O's. Logical I/O is an I/O that is satisfied in the ASEs' buffer cache. Obviously, this is the preferred condition.

### DSS vs OLTP

We throw around terms like everyone is supposed to know this high tech lingo. The problem is that they are two different animals that require a ASE to be tuned accordingly for each.

Well, here's the low down.

DSS

Decision Support System

OLTP

Online Transaction Processing

What do these mean? OLTP applications are those that have very short orders of work for each connection: fetch this row and with the results of it update one or two other rows. Basically, small number of rows affected per transaction in rapid sucession, with no significant wait times between operations in a transaction.

DSS is the lumbering elephant in the database world (unless you do some tricks... out of this scope). DSS requires a user to comb through gobs of data to aggregate some values. So the transactions typically involve thousands of rows. Big difference than OLTP.

We never want to have DSS and OLTP on the same ASE because the nature of OLTP is to grab things quickly but the nature of DSS is to stick around for a **long** time reading tons of information and summarizing the results.

What a DSS application does is flush out the ASE's data page cache because of the tremendous amount of I/O's. This is obviously very bad for OTLP applications because the small transactions are now hurt by this trauma. When it was only OLTP a great percentage of I/O was logical (satisfied in the cache); now transactions must perform physical I/O.

That's why it's good not to mix DSS and OLTP if at all possible.

If mixing them cannot be avoided, then you need to think carefully about how you configure your server. Use named data caches to ensure that the very different natures of OLTP and DSS do not conflict with each other. If you tables that are shared, consider using dirty reads for the DSS applications if at all possible, since this will help not to block the OLTP side.

### Asynchronous I/O

Why async I/O? The idea is that in a typical online transaction processing (OLTP) application, you have many connections (over 200 connections) and short transactions: get this row, update that row. These transactions are typically spread across different tables of the databases. The ASE can then perform each one of these asynchronously without having to wait for others to finish. Hence the importance of having async I/O fixed on our platform.

### Engines

Sybase System 10 can have more than one engine (as stated above). Sybase has trace flags to pin the engines to a given CPU processor but we typically don't do this. It appears that the master engine goes to processor 0 and subsequent subordinates to the next processor.

Currently, Sybase does not scale linearly. That is, five engines do not make Sybase perform five times as fast however we do max out with four engines. After that performance starts to degrade. This is supposed to be fixed with Sybase System 11.

## Putting Everything Together

As previously mentioned, an ASE is a collection of databases with connections (that are the users) to apply and retrieve information to and from these containers of information (databases).

The ASE is built and its master device is typically built over a medium sized (50MB) raw partition. The tempdb is built over a cooked (regular - as opposed to a raw device) file system to realize any performance gains by buffered writes. The databases themselves are built over the raw logical devices to ensure their integrity. (Note: in System 12 you can use the *dsync* flag to ensure that writes to file system devices are secure.

### Physical and Logical Devices

Sybase likes to live in its own little world. This shields the DBA from the outside world known as Unix, VMS or NT. However, it needs to have a conduit to the outside world and this is accomplished via devices.

All physical devices are mapped to logical devices. That is, given a physical device (such as /lv1/dumps/tempdb\_01.efs or /dev/rdsk/dks1ds0) it is mapped by the DBA to a logical device. Depending on the type of the device, it is allocated, by the DBA, to the appropriate place (vague enough?).

Okay, let's try and clear this up...

### Dump Device

The DBA may decide to create a device for dumping the database nightly. The DBA needs to create a dump device.

We'll call that logically in the database **datadump\_for\_my\_db** but we'll map it to the physical world as **/lv1/dumps/in\_your\_eye.dat** So the DBA will write a script that connects to the ASE and issues a command like this:

dump database my\_stinking\_db to datadump\_for\_my\_db

go

and the **backupserver** (out of this scope) takes the contents of **my\_stinking\_db** and writes it out to the disk file **/lv1/dumps/in\_your\_eye.dat**

That's a dump device. The thing is that it's not preallocated. This **special** device is simply a window to the operating system.

## Data and Log Devices

Ah, now we are getting into the world of pre-allocation. Databases are built over raw partitions. The reason for this is because Sybase needs to be guaranteed that all its writes complete successfully. Otherwise, if it posted to a file system buffer (as in a cooked file system) and the machine crashed, as far as Sybase is concerned the write was committed. It was not, however, and integrity of the database was lost. That is why Sybase needs raw partitions. But back to the matter at hand...

When building a new ASE, the DBA determines how much space they'll need for all the databases that will be housed in this ASE.

Each production database is composed of **data** and **log**.

The **data** is where the actual information resides. The **log** is where the changes are kept. That is, every row that is updated/deleted/inserted gets placed into the **log** portion then applied to the **data** portion of the database.

That's why DBA strives to place the raw devices for logs on separate disks because everything has to single thread through the log.

A transaction is a collection of SQL statements (insert/delete/update) that are grouped together to form a single unit of work. Typically they map very closely to the business.

I'll quote the Sybase ASE Administration Guide on the role of the **log:**

The transaction log is a write-ahead log. When a user issues a statement that would modify the database, ASE automatically writes the changes to the log. After all changes for a statement have been recorded in the log, they are written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page. At that time, it is written to disk. If any statement in a transaction fails to complete, ASE reverses all changes made by the transaction. ASE writes an "end transaction" record to the log at the end of each transaction, recording the status (success or failure) of the transaction

As such, the **log** will grow as user connections affect changes to the database. The need arises to then clear out the log of all transactions that have been flushed to disk. This is performed by issuing the following command:

dump transaction my\_stinking\_db to logdump\_for\_my\_db

go

The ASE will write to the dumpdevice all transactions that have been committed to disk and will delete the entries from its copy, thus freeing up space in the **log**. Dumping of the transaction logs is accomplished via **cron** (the Unix scheduler, NT users would have to resort to **at** or some third party tool) . We schedule the heavily hit databases every 20 minutes during peak times.

A single user can fill up the log by having **begin transaction** with no corresponding **commit/rollback transaction**. This is because all their changes are being applied to the **log** as an open-ended transaction, which is never closed. This open-ended transaction cannot be flushed from the log, and therefore grows until it occupies all of the free space on the log device.

And the way we dump it is with a dump device. :-)

### An Example

If the DBA has four databases to plop on this ASE and they need a total of 800MB of data and 100MB of log (because that's what really matters to us), then they'd probably do something like this:

1. allocate sufficient raw devices to cover the **data** portion of **all** the databases
2. allocate sufficient raw devices to cover the **log** portion of **all** the databases
3. start allocating the databases to the devices.

For example, assuming the following database requirements:

|  |  |  |
| --- | --- | --- |
| Database Requirements | | |
| **DB** | **Data** | **Log** |
| a | 300 | 30 |
| b | 400 | 40 |
| c | 100 | 10 |

and the following devices:

|  |  |  |
| --- | --- | --- |
| Devices | | |
| **Logical** | **Physical** | **Size** |
| dks3d1s2\_data | /dev/rdsk/dks3d1s2 | 500 |
| dks4d1s2\_data | /dev/rdsk/dks4d1s2 | 500 |
| dks5d1s0\_log | /dev/rdsk/dks5d1s0 | 200 |

then the DBA may elect to create the databases as follows:

create database a on dks3d1s2\_data = 300 log on dks5d1s0\_log = 30

create database b on dks4d1s2\_data = 400 log on dks5d1s0\_log = 40

create database c on dks3d1s2\_data = 50, dks4d1s2\_data = 50 log on dks5d1s0\_log = 10

Some of the devices will have extra space available because out database allocations didn't use up all the space. That's fine because it can be used for future growth. While the Sybase ASE is running, no other Sybase ASE can re-allocate these physical devices.

### TempDB

TempDB is simply a scratch pad database. It gets recreated when a SQL Server is rebooted. The information held in this database is temporary data. A query may build a temporary table to assist it; the Sybase optimizer may decide to create a temporary table to assist itself.

Since this is an area of constant activity we create this database over a cooked file system which has historically proven to have better performance than raw - due to the buffered writes provided by the Operating System.

### Port Numbers

When creating a new ASE, we allocate a port to it (currently, DBA reserves ports 1500 through 1899 for its use). We then map a host name to the different ports: hera, fddi-hera and so forth. We can actually have more than one port number for an ASE but we typically don't do this.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.2: How to start/stop ASE when CPU reboots

Below is an example of the various files (on *Irix*) that are needed to start/stop an ASE. The information can easily be extended to any UNIX platform.

The idea is to allow as much flexibility to the two classes of administrators who manage the machine:

* The System Administrator
* The Database Administrator

Any errors introduced by the DBA will not interfere with the System Administrator's job.

With that in mind we have the system startup/shutdown file */etc/init.d/sybase* invoking a script defined by the DBA: */usr/sybase/sys.config/{start,stop}.sybase*

## */etc/init.d/sybase*

On some operating systems this file must be linked to a corresponding entry in */etc/rc.0* and */etc/rc.2* -- see *rc0(1M)* and *rc2(1M)*

#!/bin/sh

# last modified: 10/17/95, sr.

#

# Make symbolic links so this file will be called during system stop/start.

# ln -s /etc/init.d/sybase /etc/rc0.d/K19sybase

# ln -s /etc/init.d/sybase /etc/rc2.d/S99sybase

# chkconfig -f sybase on

# Sybase System-wide configuration files

CONFIG=/usr/sybase/sys.config

if $IS\_ON verbose ; then # For a verbose startup and shutdown

ECHO=echo

VERBOSE=-v

else # For a quiet startup and shutdown

ECHO=:

VERBOSE=

fi

case "$1" in

'start')

if $IS\_ON sybase; then

if [ -x $CONFIG/start.sybase ]; then

$ECHO "starting Sybase servers"

/bin/su - sybase -c "$CONFIG/start.sybase $VERBOSE &"

else

<error condition>

fi

fi

;;

'stop')

if $IS\_ON sybase; then

if [ -x $CONFIG/stop.sybase ]; then

$ECHO "stopping Sybase servers"

/bin/su - sybase -c "$CONFIG/stop.sybase $VERBOSE &"

else

<error condition>

fi

fi

;;

\*)

echo "usage: $0 {start|stop}"

;;

esac

## */usr/sybase/sys.config/{start,stop}.sybase*

### start.sybase

#!/bin/sh -a

#

# Script to start sybase

#

# NOTE: different versions of sybase exist under /usr/sybase/{version}

#

# Determine if we need to spew our output

if [ "$1" != "spew" ] ; then

OUTPUT=">/dev/null 2>&1"

else

OUTPUT=""

fi

# 10.0.2 servers

HOME=/usr/sybase/10.0.2

cd $HOME

# Start the backup server

eval install/startserver -f install/RUN\_BU\_KEPLER\_1002\_52\_01 $OUTPUT

# Start the dataservers

# Wait two seconds between starts to minimize trauma to CPU server

eval install/startserver -f install/RUN\_FAC\_WWOPR $OUTPUT

sleep 2

eval install/startserver -f install/RUN\_MAG\_LOAD $OUTPUT

exit 0

### stop.sybase

#!/bin/sh

#

# Script to stop sybase

#

# Determine if we need to spew our output

if [ -z "$1" ] ; then

OUTPUT=">/dev/null 2>&1"

else

OUTPUT="-v"

fi

eval killall -15 $OUTPUT dataserver backupserver sybmultbuf

sleep 2

# if they didn't die, kill 'em now...

eval killall -9 $OUTPUT dataserver backupserver sybmultbuf

exit 0

If your platform doesn't support *killall*, it can easily be simulated as follows:

#!/bin/sh

#

# Simple killall simulation...

# $1 = signal

# $2 = process\_name

#

#

# no error checking but assume first parameter is signal...

# what ya want for free? :-)

#

kill -$1 `ps -ef | fgrep $2 | fgrep -v fgrep | awk '{ print $1 }'`

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.3: How do I move *tempdb* off of the Master Device?

There used to be a section in the FAQ describing how to drop all of tempdb's devices physically from the master device.  This can make recovery of the server impossible in case of a serious error and so it strongly recommended that you **do not** do this but simply drop the segments as outlined below.

### Sybase TS Preferred Method of Moving *tempdb* off the Master Device.

This is the Sybase TS method of removing **most** activity from the master device:

1. Alter tempdb on another device:
2. 1> alter database tempdb on ...

2> go

1. Use the tempdb:
2. 1> use tempdb

2> go

1. Drop the segments:
2. 1> sp\_dropsegment "default", tempdb, master
3. 2> go
4. 1> sp\_dropsegment "logsegment", tempdb, master
5. 2> go
6. 1> sp\_dropsegment "system", tempdb, master

2> go

Note that there is still *some* activity on the master device. On a three connection test that I ran:

while ( 1 = 1 )

begin

create table #x (col\_a int)

drop table #x

end

there was one write per second. Not bad.

### An Alternative

(I recently did some bench marks comparing this method, the previous method and a combination of both.  According to sp\_sysmon there was no difference in activity at all.  I leave it here just in case it proves useful to someone.)

The idea of this handy script is to simply fill the first 2MB of tempdb thus effectively blocking anyone else from using it. The **slight** gotcha with this script, since we're using model, is that all subsequent database creates will also have *tempdb\_filler* installed. This is easily remedied by dropping the table after creating a new database.

This script works because tempdb is rebuilt every time the ASE is rebooted. Very nice trick!

/\* this isql script creates a table in the model database. \*/

/\* Since tempdb is created from the model database when the \*/

/\* server is started, this effectively moves the active \*/

/\* portion of tempdb off of the master device. \*/

use model

go

/\* note: 2k row size \*/

create table tempdb\_filler(

a char(255) not null,

b char(255) not null,

c char(255) not null,

d char(255) not null,

e char(255) not null

)

go

/\* insert 1024 rows \*/

declare @i int

select @i = 1

while (@i <= 1024)

begin

insert into tempdb\_filler values('a','b','c','d','e')

if (@i % 100 = 0) /\* dump the transaction every 100 rows \*/

dump tran model with truncate\_only

select @i=@i+1

end

go

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.4: How do I correct *timeslice -201*

(Note, this procedure is only really necessary with pre-11.x systems.  In system 11 systems, these parameters are tunable using sp\_configure.)

### Why Increase It?

Basically, it will allow a task to be scheduled onto the CPU for a longer time. Each task on the system is scheduled onto the CPU for a fixed period of time, called the timeslice, during which it does some work, which is resumed when its next turn comes around.

The process has up until the value of *ctimemax* (a config block variable) to finish its task. As the task is working away, the scheduler counts down ctimemax units. When it gets to the value of *ctimemax* - 1, if it gets **stuck** and for some reason cannot be taken off the CPU, then a timeslice error gets generated and the process gets infected.

On the other hand, ASE will allow a server process to run as long as it needs to. It will not swap the process out for another process to run. The process will decide when it is "done" with the server CPU. If, however, a process goes on and on and never relinquishes the server CPU, then Server will timeslice the process.

### Potential Fix

1. Shutdown the ASE
2. %buildmaster -d*your\_device* -yctimemax=2000
3. Restart your ASE. If the problem persists contact Sybase Technical Support notifying them what you have done already.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.5: Certified Sybase Professional

There have been changes in the process of becoming a Sybase Certified Professional. There's a very informative link at <http://www.sybase.com/education/profcert>, Professional Certification.

Rob Verschoor has put together some good stuff on his pages ( <http://www.euronet.nl/~syp_rob/certtips.html>) that have pretty much all that you need to know. He also has a quiz which is intended to test each and everyone's knowledge of ASE and RepServer.

Sybase have released some sample questions (look for them at <http://www.sybase.com/education/>). The GUI requires MS Windows (at the time of writing), but they are definitely a sample of what you will be asked. There are also a couple of CDs available with yet more questions on them.

*The Certification Kickback*

There have been a couple of articles recently covering the kickback that seems to be happening as far as certification is concerned. Serveral HR people have said that if a person's CV (resume) is sent in covered in certifications then it goes straight into the bit bucket. I do not know if this is true or not, but one thing that you might wish to consider is the preparation of two CVs, one with certifications, one without. If the job request specifies certification is necessary, then send in the appropriate CV. If it does not specifiy certification, send in the clean version. If you go into the interview for a job that did not specify certifications up front and the interviewer starts going about you not being certificated, you simply produce your card as proof.

# 1.1.6: RAID and Sybase

Here's a short summary of what you need to know about Sybase and RAID.

The newsgroup [comp.arch.storage](news:comp.arch.storage) has a detailed FAQ on RAID, but here are a few definitions:

### RAID

RAID means several things at once. It provides increased performance through disk striping, and/or resistance to hardware failure through either mirroring (fast) or parity (slower but cheaper).

### RAID 0

RAID 0 is just striping. It allows you to read and write quickly, but provides no protection against failure.

### RAID 1

RAID 1 is just mirroring. It protects you against failure, and generally reads and writes as fast as a normal disk. It uses twice as many disks as normal (and sends twice as much data across your SCSI bus, but most machines have plenty of extra capacity on their SCSI busses.)

*Sybase mirroring always reads from the primary copy, so it does not increase read performance.*

### RAID 0+1

RAID 0+1 (also called RAID 10) is striping and mirroring together. This gives you the highest read and write performance of any of the raid options, but uses twice as many disks as normal.

### RAID 4/RAID 5

RAID 4 and 5 have disk striping and use 1 extra disk to provide *parity*. Various vendors have various optimizations, but this RAID level is generally much slower at writes than any other kind of RAID.

### RAID 7

I am not sure if this is a genuine RAID standard, further checking on your part is required.

### Details

Most hardware RAID controllers also provide a battery-backed RAM cache for writing. This is very useful, because it allows the disk to claim that the write succeeded before it has done anything. If there is a power failure, the information will (hopefully) be written to disk when the power is restored. The cache is very important because database log writes cause the process doing the writes to stop until the write is successful. Systems with write caching thus complete transactions much more quickly than systems without.

What RAID levels should my data, log, etc be on? Well, the log disk is **frequently written**, so it should not be on RAID 4 or 5. If your data is **infrequently written**, you could use RAID 4 or 5 for it, because you don't mind that writes are slow. If your data is frequently written, you should use RAID 0+1 for it. Striping your data is a very effective way of avoiding any one disk becoming a hot-spot. Traditionally Sybase databases were divided among devices by a human attempting to determine where the hot-spots are. Striping does this in a straight-forward fashion, and also continues to work if your data access patterns change.

Your tempdb is data but it is frequently written, so it should not be on RAID 4 or 5.

If your RAID controller does not allow you to create several different kinds of RAID volumes on it, then your only hope is to create a huge RAID 0+1 set. If your RAID controller does not support RAID 0+1, you shouldn't be using it for database work.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.7: How to swap a db device with another

Here are four approaches. Before attempting **any** of the following: Backup, Backup, Backup.

#### Dump and Restore

1. Backup the databases on the device, drop the databases, drop the devices.
2. Rebuild the new devices.
3. Rebuild the databases (Make sure you recreate the fragments correctly - See Ed Barlow's scripts (<http://www.tiac.net/users/sqltech/>) for an sp that helps you do this if you've lost your notes. Failure to do this will possibly lead to data on log segments and log on data segments).
4. Reload the database dumps!

#### Twiddle the Data Dictionary - for brave ****experts**** only.

1. Shut down the server.
2. Do a physical dump (using *dd(1)*, or such utility) of the device to be moved.
3. Load the dump to the new device
4. Edit the data dictionary (sysdevices.physname) to point to the new device.

#### The Mirror Trick

1. Create a mirror of the old device, on the new device.
2. Unmirror the primary device, thereby making the \_backup\_ the primary device.
3. Repeat this for all devices until the old disk is free.

#### dd (Unix only)

(This option is no use if you need to move a device now, rather if you anticipate moving a device at some point in the future.)

You may want to use this approach for creating **any** database.

Create (or use) a directory for symbolic links to the devices you wish to use. Then create your database, but instead of going to /dev/device, go to /directory/symlink - When it comes time to move your devices, you shut down the server, simply *dd(1)* the data from the old device to the new device, recreate the symbolic links to the new device and restart the ASE. Simple as that.

**Backups are a requisite in all cases, *just in case*.**

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.8: Server naming and renaming

There are three totally separate places where ASE *names* reside, causing much confusion.

### ASE Host Machine *interfaces* File

A *master* entry in here for server **TEST** will provide the network information that the server is expected to listen on. The -S parameter to the dataserver executable tells the server which entry to look for, so in the RUN\_TEST file, -STEST will tell the dataserver to look for the entry under TEST in the interfaces file and listen on any network parameters specified by 'master' entries.

TEST

master tcp ether hpsrv1 1200

query tcp ether hpsrv1 1200

Note that preceding the **master/query** entries there's a tab.

This is as far as the name **TEST** is used. Without further configuration the server does not know its name is **TEST**, nor do any client applications. Typically there will also be *query* entries under **TEST** in the local *interfaces* file, and client programs running on the same machine as the server will pick this connection information up. However, there is nothing to stop the *query* entry being duplicated under another name entirely in the same *interfaces* file.

ARTHUR

query tcp ether hpsrv1 1200

*isql -STEST* or *isql -SARTHUR* will connect to the same server. The name is simply a search parameter into the *interfaces* file.

### Client Machine *interfaces* File

Again, as the server name specified to the client is simply a search parameter for Open Client into the *interfaces* file, SQL.INI or WIN.INI the name is largely irrelevant. It is often set to something that means something to the users, especially where they might have a choice of servers to connect to. Also multiple query entries can be set to point to the same server, possibly using different network protocols. eg. if **TEST** has the following master entries on the host machine:

TEST

master tli spx /dev/nspx/ \xC12082580000000000012110

master tcp ether hpsrv1 1200

Then the client can have a meaningful name:

ACCOUNTS\_TEST\_SERVER

query tcp ether hpsrv1 1200

or alternative protocols:

TEST\_IP

query tcp ether hpsrv1 1200

TEST\_SPX

query tli spx /dev/nspx/ \xC12082580000000000012110

### sysservers

This system table holds information about remote ASEs that you might want to connect to, and also provides a method of naming the local server.

Entries are added using the sp\_addserver system procedure - add a remote server with this format:

sp\_addserver server\_name, null, network\_name

server\_name is any name you wish to refer to a remote server by, but network\_name must be the name of the remote server as referenced in the interfaces file local to your local server. It normally makes sense to make the server\_name the same as the network\_name, but you can easily do:

sp\_addserver LIVE, null, ACCTS\_LIVE

When you execute for example, exec LIVE.master..sp\_helpdb the local ASE will translate LIVE to ACCTS\_LIVE and try and talk to ACCTS\_LIVE via the ACCTS\_LIVE entry in the local interfaces file.

Finally, a variation on the sp\_addserver command:

sp\_addserver LOCALSRVNAME, local

names the local server (after a restart). This is the name the server reports in the errorlog at startup, the value returned by @@SERVERNAME, and the value placed in Open Client server messages. It can be completely different from the names in RUN\_SRVNAME or in local or remote interfaces - it has **no** bearing on connectivity matters.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.9: How do I interpret the tli strings in the interface file?

The tli string contained with Solaris interface files is a hex string containing port and IP address.  If you have an entry

SYBSRVR  
    master tli tcp /dev/tcp \x000204018196c4510000000000000000

Then it can be interpreted as follows:

x0002    no user interpretation (header info?)  
0401     port number (1025 decimal)  
81       first part of IP address  (129 decimal)  
96       second part of IP address (150 decimal)  
c4       third part of IP address (196 decimal)  
51       fourth part of IP address (81 decimal)

So, the above tli address is equivalent to

SYBSRVR  
    master tcp ether sybhost 1025

where sybhost's IP address is 129.150.196.81.

The following piece of Sybperl (courtesy of Michael Peppler) takes a tli entry and returns the IP address and port number for each server in a Solaris' interfaces file.

#!/usr/local/bin/perl -w

use strict;

my $server;

my @dat;

my ($port, $ip);

while(<>) {

next if /^\s\*$/;

next if /^\s\*\#/;

chomp;

if(/^\w/) {

$server = $\_;

$server =~ s/\s\*$//;

next;

}

@dat = split(' ', $\_);

($port, $ip) = parseAddress($dat[4]);

print "$server - $dat[0] on port $port, host $ip\n";

}

sub parseAddress {

my $addr = shift;

my $port;

my $ip;

my (@arr) = (hex(substr($addr, 10, 2)),

hex(substr($addr, 12, 2)),

hex(substr($addr, 14, 2)),

hex(substr($addr, 16, 2)));

$port = hex(substr($addr, 6, 4));

$ip = join('.', @arr);

($port, $ip);

}

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.10: How can I tell the datetime my Server started?

### Method #1

The normal way would be to look at the errorlog, but this is not always convenient or even possible. From a SQL session you find out the server startup time to within a few seconds using:

select "Server Start Time" = crdate

from master..sysdatabases

where name = "tempdb"

### Method #2

Another useful query is:

select \* from sysengines

which gives the address and port number at which the server is listening.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.11: Raw partitions or regular files?

Hmmm... as always, this answer depends on the vendor's implementation on a cooked file system for the ASE...

## Performance Hit (synchronous vs asynchronous)

If on this platform, the ASE performs file system I/O synchronously then the ASE is blocked on the read/write and throughput is decreased tremendously.

The way the ASE typically works is that it will issue an I/O (read/write) and save the I/O control block and continue to do other work (on behalf of other connections). It'll periodically poll the workq's (network, I/O) and resume connections when their work has completed (I/O completed, network data xmit'd...).

## Performance Hit (bcopy issue)

Assuming that the file system I/O is asynchronous (this can be done on SGI), a performance hit may be realized when bcopy'ing the data from kernel space to user space.

Cooked I/O typically (again, SGI has something called directed I/O which allows I/O to go directly to user space) has to go from disk, to kernel buffers and from kernel buffers to user space; on a read. The extra layer with the kernel buffers is inherently slow. The data is moved from kernel buffers to/from user space using bcopy(). On small operations this typically isn't that much of an issue but in a RDBMS scenario the bcopy() layer is a significant performance hit because it's done so often...

## Performance Gain!

It's true, using file systems, at times you can get performance gains assuming that the ASE on your platform does the I/O asynchronously (although there's a caveat on this too... I'll cover that later on).

If your machine has sufficient memory and extra CPU capacity, you can realize some gains by having writes return immediately because they're posted to memory. Reads will gain from the anticipatory fetch algorithm employed by most O/S's.

You'll need extra memory to house the kernel buffered data and you'll need extra CPU capacity to allow bdflush() to write the dirty data out to disk... eventually... but with everything there's a cost: extra memory and free CPU cycles.

One argument is that instead of giving the O/S the extra memory (by leaving it free) to give it to the ASE and let it do its caching... but that's a different thread...

## Data Integrity and Cooked File System

If the Sybase ASE is **not** certified to be used over a cooked file system, because of the nature of the kernel buffering (see the section above) you may face database corruption by using cooked file system anyway. The ASE **thinks** that it has posted its changes out to disk but in reality it has gone only to memory. If the machine halts without bdflush() having a chance to flush memory out to disk, your database **may** become corrupted.

Some O/S's allow cooked files to have a *write through* mode and it really depends if the ASE has been certified on cooked file systems. If it has, it means that when the ASE opens a device which is on a file system, it fcntl()'s the device to write-through.

## When to use cooked file system?

I typically build my tempdb on cooked file system and I don't worry about data integrity because tempdb is **rebuilt** every time your ASE/SQL Server is rebooted.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.12: Is Sybase Y2K (Y2000) compliant?

Sybase is year 2000 compliant at specific revisions of each product. Full details are available at <http://www.sybase.com/>, specifically (as these links will undoubtedly change):

<http://www.sybase.com/success/inc/corpinfo/year2000_int.html>  
<http://www.sybase.com/Company/corpinfo/year2000_matrix.html>

**Note: Since we have made it to 2000 more or less intact, I see no reason to include this question.  I plan to remove with the next release of the FAQ.  If you feel strongly about leaving it in then let me know.**

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.13 How Can I Run the ASE Upgrade Manually?

### How to Run the ASE Upgrade Manually

This document describes the steps required to perform a manual upgrade for ASE from release 4.x or 10.0x to release 11.02. In most cases, however, you should use sybinit to perform the upgrade.

BE SURE TO HAVE GOOD BACKUPS BEFORE STARTING THIS PROCEDURE.

1. Use release 11.0x sybinit to run the pre-eligibility test and Check Reserved words. Make any necessary changes that are mentioned in the sybinit log. The sybinit log is located in $SYBASE/init/logs/logxxxx.yyy.
2. Use isql to connect to the 4.x or 10.0x ASE and do the following tasks:
3. a. Turn on option to allow updates to system tables:
4. 1> sp\_configure "allow updates", 1
5. 2> go
6. b. Checkpoint all databases:
7. 1> use "dbname"
8. 2> go
9. 1> checkpoint
10. 2> go
11. c. Shutdown the 4.x or 10.0x ASE.
12. 1> shutdown
13. 2> go
14. Copy the interfaces file to the release 11.0x directory.
15. Set the environment variable SYBASE to the release 11.0x directory.
16. Copy the runserver file to the release 11.0x $SYBASE/install directory.
17. Edit the $SYBASE/install/RUN\_SYBASE (runserver file) to change the path from the 4.x or 10.x dataserver directory to the new release 11.0x directory.
18. Start ASE using the new runserver file.
19. % startserver -f$SYBASE/install/RUN\_SYBASE
20. Run the upgrade program:

UNIX: $SYBASE/upgrade/upgrade -S"servername" -P"sapassword" > $SYBASE/init/logs/mylog.log 2>&1 VMS: SYBASE\_SYSTEM[SYBASE.UPGRADE]upgrade /password="sa\_password" /servername="servername"

1. Shut down SQL server after a successful upgrade.
2. % isql -Usa -Pxxx
3. -SSYBASE
4. 1> shutdown
5. 2> go
6. Start ASE using the release 11.0x runserver file.

% startserver -f$SYBASE/install/RUN\_SYBASE

1. Create the sybsystemprocs device and database if upgrading from 4.9.x. You should create a 21mb sybsystemprocs device and database.  
   a. Use the disk init command to create the sybsytemprocs device and database manually, for example:

disk init name = "sybprocsdev", physname="/dev/sybase/rel1102/sybsystemprocs.dat", vdevno=4, size=10752 go To check to see which vdevno is available: type 1> select distinct low/16777216 from sysdevices 2> order by low 3> go A sample create database command: create database sybsystemprocs on sybprocsdev=21 go Please refer to the "Sybase ASE Reference Manual", for more information on these commands.

1. Run the installmaster and installmodel scripts:
2. UNIX: %isql -Usa -Psapassword -i$SYBASE/scripts/installmaster
3. UNIX: %isql -Usa -Psapassword -i$SYBASE/scripts/installmodel
4. VMS: $isql /user="sa" /password="sapass"
5. /input="[sybase\_system.scripts]installm aster"
6. VMS: $isql /user="sa" /password="sapass"
7. /input="[sybase\_system.scripts]installm odel"
8. If you upgraded from ASE 4.9.2, you will need to run sp\_remap to remap the compiled objects. Sp\_remap remaps stored procedures, triggers, rules, defaults, or views to be compatible with the current release of ASE. Please refer to the Reference Manual Volume II for more information on the sp\_remap command.

The syntax for sp\_remap:

sp\_remap object\_name

If you are upgrading to ASE 11.0.x and the upgrade process failed when using sybinit, you can invoke sybinit and choose remap query tress from the upgrade menu screen. This is a new option that is added, after a failed upgrade.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.14 We have lost the sa password, what can we do?

Remember Douglas Adams famous quote *"Don't panic"* is the first thing!

I know that most people use the 'sa' account all of the time, which is fine if there is only **ever** one dba administering the system. If you have more than one person accessing the server using the 'sa' account, consider using sa\_role enabled accounts and disabling the 'sa' account. Funnily enough, this is obviously what Sybase think because it is one of the questions in the certification exams.

If you see that someone is logged using the 'sa' account or is using an account with 'sa\_role' enabled, then you can do the following:

sp\_configure "allow updates to system tables",1

go

update syslogins set password=null where name = 'sa'

go

sp\_password null,newPassword

go

You **must** rememeber to reset the password before exiting isql or sqsh. I thought that setting it to null would be enough, and exited isql thinking that I would be able to get in with a null password. Take it from me that the risk is not worth it. It failed for me and I had to kill the dataserver and get a new password. I just tried the above method and it works fine.

If you have a user with sso\_role enabled, login with that account and change the 'sa' password that way. It is often a good idea to have a separate site security officer, just to get you out of this sticky situation. Certainly stops you looking an idiot in managements eyes for having to reboot production because you have locked yourself out!

OK, so we have got to the point where there are no accounts with sufficient priviledges to allow you to change the 'sa' account password. (You are sure about that, since the next part can cause data loss, so have another quick look.) We now need to some more drastic stuff.

If the server is actually running, then you need to stop it.  We know that the only accounts that can stop the server in a nice manner are not available, so it has to be some sort of kill. You can try:

kill -SIGTERM

or

kill -15

(they are identical) which is designed to be caught by ASE, which then performs the equivalent of shutdown with nowait. If ASE does not die, and you should give it a little while to catch and act on the signal, then you might have to try other measures, which is probably kill -9. Note that if you have tables with identity columns, most of these will jump alarmingly, unless you are using ASE 12.5 and the identity interval is set to 1.

Once down, edit the RUN\_SERVER file ( RUN\_SERVER.bat on NT) and add "-psa" (it is important not to leave a space between the"-p" and the "sa", and that it is all lower-case) to the end of the dataserver or sqlsrvr.exe line. You will end up with a file that looks a bit like:

#!/bin/sh

#

# Adaptive Server name: N\_UTSIRE

# Master device path: /data/sybase/databases/N\_UTSIRE/master.dat

# Error log path: /opt/sybase-11.9.2/install/N\_UTSIRE.log

# Directory for shared memory files: /opt/sybase-11.9.2

#

# Regenerate sa password -psa

#

/opt/sybase-11.9.2/bin/dataserver \

-sN\_UTSIRE \

-d/data/sybase/databases/N\_UTSIRE/master.dat \

-e/opt/sybase-11.9.2/install/N\_UTSIRE.log \

-M/opt/sybase-11.9.2 -psa \

(I add the line mentioning the regenerate, so that if I need to do this in a moment of extreme pressure it is there in front of my nose.

Now, start the server again and you should see the following on the screen:

00:00000:00001:2001/05/26 18:29:21.39 server    'bin\_iso\_1' (ID = 50)

00:00000:00001:2001/05/26 18:29:21.39 server  on top of default character set:

00:00000:00001:2001/05/26 18:29:21.39 server    'iso\_1' (ID = 1).

00:00000:00001:2001/05/26 18:29:21.39 server  Loaded default Unilib conversion handle.

New SSO password for sa:tmfyrkdwpibung

Note that it is **not** written to the log file, so keep your eyes peeled.

On NT you will have to start the server from the command line and not use Sybase Central or the control panel.

Obviously, you will want to change the password to something much more memorable as soon as possible.

Remember to remove the "-psa" from the "RUN" file before you start the server again or else the password will be changed again for you.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.15 How do I set a password to be null?

Since ASE 11 (I cannot remember if it was with the very first release of 11, but certainly not before) the password column in syslogins has been encrypted. Setting this column to NULL does not equate to that login having a NULL password. A NULL password still requires the correct binary string to be in place.

In release 12 and above, set the minimum password length to be 0 using sp\_configure and give that account a null password, and all should be fine.

Before 12, it is not possible to set the minimum password length, so the direct approach is not possible. So, update the relevant record in syslogins setting the password column to be the same as that of an account with a NULL password already.

How does one get the correct binary value? When a new ASE is built, the 'sa' account has a NULL password to start with. Setting an account to have the same binary value as such an 'sa' account should work. Remember that the binary string is going to be specific to the operating system and the *exact* release of ASE etc. Obviously, if you have set the password of your 'sa' accounts to be something other than NULL (sensible move), then you are going to have to build yourself a dummy server just to get the correct string. If this is important to you, then you may wish to store the value somewhere safe once you have generated it.

Yet another method would be to simply insert the correct hex string into the password column. Rob Verschoor has a very nice stored proc on his site called sp\_blank\_password to allow you to do just this. Go to <http://www.sypron.nl/blankpwd.html>.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.16: Does Sybase support Row Level Locking?

With Adaptive Server Enterprise 11.9 Sybase introduced row level locking into its product.  In fact it went further than that, it introduced 3 different locking levels:

* All Pages Locking

This is the scheme that is implemented in all servers prior to 11.9.   Here locks are taken out at the page level, which may included many rows.  The name refers to the fact that all of the pages in any data manipulation statement are locked, both data and index.

* Data Page Locking

The other two locking schemes are bundled together under the title Data Page Locking, refering to the fact that only data pages are ever locked in the conventional sense.   Data Page Locking is divided into two categories

* Data Only Locking

This locking scheme still locks a page at a time, including all of the rows contained within that page, but uses a new mechanism, called latches, to lock index pages for the shortest amount of time.  One of the consequences of this scheme is that it does not update index pages.  In order to support this Sybase has introduced a new concept, forwarded rows.  These are rows that have had to move because they have grown beyond space allowed for them on the page they were created. 2002 bytes per page.

* Row Level Locking

Just as it sounds, the lock manager only locks the row involved in the operation.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)

# 1.1.17: What platforms does ASE run on?

Sybase has an excellent lookup page that tells you all of the releases that Sybase has certifies as running on a particular platform. Got to <http://ohno.sybase.com/cgi-bin/ws.exe/cert/ase_cert.hts>.

[**Back to top**](http://www.isug.com/Sybase_FAQ/ASE/section1.1.html#BasicAdmin)