Global Information Assurance Certification Paper

Name: Chuck Ellis
Assignment Version: 1.4
Title: Disaster Recovery for a Sybase Data Server – A Case Study

Abstract

This case study outlines the process of developing and implementing a disaster recovery plan for a production Sybase data server. The Sybase data server is an essential part of the organizations infrastructure, supporting several crucial Internet and intranet applications. The systems environment before, during, and after implementation of the disaster recovery plan is discussed in detail.

The disaster recovery plan was developed to ensure full business continuity in the event of a major disaster. The primary goal was to ensure that all production database systems were operational within 24 hours of arrival at a predetermined disaster recovery site.

The disaster recovery plan provides systematic instructions for recovering the organizations databases. The plan has been successfully tested and evaluated, and will continue to be tested and evaluated twice a year during regularly scheduled off-site disaster recovery drills. Most importantly, the plan helps provide consistent success in recovery of the organization's databases.

BEFORE SNAPSHOT

> When a database server fails and critical business applications are unavailable, users sit idle, and e-commerce quickly grinds to a halt. Revenues that flow through these applications disappear and customer relationships are impacted. Not surprisingly, many businesses are taking database and data-center availability very seriously.[i]

In today's information-technology climate, there is little doubt that a solid, reliable disaster recovery plan is a necessity for any organization in meeting its business goals. Security professionals will often focus on proactive measures, in an effort to stop security problems before they start. This may lead security professionals to underestimate the admittedly reactive task of performing a disaster recovery. However, while performing a disaster recovery is reactive, effectively preparing for a disaster is proactive. Even though most businesses recognize the need, it has been estimated that only 25% of fortune 1000 companies have disaster recovery plans in place[ii].

Scope

The organization began an overall effort to improve all of its disaster recovery capabilities, which included recovery of Sun and Windows servers as well as the Sybase data server. This case study focuses primarily on the recovery process

of the Sybase data server. While the case study refers to the recovery of the Sun server, the specifics of recovering the Sun host are beyond the scope of this document.

Environment

- Database Server: Sybase Adaptive Server Enterprise, Version 12.5.0.1
- Host Server: Sun Microsystems SunFire 6800
- Host Operating System: Solaris 8

Problem Statement

The problem examined by this case was that a comprehensive database-related disaster recovery plan had not been implemented within the organization. System recovery had been considered in the past, but a full-fledged and documented plan had never been developed.

To its credit, the organization had engaged in off-site disaster recovery drills for several years. Unfortunately, the database portion of the drills was often unsuccessful due to the lack of a documented, repeatable process. Most of the knowledge required to restore the data server and applications existed only in the minds of a few key organization staff. The simple fact that there was no written documentation made the success of each disaster recovery drill completely unpredictable.

Vulnerabilities

As with most companies, the organizations vulnerabilities come from many sources. Hackers (from both outside and inside the organization[iii]), fire, terrorist activity, disgruntled employee revenge, and simple operator error are just a few of the potential vulnerabilities from which the organization must protect itself. New vulnerabilities, such as SQL injection attacks, were learned after attending the SANS Security Essentials course and browsing the SANS reading room[iv]. The organization is especially vulnerable to SQL Injection attacks because its databases support externally accessible web-based applications. A well-informed hacker could conceivably wipeout the database or cause corruption within the database using SQL injection.

Risks

The risks of not having a well-documented disaster recovery plan are considerable. Many of the above stated vulnerabilities can result in a complete loss of data, and thus an inability to support the customer base and achieve business goals. Unrecoverable data loss can lead to devastating financial ruin for many corporations. Companies that cannot quickly recover from a disaster run the risk experienced by Grafix Softech, F.A., a relatively small online casino

company that was recently targeted by a group of Russian hackers[v]. The hackers encrypted all of Softech's data and demanded a ransom for the encryption key. The company paid the ransom but unfortunately, the key didn't decrypt all of the data. A consulting service was hired and managed to recover the data, but it took several days and a 35,000-dollar fee. The company's website was down for 9 days at an estimated cost of 75,000 dollars per day. This puts the total cost to the company at roughly 700,000 dollars, not including the amount of the undisclosed ransom payout. These costs could have been significantly reduced, if not eliminated, had the company implemented a disaster recovery plan.

Goals

The primary goal for the disaster recovery plan was to provide an operational database within 24 hours of the arrival at a predetermined disaster recovery site.

Secondary goals included:

- The plan should be repeatable.
- The plan should minimize the need for plan modifications as the application environment changed over time.
- The plan steps should require minimal human interaction.
- The plan should be simple enough that users of moderate technical ability could follow the steps and recover the databases required for applications.
- All recovery scripts should be backed up on a daily basis and should be available on the recovery host after a system restore.
- The plan should be posted to the internal documentation website.
- The plan should include the expected duration of each task.

DURING SNAPSHOT

The disaster recovery plan was written such that users of moderate technical ability can follow the steps and recover the application databases. The plan was not intended to explain the underlying steps that the recovery scripts perform. However, for troubleshooting purposes a technical user should be able to view and edit the recovery scripts in the event of an unforeseen problem. Expected durations were included in order to provide some idea of how long each step might reasonably last.

Methodology

The chosen method for developing the disaster recovery plan was to perform a test database recovery on excess hardware and document throughout the process. Several non-DBA users reviewed and tested the process, making clarifications where needed. This helped achieve one of the secondary goals, that the directions be understandable to users of moderate technical ability.

A daily cron job was created to minimize changes to the plan as the application environments changed. This cron job executes SQL commands that in turn create valid SQL scripts used for creating and configuring the database. The scripts are always valid and accurate since the cron job runs daily. Additionally, these scripts are saved during each days backups and thus are always available if a server needs to be restored. The code for these scripts is included in the corresponding sections or the appendix. The scripts are saved in the /home/sybase/scripts/disaster_recovery directory.

After several iterations of the recovery process, the following distinct recovery steps were identified and documented.

- General Information
- Validate Prerequisites
- Delete Entries from the Interfaces file
- Backup the Existing Sybase Configuration File
- Install the Sybase Server
- Localize the Sybase Server
- Set the SA Password
- Create the Database Devices
- Create the Databases
- Create the Server Logins
- Configuration Changes to the Interfaces File
- Load the Databases
- Install jConnect
- Restart the Database Server

General Information

- Except where noted perform all steps while logged in as the Sybase user.
- All durations are estimated based upon a Sunfire 6800 server. Slower machines will yield slower durations for various recovery steps.
- All scripts are located in the /home/sybase/scripts/disaster_recovery directory.

Validate Prerequisites

Several prerequisites must be met before a database recovery can start. These prerequisites should be performed by the UNIX administrator.

- Specific files and directories must be restored to the recovery host server:
    - System files (e.g. /etc/system/*)
    - Sybase file system (This is the $SYBASE directory and all subdirectories),

- Application-code file systems and system files must all be restored to the recovery host before the database recovery steps can be executed. These files are backed up on a daily basis. During any disaster recovery, these files would be restored by a UNIX administrator. The organization uses Legato Networker to back up and restore these files.

- The $SYBASE and $SYBASE_ASE environment variables must be set. These will most likely have been set for the Sybase user by the /etc/.profile or the Sybase users local .profile.

- Existence of raw devices
  - The UNIX administrator must create the raw devices used by Sybase. These raw devices should use the same names as used in production in order to allow the disk-init script, which is executed in the "Create the Database Devices" section, to function without modification.

- Ensure that the error log directory for the Sybase server exists.

- Set the $DISPLAY environment variable so that GUI applications can be executed. This is to enable the use of Sybase's DSEDIT tool in the "Delete Entries from the Interfaces File" step. If an X Windows display is not available, it is possible to use Sybase's DSCP application instead of DSEDIT.

Delete entries from the interfaces file
Approximate Duration: < 5 minutes

Entries for the data server must be deleted from the interfaces file. This is necessary because the server install will fail if the installation program thinks that the server names or ports are already in use. As the sybase user, login to the host machine and type the following:

```
cd $SYBASE
```

Create a backup copy of the interfaces file by typing the following:

```
cp interfaces interfaces.bak
```

Next, edit the interfaces file and remove the entry (usually 3 or 4 lines) for the data server that is being recovered. Also remove the entry for the database backup server, SYB_BACKUP, which is the three lines starting at the line 'SYB_BACKUP'. Save the file when complete.

Backup the existing Sybase Configuration File
Approximate Duration: < 1 minute

The existing Sybase configuration file will be overwritten when the server is created so make a backup copy. This file can later be used as a reference when configuring the data server. Type the following command from the $SYBASE/$SYBASE_ASE directory:

cp *server_name*.cfg *server_name*.cfg.bak

where *server_name* is the name of the Sybase data server that is being recovered.

Install the Sybase Server and Backup Server
Approximate Duration: < 10 minutes

Create the data server and backup server using resource files and the srvbuildres program. The resource files are located in $SYBASE/$SYBASE_ASE/init/logs/. The following command will display the resource files:

ls -ltr $SYBASE/$SYBASE_ASE/init/logs/srvbuild*.rs

The correct resource file will be the most recent file, by file date, that starts with "srvbuild" and ends with "*server_name*.rs", where *server_name* is the name of the data server being recovered. Verify that the values in the resource file are correct by editing the file. If the UNIX system administrator created the Sybase raw devices in a different location, it will be necessary to change the paths to the raw devices. To do this, edit the resource file and verify that the paths listed for master_device_physical_name and sybsystemprocs_device_physical_name are correct.

Next, create the server by running the following command:

$SYBASE/$SYBASE_ASE/bin/srvbuildres –r *resource_file*

Now create the Sybase backup server using its resource file. The correct backup server resource file will be the most recent file, by file date, that starts with "srvbuild" and ends with "*backup_server_name*.rs", where *backup_server_name* is the name of the backup server being recovered. Run the following command to create the backup server.

$SYBASE/$SYBASE_ASE/bin/srvbuildres –r *resource_file*

Localize the Sybase Server
Approximate Duration: < 10 minutes

Next, localize the data server using the localization resource file and the sqllocres program. Use the most recent file that starts with “sqlloc” and ends with “.rs” as the *loc_resource_file*. The following command will display the resource files:

```
ls -ltr $SYBASE/$SYBASE_ASE/init/logs/sqlloc*.rs
```

To localize the server run the following command:

```
$SYBASE/$SYBASE_ASE/bin/sqllocres –r loc_resource_file.
```

Set the SA Password
Approximate Duration: < 1 minute

When the data server is first created, the SA password will be null. Run the following command to login to the server using the Sybase isql utility:

```
isql –Satlas –Usa –P
```

Once logged in run the following SQL to set the password:

```
sp_password null, password
go
```

where *password* is the SA password of the data server being recovered. As specified in the SANS security essentials class, the password should be at least 8 characters long and contain mixed-case letters and numbers.

Create the Database Devices
Approximate Duration: < 10 minutes

The following SQL code is executed each day, via cron, and the output sent to a file called /home/sybase/scripts/disaster_recovery/disk_init.sql.

```
Select      'disk init name = ' + '"' + name + '"' + ',' + 'physname = ' + '"'
+ phyname + '"' + ',' + 'vdevno = ' + convert(char(2),low/16777216) + ',' +
'size = ' + convert(char(20), (high - low + 1))
from        master..sysdevices d
where       low/16777216 > 1
and         name not like 'sysaudit%'
order by    convert(numeric(2),low/16777216)
go
```

The disk_init.sql file will contain several disk init statements similer to the one shown here:

disk init name='systemdbdev', physname='/dev/vx/rdsk/dg/sybsystemdb', vdevno = 2 , size = 4096

Verify that the path listed for "physname" is correct for each disk init statement. Then run the script using the following command:

isql –S*server_name* –Psa -i /home/sybase/scripts/disaster_recovery/disk_init.sh

Where *server_name* is the data server being recovered. You will be prompted for the SA password, which was set in an earlier step.

Create the Application Databases
Approximate Duration: < 5 minutes

The following SQL code is executed each day, via cron, and the output sent to a file called /home/sybase/scripts/disaster_recovery/create_database.sql.

exec sybsystemprocs.dbo.pr_dbo_create_db_sql

This command will execute a stored procedure that returns "create database" SQL statements for every database on the server. The stored procedure text for pr_dbo_create_db_sql is included in Appendix A.

Edit the /home/sybase/scripts/disaster_recovery/create_database.sql file and change the entry for the tempdb database to "alter database" rather than "create database". This is necessary because the srvbuildres program will have already created tempdb. It is also necessay to remove the "for load" command at the end of the tempdb database entry since it will not be loaded.

After modifying the script, run it by typing:

isql –S*server_name* –Psa -i /home/sybase/scripts/disaster_recovery/create_database.sh

Where *server_name* is the data server being recovered. You will be prompted for the SA password, which was set in an earlier step.

Create the Server Logins
Approximate Duration: < 1 minute

The following shell script is executed each day, via cron, and the output sent to a file called /home/sybase/scripts/disaster_recovery/ syslogins.dat.

bcp master..syslogins out syslogins.dat -n -U$USER_NAME -P$SA_PASSWD -S$SERVER_NAME

The data file is used to restore the login information for all database users. Run the following command to create the database server logins:

bcp master..syslogins in syslogins.dat -F 3 -n –S*server_name* -Usa

In the above command, the –F flag tells the BCP utility to skip the first two records, which will be the SA and probe accounts. These two accounts will already exist on the new Sybase data server and thus do not need to be loaded again.

Configuration Changes to the Interfaces File
Approximate Duration: < 10 minutes

The next step is to set the interfaces file entry for the recovered server so that it point to the current host machine or IP address. Change the current directory to the $SYBASE directory. To start the dsedit application type the following command at the command prompt:

dsedit

Use the dsedit application to modify the entry for the server that is being recovered. Next, continue using the dsedit application to add an entry for SYB_BACKUP.

Load the Databases
Approximate duration of loading each database: 1 GB every 2 minutes

At this point, we are ready to load the application databases. For this purpose, a perl script was created that automatically loads any database. The perl script is configured with the specific location of the database dump files. The perl script must be modified if the database dump files are restored to a different directory. The code for the perl script is included in appendix B. Execute the following script to load a database:

/home/sybase/scripts/disaster_recovery/load_db.pl *database_name*

where *database_name* is the name of the database that is being loaded.

The progress of the database load can be monitored by tailing the backup server log.

Install jConnect
Approximate Duration: < 5 minutes

Since the organizations applications require jConnect, it must to be restored. Run the following script:

```
isql -n –Sserver_name -Usa –i$SYBASE/jConnect/sp/sql_server.sql -o/home/sybase/scripts/disaster_recovery/install_jConnect.log
```

where *server_name* is the name of the server being restored.

Restart the Database Server
Approximate Duration: < 15 minutes

Shutdown the database server by from an isql session by executing:

```
SHUTDOWN
go
```

Start the database server by executing the standard database startup script, which exists in the $SYBASE/$SYBASE_ASE/install directory.

```
/home/sybase/scripts/disaster_recovery/reboot_db_server.sh
```

Once the above commands are complete, the Sybase database server should be up with a complete copy of data.

AFTER SNAPSHOT

The development of the disaster recovery plan has ensured that the organization is much better prepared to recover quickly from a database disaster. Even so, vulnerabilities and risks are still present.

Remaining Vulnerabilities

It is important to note that a disaster recovery plan does not necessarily eliminate vulnerabilities. This is due, in part, because recovering from a disaster is a reactive task. Because vulnerabilities still exist, the concept of "defense in depth", as taught in the SANS Security Essentials class, should be employed. Ideally, the organizations vulnerabilities will be mitigated by numerous security strategies and the disaster recovery plan will be called into action only if these security strategies are breached.

Remaining Risks

The true benefit of a good disaster recovery plan is that it significantly reduces the risks that an organization faces from a disaster event. The existence of the plan puts the organization in a much more secure position to provide service to

it's customers and accomplish it's business goals. A severe disaster event that might previously have resulted in a week or more of downtime would now cause downtime of a single day.

Conclusion

A disaster recovery plan is only one piece to creating a more secure environment. Nevertheless, disaster recovery is an essential part of the organization's overall security posture. By employing a comprehensive strategy of "defense in depth" and perhaps a bit of luck, the only time this plan will need to be used is at the bi-annual disaster recovery drill.

Appendix A: SQL Code for the pr_dbo_create_db_sql Stored Procedure

```
create proc dbo.pr_dbo_create_db_sql
as

declare @db_name  char(15)
declare @hold_name  char(15)
declare @hold_log  char(1)
declare @hold_alter  char(1)
declare @dev_name  char(15)
declare @frag_size char(8)
declare @seg_no  int
declare @frag_cnt int
declare @log_cnt int
declare @db_cnt int
declare @alter_cnt int
declare @seg_cnt int
declare @db_lstart int
declare @msg varchar(30)

select  @hold_name = ' '
select  @hold_alter = ' '
select  @hold_log = ' '
select  @frag_cnt = 0
select  @log_cnt = 0
select  @alter_cnt = 0
select  @db_cnt = 0

create table tempdb..usages
       (
       db_db_name char(15),
       db_dev_name char(15),
       db_frag_size char(8),
       db_seg_no int
       )

declare user_tables cursor for
select  convert(char(15),db_name(dbid)) ,
       convert(char(15),d.name),
       convert(char(8), size/ 512),
       segmap,
       lstart

from    master..sysusages u,
       master..sysdevices d
where   vstart/16777216 >  1
```

```
and     vstart between low and high
and     d.status <> 16
order by        1, lstart

open user_tables

fetch user_tables
into    @db_name,
        @dev_name,
        @frag_size,
        @seg_no,
        @db_lstart

if      (@@sqlstatus = 2)
        begin
                select  @msg = 'no tables today'
                print   @msg
                close   user_tables
        end

while   (@@sqlstatus = 0)

        begin
                insert into     tempdb..usages
                values          (
                                @db_name,
                                @dev_name,
                                @frag_size,
                                @seg_no
                                )

                fetch   user_tables
                into    @db_name,
                        @dev_name,
                        @frag_size,
                        @seg_no,
                        @db_lstart
        end

close   user_tables

declare db_info cursor for
select  tempdb..usages.db_db_name,
        tempdb..usages.db_dev_name,
        tempdb..usages.db_frag_size,
        tempdb..usages.db_seg_no
```

```
from    tempdb..usages

open db_info

fetch   db_info
into    @db_name,
        @dev_name,
        @frag_size,
        @seg_no

if      (@@sqlstatus = 2)
        begin
                select @msg = 'no tables today'
                print @msg
                close user_tables
        end

while   (@@sqlstatus = 0)
        begin
                if      (@db_name != @hold_name and @db_cnt > 0)
                        begin
                                select  @frag_cnt = 0
                        end

                if      (@seg_no != 4 and @hold_log = 'Y')
                        begin
                                select  @hold_log = 'N'
                                select  @log_cnt = 0
                        end

                if      (@seg_no != 3 and @seg_no < 8 and @hold_alter = 'Y')
                        begin
                                select @hold_alter = 'N'
                        end

                if      (@db_name != @hold_name and @db_cnt > 0)
                        begin
                                select  'for load'
                        end

                if      (@db_name != @hold_name)
                        begin
                                select  @alter_cnt = 0
                                select  @log_cnt = 0
                                select  @frag_cnt = 0
```

```
                        select  @frag_cnt = @frag_cnt + 1
                        select  @db_cnt = @db_cnt + 1
                        select  @hold_name = @db_name
                        select  @hold_alter = 'N'
                        select  @hold_log = 'N'
                        select  'create database ' + @db_name  , ' on ', @dev_name
+ ' = ' +  @frag_size
                    end
                else
                if      (@db_name = @hold_name and @frag_cnt > 0 and (@seg_no =
3

        or @seg_no > 7) and @hold_alter = 'N')
                    begin
                        select  @frag_cnt = @frag_cnt + 1
                        select  @alter_cnt = @alter_cnt + 1
                        select  @hold_alter = 'Y'
                        select  'for load'
                        select  'alter database ' + @db_name  , ' on ', @dev_name +
' = ' +  @frag_size
                    end
                else
                if      (@db_name = @hold_name and @frag_cnt > 1 and (@seg_no =
3 or @seg_no > 7) and @hold_alter = 'Y')
                    begin
                        select  @frag_cnt = @frag_cnt + 1
                        select  @alter_cnt = @alter_cnt + 1
                        select  ', ', @dev_name + ' = ' +  @frag_size
                    end

                if      (@db_name = @hold_name and @frag_cnt > 0 and @seg_no = 4
and @log_cnt = 0)
                    begin
                        select  @frag_cnt = @frag_cnt + 1
                        select  @log_cnt = @log_cnt + 1
                        select  @hold_log = 'Y'
                        select  'log on ', @dev_name + ' = ' +  @frag_size
                    end
                else
                if      (@db_name = @hold_name and @frag_cnt > 0 and @seg_no = 4
and @log_cnt >= 0 and @hold_log = 'Y')
                    begin
                        select  @frag_cnt = @frag_cnt + 1
                        select  @log_cnt = @log_cnt + 1
                        select  @hold_alter = 'N'
                        select  @hold_log = 'Y'
```

```
                    select  ',', @dev_name + ' = ' +  @frag_size
                end

            fetch   db_info
            into    @db_name,
                    @dev_name,
                    @frag_size,
                    @seg_no
end

close db_info

select 'for load'

drop table tempdb..usages
```

Appendix B: Perl Code for load_db.pl

```
#!/usr/local/bin/perl5 -w

use Sybase::CTlib;

$DB_user = "sa";
$ DB _password = "password";
# Replace password with the actual sa password.
$ DB _server = "server_name";
# Replace server_name with the actual server name.

$fl_d = new Sybase::CTlib $DB_user, $ DB _password, $ DB _server;

my $curdb;
if(defined $ARGV[0])
{
	$curdb = $ARGV[0];
} else {
	print "Error no database name passed in as an argument\n";
	exit 1;
}

$use_db = "use $curdb ";
$use_master = "use master ";
$checkpoint = "checkpoint ";

$ref = $fl_d->ct_sql("set textsize 1000000");

ct_callback(CS_SERVERMSG_CB, "srv_cb");

my $time = localtime;
print $time, "\n";
&validate_db;
&set_db_option ($curdb, "dbo use only", "true");
&kill_db_processes;
&set_db_option ($curdb, "single user", "true");
&load_db;
&bring_db_online;
&set_db_option ($curdb, "dbo use only", "false");
&set_db_option ($curdb, "single user", "false");
$time = localtime;
print $time, "\n";

sub validate_db
{
```

```
        my $sql_line = "select name from master..sysdatabases where name =
'$curdb' ";
    print "select line is:\n $sql_line\n";
    $ref = $fl_d->ct_sql($sql_line);
    print "validate_db results are:\n";
    foreach $line (@{$ref})
    {
        print join("\t", @{$line}), "\n";
        if ($line->[0] eq $curdb)
        {
                return 1;
        }
        else
        {
                print "db does not match\n";
        }
    }
    print "error db not found\n";
    exit 0;
}

sub set_db_option
{
        my ($dbname, $option_name, $flag, @stuff) = @_;

        print "$use_master\n";
    $ref = $fl_d->ct_sql($use_master);
        &print_db_res($ref);
    my $sql_line = "EXEC sp_dboption '$dbname','$option_name', $flag  ";
    print "select line is:\n $sql_line\n";
    $ref = $fl_d->ct_sql($sql_line);
    &print_db_res($ref);
    print "$use_db\n";
    $ref = $fl_d->ct_sql($use_db);
        &print_db_res($ref);
    print "$checkpoint\n";
    $ref = $fl_d->ct_sql($checkpoint);
    &print_db_res($ref);
}
sub print_db_res
{
        my ($refobj, @s) = @_;
        foreach $line (@{$refobj})
        {
                print join(", ", @{$line}), "\n";
        }
```

```
}

sub kill_db_processes
{
        my $sql_line = "sp_who";
        print "select line is:\n $sql_line\n";
    $ref = $fl_d->ct_sql($sql_line);
    print "results are:\n";
    foreach $line (@{$ref})
    {
#               print $line->[1], "\t", $line->[7], "\n";
        if ((defined $line->[7]) and ($line->[7] eq $curdb))
        {
                #print "Killing ", $line->[1], "\n", join("\t", @{$line}), "\n";
                my $kill_line = "kill $line->[1] ";
                print $kill_line, "\n";
                $ref = $fl_d->ct_sql($kill_line);
        }
#       print join("\t", @{$line}), "\n";
    }
    print "Done reading sp_who\n";
}

sub load_db
{
        print "$use_master\n";
    $ref = $fl_d->ct_sql($use_master);
    ##############################################
    #Specify the location of database dump files here:
    my $dir = "/opt/tools/Sybase/dumps";
    ##############################################
    opendir(DIRHAN, $dir);
    my @all_files = readdir(DIRHAN);
    closedir(DIRHAN);
    ##############################################
    #Replace “diskdump” with a common characteristic
    # of your database dump file names here:
    my @files = grep(/^diskdump/, @all_files);
    ##############################################
    print "files are: \n\t", join("\n\t", @files), "\n";
        my $sql_line = "load database $curdb from '$dir/";
        $sql_line .= join("'\nstripe on '$dir/", sort(@files))."'";
        print "$sql_line\n";
    $ref = $fl_d->ct_sql($sql_line);
}
```

```
sub bring_db_online
{
        print "$use_master\n";
    $ref = $fl_d->ct_sql($use_master);

        my $sql_line = "online database $curdb ";
    print "$sql_line\n";
    $ref = $fl_d->ct_sql($sql_line);
}
```

END NOTES

[i] "Disaster Recovery Package for Sybase Adaptive Server Enterprise". 27 Jun 2002. URL: http://www.sybase.com/detail?id=1019939 (25 March 2003)
[ii] Boroski, Doran. "Protect processes as well as investments in disaster recovery plans". Computerworld. 15 Feb 2002. URL: http://www.computerworld.com/securitytopics/security/recovery/story/0,10801,68345,00.html (26 Mar 2003)
[iii] Wilson, Zachary. "Hacking: The Basics". April 4, 2001. URL: http://www.sans.org/rr/hackers/hack_basics.php (24 Mar 2003)
[iv] McDonald, Stuart. "SQL Injection: Modes of Attack, Defence, and Why It Matters". 18 Jul 2002. URL: http://www.sans.org/rr/appsec/SQL_injection.php (24 Mar 2003)
[v] "Crucial Data Rescued After Hacker Raid: Key Server Stripped : 'It Was Akin to Hacking Into The Pentagon'". National Post's Financial Post & FP Investing. 24 Feb 2003. URL: http://www.ds-osac.org/view.cfm?KEY=7E4454404050&type=2B170C1E0A3A0F162820 (25 Mar 2003)