

Backup and Recovery

What Backup, Recovery, and Disaster Recovery Mean to Your SQL Anywhere Databases

CONTENTS

Introduction	3
Terminology and concepts	3
Database files that make up a database	3
Client-side and server-side backup	4
Online backup, offline backup, and live backup	4
Full and incremental database backup	4
Developing a backup and recovery strategy	4
Understand what backup and recovery means to your business	4
Decide if high availability is right for your business	5
Management commits time and resources for the project	5
Develop, test, time, document, health check, deploy, and monitor	6
Beware of external factors that affect recovery	6
Protect database backups by performing health checks	6
Create a scheduled automated backup plan	7
Backup and recovery strategy example	7
Time the basic backup and recovery operations	8
Estimate a total recovery time from failure	8
What do you do when you encounter errors during recovery?	9
Recovery from the previous day's backup	9
Additional information regarding recovery	10
Secondary backup issues	10
Backup and recovery commands	10
Test your commands	10
Directories used	10
Creating backups	11
Recovering backups	11
Perform health checks	12
Create a post-recovery backup	12
Allow user to access the database	12
Creating a scheduled automated backup plan	12
Disaster recovery	16
Additional information regarding disaster recovery planning	16
Security concerns	17
Summary	17

INTRODUCTION

The development of a backup and recovery strategy is an exercise carried out in phases. To work through these phases you must understand some database terminology such as backup and recovery, the difference between a full database backup and an incremental database backup, and the meaning of an online, offline, and live backup.

This paper describes the phases in developing a backup and recovery plan and cites an example. It shows how time becomes an important factor in database recovery. It also explains what to do when the backed up database fails to restore from your backup medium, and discusses what database health checks should be performed to ensure the database and log files are valid.

An example of database backup and recovery commands is provided to illustrate the statements used to backup and recover a database. Finally, this paper discusses key points in developing a disaster recovery strategy in the event that the physical computer running your database is no longer available.

TERMINOLOGY AND CONCEPTS

Oftentimes, backup and recovery are thought of as one topic, while disaster recovery is thought of as another.

Backup is a process used to make a copy of the contents of database files and log files. The database files consist of a database root file, log file, mirror log file, and other database files called dbspaces.

Recovery is a sequence of tasks performed to restore a database to some point in time. Recovery is performed when either a hardware or media failure occurs. Hardware failure is a physical component failure in your computer, such as a disk drive, controller card, or power supply. Media failure is a failure of the actual physical medium that is storing the data. If a hardware failure occurs, such as a power supply malfunction, the data becomes unavailable. Once the faulty component is replaced, the data is accessible again. If a media failure occurs, such as bad sectors forming on the disk, the component cannot simply be replaced because the actual storage media has been corrupted. You *must* have backups on secondary storage to be able to recover from these failures.

Before you begin recovery, it is a good practice to back up the failing database. Backing up the failing database preserves the situation, provides a safe location so files are not accidentally overwritten, and if unexpected errors occur during the recovery process, Sybase Technical Support may request that you forward these files to them.

Disaster recovery differs from a database recovery scenario because the operating system and all related software must be recovered before any database recovery can begin.

FILES THAT MAKE UP A DATABASE

SQL Anywhere databases consist of disk files that store data. When you create a database using Sybase Central, the CREATE DATABASE statement, or the dbinit command line utility, a main database file or **root file** is created. The default database file is created as *database-name.db*. This main database file contains database tables, system tables, and indexes.

Additional database files expand the size of the database and are called **dbspaces**. A dbspace contains tables and indexes, but not system tables. A dbspace can be created using Sybase Central or Interactive SQL by issuing a CREATE DBSPACE command. By default, the dbspace file created is in the format of *drive:\path\dbspace-name.db*. You can find out if your database is using one or more dbspaces by opening the DBSpaces folder in Sybase Central or querying the SYS.SYSFILE system table, which has a row for each dbspace.

A **transaction log** is a file that records database modifications. Database modifications consist of inserts, updates, deletes, and database schema changes. When you create a database using the

database initialization utility (dbinit), a log file is created with a default file name of *database-name.log*. When you use Sybase Central to create a database log file, the default displayed is the full path to the log file such as *drive:\path\database-name.log*. During recovery the database server must find the log file at this location. When the transaction log file is not specifically identified, the database server assumes that the log file is in the same directory as the database file.

A **mirror log** is an optional file and has a file extension of *.mlg*. It is a copy of a transaction log and provides additional protection against the loss of data in the event the transaction log becomes unusable. The mirror log is never backed up, since it is a duplicate copy of the transaction log.

CLIENT-SIDE AND SERVER-SIDE BACKUP

The database backup can either be performed as a **client-side** backup, or a **server-side** backup. The client-side backup is done using an external program (dbbackup in the example to follow) that performs the backup by making a connection to the database. This allows the backup to be done from anywhere that can connect to the database. If you are using embedded SQL, client-side backups can be initiated by using the db_backup function.

A server-side backup is initiated by the SQL statement BACKUP DATABASE and the backup is performed inside the server. One restriction of this method is that it can only be backed up to a disk that is local to the server. However, a server-side backup is not limited by the network speed, so the backup often will have better performance.

ONLINE BACKUP, OFFLINE BACKUP, AND LIVE BACKUP

Database backups can be performed while the database is being actively accessed (**online**) or when the database is shut down (**offline**). An online database backup is performed by executing dbbackup from the command line, using the Sybase Central Backup Database wizard, or executing the BACKUP DATABASE SQL statement. When an online backup process begins, the database server writes all cached data pages to the database file(s) on disk. This process is called a **checkpoint**. The database server continues recording activity in the transaction log file while the database is being backed up. The log file is backed up after the backup utility finishes backing up the database. The log file contains all of the transactions recorded since the last database backup. For this reason, the log file from an online full backup must be applied to the database during recovery. The log file from an offline backup does not have to participate in recovery, but it can be used in recovery if a prior database backup is used.

A **live backup** is carried out by using the dbbackup utility with the -l option. A live backup provides a redundant copy of the transaction log for restarting your system on a secondary computer should the main database server computer become unusable.

FULL AND INCREMENTAL DATABASE BACKUP

A database backup is either a **full** or **incremental** backup. For a **full** backup, the backup includes the database and log. An incremental backup includes only the transaction log. When you perform an **incremental** backup, the mirror log is not backed up. When you back up and rename the log files, the transaction and mirror log file are renamed and new log files are created. You must back up the mirror log manually. Be aware of this when planning your backup and recovery strategy.

DEVELOPING A BACKUP AND RECOVERY STRATEGY

UNDERSTANDING WHAT BACKUP AND RECOVERY MEANS TO YOUR BUSINESS

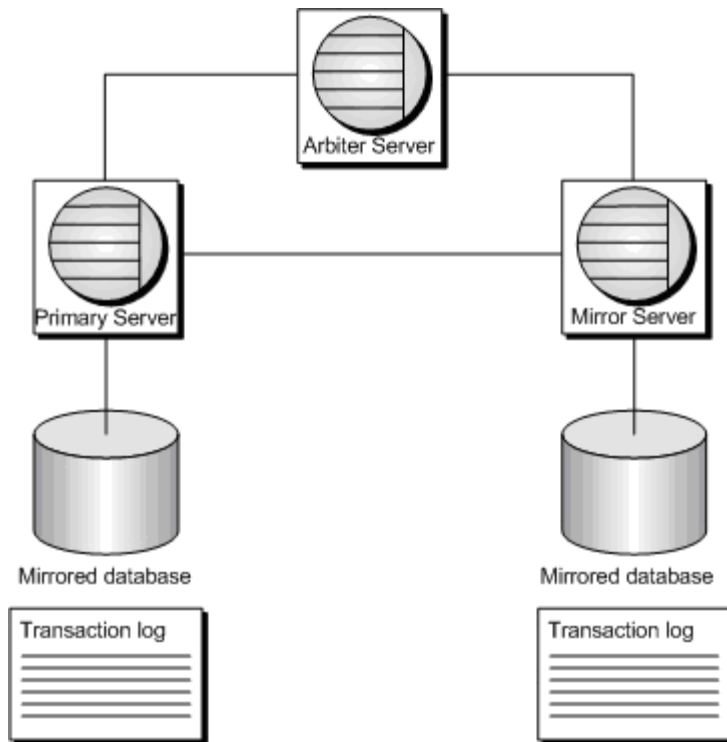
How long can your business survive without access to the corporate data? Express your answer in terms of minutes, hours, or days.

If your recovery time is in minutes, then database backup and recovery are critical to your business needs and it is paramount that you implement some kind of backup and recovery strategy. If recovery can take hours, then you have more time to perform the tasks. If recovery can be expressed in terms of days, then the urgency to recover the database still exists, but time appears to be less of a factor.

DECIDE IF HIGH AVAILABILITY IS RIGHT FOR YOUR BUSINESS

If you answered the above question in terms of seconds or minutes, another option that you should consider is high availability. SQL Anywhere 10 introduced an option for high availability using database mirroring to allow an automatic hot fail-over to a second database server. This allows continuous access to corporate data in the event that the primary server becomes unavailable.

High availability works by maintaining two separate copies of the database (a primary server and a mirror server) in a consistent state following each transaction. A third server acts as an arbiter, coordinating which of the two databases acts as the primary server. In the event that the primary server becomes unavailable, the mirror server assumes the role of primary server. This allows data access to continue normally in the event of failure.



High availability is extremely useful for companies that cannot be without access to corporate data for more than a few seconds or minutes. For more information about high availability, see the whitepaper [Running Multiple Databases with SQL Anywhere 10 Database Mirroring](#).

MANAGEMENT COMMITS TIME AND RESOURCES FOR THE PROJECT

Management must decide to commit financial resources toward the development and implementation of a backup and recovery strategy. The strategy can be basic or quite extensive, depending upon the business needs of the company. After developing a backup and recovery strategy, management should be informed of the expected backup and recovery times. Anticipate management countering the timings by preparing alternative solutions. These alternative solutions could be requesting additional hardware, improved backup medium, altering the backup schedule, or accepting a

longer recovery time versus backup time. Then it will be up to management to decide what solution fits their corporate needs.

DEVELOP, TEST, TIME, DOCUMENT, HEALTH CHECK, DEPLOY, AND MONITOR

These phases are essential for developing a backup and recovery strategy:

- Create backup and recovery commands. Verify that these commands work as designed. Does your full or incremental online backup work? Verify that your commands produce the desired results.
- Time estimates from executing backup and recovery commands to get a feel for how long these tasks will take. Use this information to identify what commands will be executed and when.
- Document the backup commands and create written procedures outlining where your backups are kept and identify the naming convention used, as well as the kind of backups performed. This information can be very important when an individual must check the backups or perform a database recovery when the database administrator (DBA) is not available.
- Incorporate health checks into the backup procedures. You should check the database to ensure that it is not corrupt. You can perform a database health check prior to backing up the database or on a copy of the database from your backup.
- Deployment of your backup and recovery strategy consists of setting up your backup procedures on the production server. Verify that the necessary hardware is in place, as well as any other supporting software that is necessary to perform these tasks. Modify the procedures to reflect the change in environment. Change the user ID, password, server, and database name to reflect the change in environment.
- Monitor backup procedures to avoid unexpected errors. Make sure any changes in the process are documented.

BEWARE OF EXTERNAL FACTORS THAT AFFECT RECOVERY

External factors that affect database recovery are time, hardware, and software. Allow additional recovery time for miscellaneous tasks that must be performed. These tasks could be as simple as entering recovery commands, or retrieving and loading tapes, or retrieving the backups from a SAN drive. Factors that influence the recovery time are the size of database files, recovery medium, disk space, and unexpected errors. If you use a dbspace with your database and it fails to be restored from a backup, then the backup is not valid. Adding more files into the recovery scenario increases the places where recovery can fail. Suppose you use four dbspaces, a main database file, and a transaction log file. If a dbspace fails to restore from your backup to disk, then you must revert to a prior full database backup. As the backup and recovery strategy develops, it may be necessary to check the performance of the equipment and software to ensure that it meets your expectations.

SQL Anywhere 10 introduced support for parallel backups. Parallel backups can be used to decrease the total time needed to perform the backup. A separate reader and writer thread are created for each physical device, allowing the reads and writes to be performed in parallel. This effectively increases the speed of the backup to that of the slowest component in the system.

PROTECT DATABASE BACKUPS BY PERFORMING HEALTH CHECKS

Database health checks are run against the database and log files to ensure that they are not corrupt. The database validation utility (dbvalid) is used to scan every record in every table and look up each record in each index on the table. If the database file is corrupt, you must recover from your previous database backup. A database can be validated before being backed up, or validation can be performed on a copy of the database from your backup.

Dbvalid can take time to check the database. You must decide when to run dbvalid. A database validity check done before running a full database backup would ensure that the database is valid

before being backed up. When you run `dbvalid` against a restored copy from a backup medium, you will be accomplishing the following checks:

- You will that verify the database can be successfully restored from the backup medium.
- You will be able to run a database validity check against the database without impacting your production environment.

Run `dbvalid` on a non-production computer to avoid impacting your production environment. The database used in the validation process cannot participate in recovery since log offsets will be changed once the database is started.

Problems can arise when trying to start the backed-up database if the associated `dbspaces` use fully qualified paths. When the backed-up database starts, it attempts to use the production `dbspace` files already in use by the production server. To remedy this, the database should be started with the `-ds dbspaces-directory` option to instruct the server to look only in the supplied directory for the `dbspaces`.

Log files are checked for corruption by using the `dbtran` utility. The `dbtran` utility translates a log file into a `.sql` file and verifies that the log file is free of corruption. This is important to keep in mind when you have to recover a large database and apply several log files. If, during a database recovery, a log file is found to be corrupt, then your recovery stops and you must determine if you can manually recover from that point to the time of failure. An alternative approach that can be used if the log file was previously translated, is to read the translated SQL file into the database, along with the remaining translated `.sql` files. Reading the `.sql` file(s) into the database in lieu of applying log file(s) is a recovery alternative, but this technique *cannot* be used when the database participates in replication or synchronization.

CREATE A SCHEDULED AUTOMATED BACKUP PLAN

One of the most common reasons that companies do not do due diligence on database backups is because they either forget, or simply do not have the time. SQL Anywhere 10 introduced the ability to create scheduled, automated maintenance plans using system events. The maintenance plan allows you to choose the specific operation that should be performed at each backup, such as validation and transaction log backup. The maintenance plan optionally allows configurable email alerts to notify you that the backup has taken place. A scheduled plan can be created using Sybase Central. The maintenance plan can only be used to implement server-side backups.

BACKUP AND RECOVERY STRATEGY EXAMPLE

It is extremely difficult to find a representative scenario for backup and recovery because the backup and recovery times depend heavily on the environment, especially the I/O subsystem. Furthermore, the requirements of the business dictate what the best backup procedure is. Therefore, the example situation described in this section does not use any specific times because they are likely to be very unrepresentative depending on the hardware you are using. The importance of testing on your own system and hardware many times to determine the best procedure cannot be overemphasized.

This example develops a backup and recovery strategy for a fictitious company called Data Online. Data Online requires access to corporate data stored in this database during business hours. Business hours are defined as a time period from Monday through Friday between the hours of 8 A.M. to 8 P.M. During business hours, Data Online cannot be without this information for more than `MAX_DOWNTIME`. Access to the corporate data outside of business hours is not critical. You can now try to develop a baseline for the time it will take to back up and recover the database.

The name of the database backup utility is `dbbackup`. It can be used to back up database and log files. A full database backup using `dbbackup` makes a copy of the main database file, database spaces, and transaction log file.

TIME THE BASIC BACKUP AND RECOVERY OPERATIONS

After running several full database backup tests, you calculated the average backup time to be FULL_BACKUP_TIME. Incremental backups should also rename and restart the log files to "prune" their growth. You now run several test incremental backups where you only back up the changes made to the database by backing up the transaction and mirror log file. After performing several backup tests you calculate the average incremental backup time to be INCREMENTAL_BACKUP_TIME. You decide to review a backup strategy that performs a full database backup nightly, and incremental database backups at 12:00 P.M. and 6:00 P.M. You now need to check if this backup strategy meets your recovery requirements of full recovery within MAX_DOWNTIME.

Data Online has a third party software tool to back up disk files to an offsite SAN drive. All database backups are copied onto this SAN drive. You calculate the average database and log backup and restore time from the SAN drive to be FULL_TRANSFER_TIME and INCREMENTAL_TRANSFER_TIME for full and incremental backups respectively. Now you can use these timings to develop a recovery strategy.

ESTIMATE A TOTAL RECOVERY TIME FROM FAILURE

The times estimated in the previous section give you a way to estimate what the recovery time will be given the following conditions:

- If you need to recover the database from a backup, it will take FULL_TRANSFER_TIME to restore the database from the SAN drive to disk.
- Each day's two incremental database backups additionally add their own INCREMENTAL_TRANSFER_TIME.
- You assume that the current database transaction log file is not corrupt and can be applied to the database. The recovery process of applying a log to the restored database takes approximately APPLY_LOG_TIME for each log file.
- You allow ISSUE_COMMAND_TIME for the actual issuing of the commands to perform the recovery.
- The calculation of the overall ESTIMATED_RECOVERY_TIME is shown below.

Recovery from backup medium phase

Operation	Time
Full database backup	FULL_TRANSFER_TIME
Incremental database backup	2 x INCREMENTAL_TRANSFER_TIME

Database recovery phase (applying incremental database backups to recovered database)

Operation	Time
Apply three log files	3 x APPLY_LOG_TIME
Issuing commands	ISSUE_COMMAND_TIME

Total recovery time

$$\begin{aligned} \text{ESTIMATED_RECOVERY_TIME} = & \text{FULL_TRANSFER_TIME} + \\ & (2 \times \text{INCREMENTAL_TRANSFER_TIME}) + \\ & (3 \times \text{APPLY_LOG_TIME}) + \\ & \text{ISSUE_COMMAND_TIME} \end{aligned}$$

This recovery example indicates that it takes approximately ESTIMATED_RECOVERY_TIME to recover the database and apply three incremental backups. A full database backup from the night before, along with the next day's two incremental backups will recover the database to the end of the next business day. If ESTIMATED_RECOVERY_TIME is less than MAX_DOWNTIME, then this backup strategy meets the business objective of database recovery within MAX_DOWNTIME.

You now have a backup and recovery strategy to present to management.

If ESTIMATED_RECOVERY_TIME turned out to be greater than MAX_DOWNTIME, then this strategy would not be acceptable for the business objectives, and a new strategy would need to be created and tested in a similar way.

WHAT DO YOU DO WHEN YOU ENCOUNTER ERRORS DURING RECOVERY?

During the restore process of the full database backup, a dbspace fails to restore from the SAN drive. This backup copy of the database is now invalid. Identify the next available full database backup and the incremental backups needed to recover the database to the original point of failure. You have the full database backup of the previous night to begin the recovery process. Use the previous day's full database backup to begin the recovery process. Recovery time would add an additional $FULL_TRANSFER_TIME + (2 \times INCREMENTAL_TRANSFER_TIME)$ to restore the previous day's files. It would also add an additional time to apply six log files (four incremental database backups and two transaction log files accompanying the previous day's full database backup and the failed full database backup).

RECOVERY FROM THE PREVIOUS DAY'S BACKUP

Recovery from medium phase (the first attempt fails to restore a full database backup from the SAN drive successfully)

Operation	Time
Full database backup	FULL_TRANSFER_TIME
Incremental database backup	2 x INCREMENTAL_TRANSFER_TIME

Second attempt at successfully restoring the previous day's full database backup from the SAN drive

Operation	Time
Full database backup	FULL_TRANSFER_TIME
Incremental database backup	2 x INCREMENTAL_TRANSFER_TIME

Database recovery phase (applying incremental database backups to recovered database)

Operation	Time
Apply six logs to the recovered database	6 x APPLY_LOG_TIME
Issuing commands	ISSUE_COMMAND_TIME

Total recovery time

$ESTIMATED_RECOVERY_TIME = FULL_TRANSFER_TIME +$

(2 x INCREMENTAL_TRANSFER_TIME) +
FULL_TRANSFER_TIME +
(2 x INCREMENTAL_TRANSFER_TIME) +
(6 x APPLY_LOG_TIME) +
ISSUE_COMMAND_TIME

This second scenario illustrates the need to be ready for the unexpected. Every database recovery will be different from the previous one. If one of the database files from a full database backup is corrupt, verify whether the log files are valid. If so, then at least you can recover from a previous day's full database backup and recover past the point of failure by applying the log.

ADDITIONAL INFORMATION ABOUT RECOVERY

Before doing any kind of database recovery, you should back up your failing database. If you feel uncomfortable or unsure of what to do, then you should contact Sybase Technical Support. Even if you have already come up with a way to recover the database, it is worth getting a second opinion and to have your recovery plans verified by Sybase Technical Support.

It should be stressed that multiple full database backups allow for multiple failsafe points of recovery. Run dbtran against each incremental backup to verify that the log is valid. Do not replace the log file with the translated *.sql* file if your system participates in replication. This alternative approach will break your replication. If you do use a translated *.sql* file, then all remaining log files must be translated and read into the database. This alternative recovery will take longer than applying the log files.

SECONDARY BACKUP ISSUES

- Define a backup retention timeframe for your backups.
- If your backup medium is tape, then decide the number of times they will be used before discarding them.
- Test your backup procedures by restoring from them to ensure that they can be restored.
- Make up different recovery scenarios and try to recover the database from them.
- Keep your recovery procedures current.

BACKUP AND RECOVERY COMMANDS

The database server used in this section is SQL Anywhere 10. The path in the following examples is not indicated since you will either use the installation default or change the path accordingly. The other utilities are the same between database versions unless specified.

TEST YOUR COMMANDS

It is not intended for these commands to be copied as is into a backup and recovery script. Always test your commands before deploying them.

DIRECTORIES USED

Some file directories that are used in this example:

- *C:\prod* (production database directory)
- *C:\bkdb* (failing database directory)
- *D:\rcv* (recovery database directory)
- *D:\dbackup* (full database backup)
- *D:\lgbk* (incremental log backup)
 - *h0800* (backup at 8:00 A.M.)

- 1h1200 (backup at 12:00 P.M.)
- 1h1700 (backup at 5:00 P.M.)

In this example, the database is run on a personal server.

```
dbeng10 -n server C:\prod\database.db
```

CREATING BACKUPS

Create a full online backup of the database and log file.

```
dbbackup -c "UID=DBA;PWD=sql;ENG=server;DBN=database" -y D:\dbbkup
```

An incremental online backup of the transaction log file and the log file is truncated at 8:00 A.M.

```
dbbackup -t -x -c "UID=DBA;PWD=sql;ENG=server;dbn=database" -y  
D:\lgbk\h0800
```

An incremental online backup of the transaction log file and the log file is truncated at 12:00 P.M.

```
dbbackup -t -x -c "UID=DBA;PWD=sql;ENG=server;DBN=database" -y  
D:\lgbk\h1200
```

An incremental online backup of the transaction log file and the log file is truncated at 5:00 P.M.

```
dbbackup -t -x -c "UID=DBA;PWD=sql;ENG=server;DBN=database" -y  
D:\lgbk\h1700
```

If you are using SQL Anywhere 10, the last successful backup is stored in SYS.ISYSHISTORY. The details of the last backup can be viewed by executing the following SQL command from Interactive SQL.

```
SELECT * FROM SYS.SYSHISTORY WHERE operation = 'LAST_BACKUP';
```

RECOVERING BACKUPS

To recover the database from the full backup and apply all of the incremental backups (transaction logs) and log file at the time of failure to recover the database to a current point in time:

1. Create a backup copy of the failing database and log to *c:\bkdb*.
2. Create a recovery directory *d:\rcv*.
3. Copy only the database from the full backup to *d:\rcv*.

Start a full database backup by applying the log file using the *-a* (apply log) option. Monitor the Server Messages window for error messages. You should see a message indicating that recovery is in progress.

```
dbeng10 D:\rcv\database.db -a D:\dbbkup\database.log
```

Apply the remaining log files:

```
dbeng10 D:\rcv\database.db -a D:\lgbk\h0800\database.log
```

```
dbeng10 D:\rcv\database.db -a D:\lgbk\h1200\database.log
```

```
dbeng10 D:\rcv\database.db -a D:\lgbk\h1700\database.log
```

Apply the log file at the time of the database failure:

```
DBENG10 D:\rcv\database.db -a D:\bkdb\database.log
```

Copy only the database file to the production directory. A new log file will be created. Now you are ready to perform health checks against the database.

If you are using SQL Anywhere 10, multiple log files can be applied to the database using a single command. You can use the *-ad log-directory* option to instruct the database to apply all log files it can find in the supplied directory. Alternatively, the *-ar* option instructs the database to apply any log files that reside in the same directory as the current transaction log file. Since the database is

not stopped between applying log files, subsequent log files are applied to the “warm” database. This reduces the overall time needed to perform the recovery.

PERFORM HEALTH CHECKS

Now start the database using the personal database server (dbeng10). This ensures that other users cannot connect to the database until you are certain it is valid.

The database is loaded on a personal server.

```
dbeng10 -n server C:\prod\database.db
```

Verify that the database is not corrupt by running DBVALID.

```
dbvalid -c "UID=DBA;PWD=sql;ENG=server;DBN=database"
```

CREATE A POST-RECOVERY BACKUP

Delete old copies of the database in the backup directory. You may want to back up the files to a backup medium.

Create a post-recovery backup of the database and log file.

```
DBBACKUP -c "UID=DBA;PWD=sql;ENG=server;DBN=database" -y D:\dbbkup
```

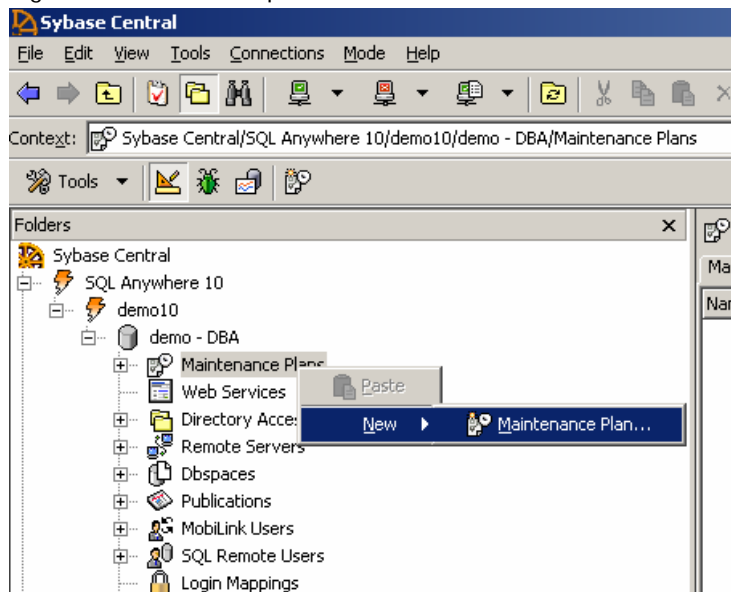
ALLOW USER TO ACCESS THE DATABASE

Shut down the database because you used the standalone (personal) database server to perform health checks, verify that the database was valid, and perform post-recovery of the database. Start the database server in the normal manner prior to the recovery. Monitor the database server and user activity against the database for unexpected error messages.

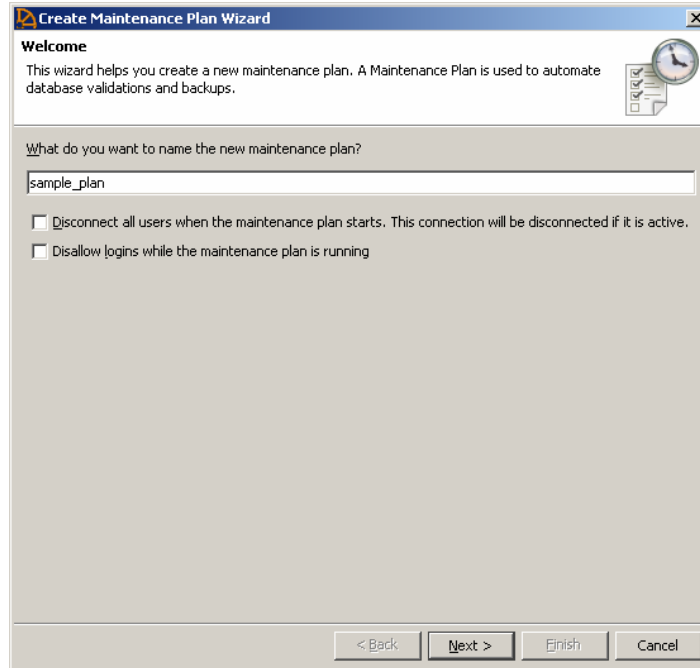
CREATING A SCHEDULED AUTOMATED BACKUP PLAN

If you are using SQL Anywhere 10, most of the backup operations can be performed automatically on a schedule. This is accomplished by using events to trigger the backup process. In this example, you create a simple automated backup plan using Sybase Central.

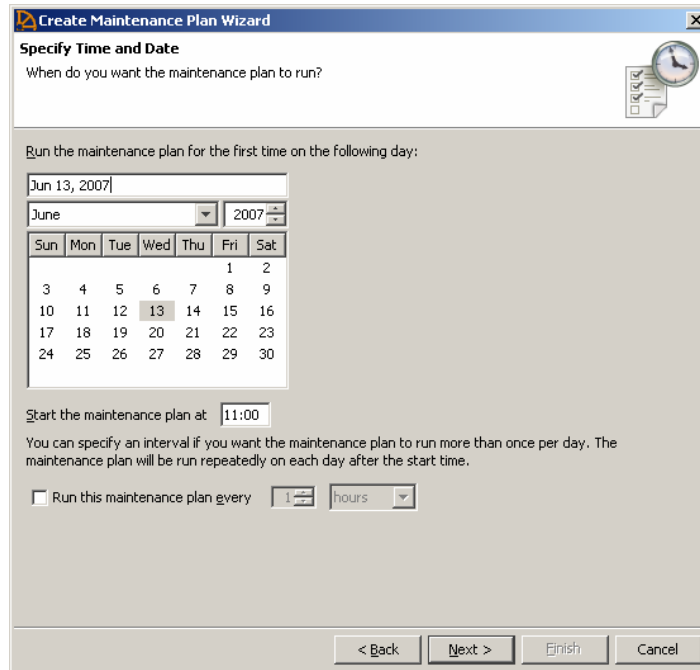
1. Start your database.
2. Open Sybase Central and connect to the database using the SQL Anywhere 10 plug-in.
3. Right-click Maintenance plan. Choose New > Maintenance Plan.



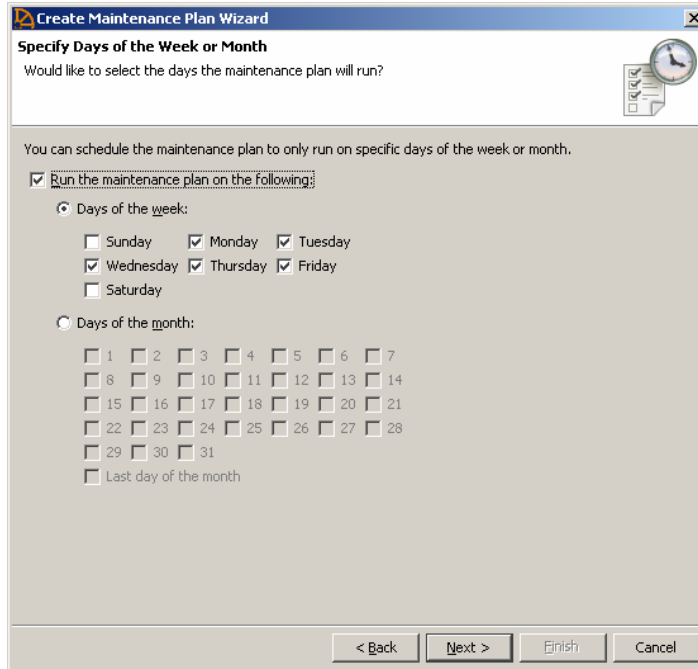
- Name the maintenance plan.



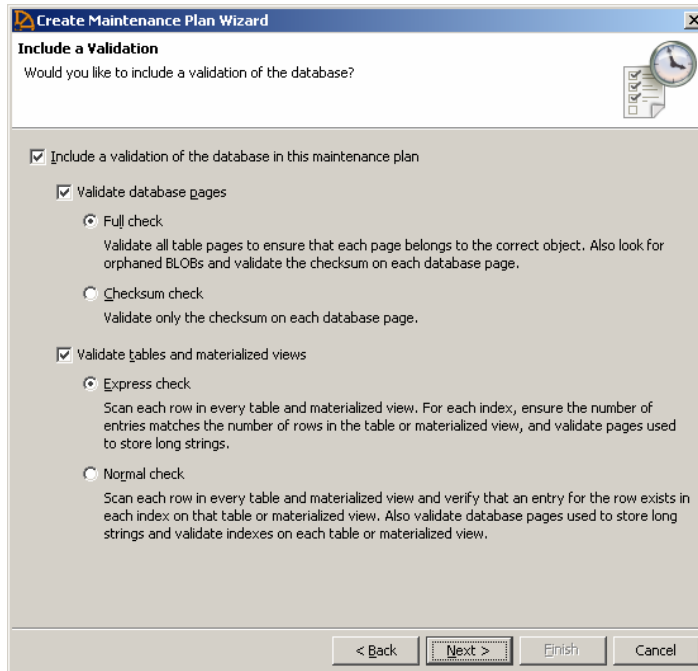
- Choose the first day and time that the plan is to run on.
- Optionally, specify how often the plan is run on a scheduled day.



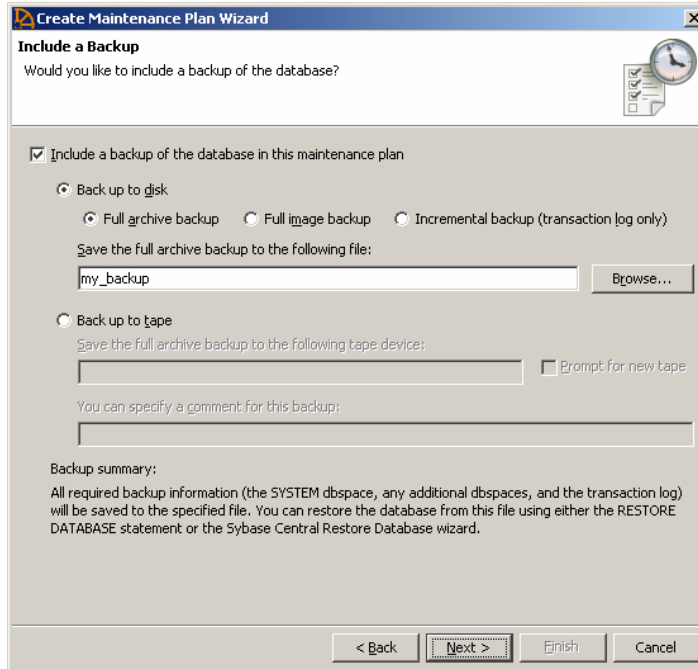
- Choose either the days of the week or the dates of the month when the plan should be run.



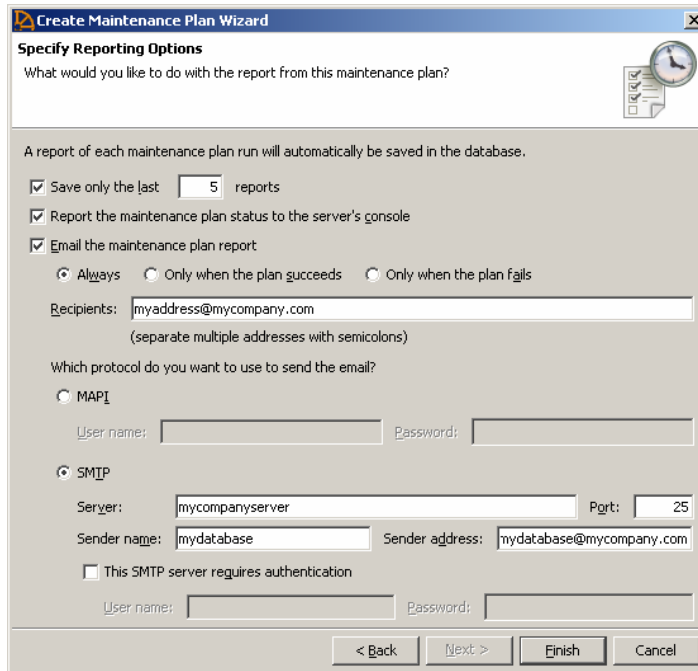
8. Optionally, choose any validation operations that you want to perform.



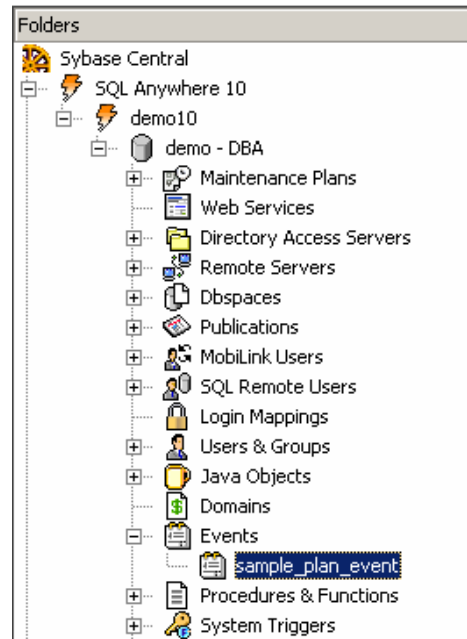
9. Choose the type of backup (full or incremental) that this plan will perform.
10. Set the target location of the backup.



11. Optionally, choose to have backup notification reports emailed to you



You can create multiple maintenance plans and have each plan perform a different task. The actual backup is initiated using system events. You can view the actual SQL execution code for the backup by selecting Events > *maintenance-plan-name_event* in Sybase Central.



DISASTER RECOVERY

Disaster recovery is different from database recovery because it usually implies that the computer is no longer available. This may be the result of a flood, natural disaster, or an inoperable computer. This type of situation requires operating system, system software, database software, and application software recovery onto a physically different computer. The computers may be similar or identical. If the computer is different, this could impact your disaster recovery. After the system and database software are installed, then database recovery procedures can be started.

ADDITIONAL INFORMATION ABOUT DISASTER RECOVERY PLANNING

The following list is an example of what information is needed when developing a disaster recovery strategy. You should review your strategy and update it accordingly if you have omitted any of these points of interest. This list is only representative of what may be required and can vary based upon each site's configuration of hardware, software, location, and personnel.

- Establish an offsite location with a compatible system and network hardware.
- Establish procedures for retrieving offsite backups.
- Disaster recovery procedures should be kept in multiple places: at the offsite location with the backups for the operating system, system software, database, and application software, as well as with individuals involved with disaster recovery.
- Schedule regular disaster recovery tests and after each test review the outcome.
- Verify that the disaster recovery procedures contain copies of all installation software and their passwords for the operating system, system software, database, and application software.
- Attach a current list of all software vendor names, phone numbers, product support plan numbers, and software license information to the disaster recovery procedures.
- Record the total time required to complete your disaster recovery.
- Keep the disaster recovery procedures current and update the copies kept offsite.

SECURITY CONCERNS

It is important that you take similar security precautions with the database backups to those you use with your running production database. You should make sure that they are stored in a secure way with limited access. The full database backup files are complete versions of the database (this is necessary to use them for full recovery). This means that anyone can take the backup file and start it on a server elsewhere and attempt to access the data or break passwords.

Furthermore, if the database is not encrypted, anyone who has access to any of the backed up transaction log files can use the dbtran utility to view the queries stored inside that transaction log. For these reasons, it is very important to keep your backups in a secure location to prevent them from being used maliciously.

Also, be careful of possible security concerns when using a client-side backup utility such as dbbackup. The utility connects to the database using a connection string. In most cases, it will make sense to run the utility by logging in as DBA. This has the potential to expose the DBA password if you are using a script that contains the hard coded connection string. Use secure files or some other authentication method to make sure that someone cannot acquire the password by simply reading the file. Server-side backups do not have this problem because they are initiated inside the server.

SUMMARY

The material presented in this document provides background information for your development and implementation of a database backup and recovery strategy. This document identified the files used in a database and the differences between online, offline, and live backups. It stressed the importance of incorporating health checks into the backup and recovery strategy to safeguard your investment. A backup and recovery example was given showing factors involved in the development of a backup and recovery strategy. It discusses what is involved with disaster recovery. Disaster recovery should not be overlooked. An accident can happen to your equipment, causing your server and database to become inoperable. It is critical that you test your backup and recovery procedures. Assume nothing, and verify that everything works as expected. This discussion has provided you with valuable information that is necessary for developing and implementing a backup and recovery and disaster recovery strategy.

Additional information about backup and recovery can be found in the book *SQL Anywhere Server - Database Administration* in the "Backup and Data Recovery" chapter.

