4D Database Recovery Using Backups and Mirroring – Part II

By Tai Bui, Technical Services Engineer, 4D Inc.

Technical Note 18-14

Table of Contents

Introduction

A 4D database can be an integral part of any system or business operations as it can simply be a tool to store data and go above and beyond to work with the data and provide services for users. However, as Murphy's Law states, "Anything that can go wrong will go wrong." As such it is always important to have a contingency plan and know what to do in cases of emergencies and accidents. There are a number of issues that can happen and there are a number of ways to handle them to reduce the amount of time the 4D database is down.

This Technical Note is the second of a two-part series. The first part provided a general informative overview over 4D's features that can be utilized to cut downtime in case a database is damaged. These features are the backup system and how to set up and use a mirror. This one will go in-depth about the mirroring process and provide more details suggestions and examples of how to recover from a mirror.

Consideration for a Mirror

As explained in the first part of this two-part Technical Note, the 4D database can be a core system of a business. It can be important to make sure that the database is up and running at all expected times. However, uncontrollable events can occur that can cause the database to become corrupt. To mitigate the negative events of such an event, it is beneficial to have a system and plan prepared. The first part of the Technical Note provided a number of general processes to perform and implement in case of the database becomes corrupt.

Many of the recovery processes will generate some downtime depending on the size of the database. Having a mirror can be one of the plans that provide recovery with as little downtime as possible when done correctly. As such mirroring can be considered when very high uptime and stability of a database is needed. The use of a mirror will also add another level of data protection due to having a second copy of the data and will also reduce the reliance on performing and maintaining backups which can cause a pause in the database workflow and generate disk usage. However, if a mirror is considered it is highly recommended that two separate physical machines are used as well as the machine having some way to communicate with each other. The Technical Note will go into detail over a Mirror's functionality and how it provides these benefits.

Mirror Functionality

Setting up and using a mirror will require some planning. This section will describe how a mirror is typically set up and how it should function with a typical setup.

These details can be applied to any type of database but will be described in the context of 4D.

Separate Duplicate Application

Mirroring is performed by maintaining two copies of the same database running simultaneously. It is recommended that these databases be hosted on two different physical machines; this will prevent hardware malfunction form affecting both databases. When both databases are started one of them acts as the principal database, being the primary database that is interacted with and the other acts as the mirror database or secondary database that acts as a standby database maintaining the same data.

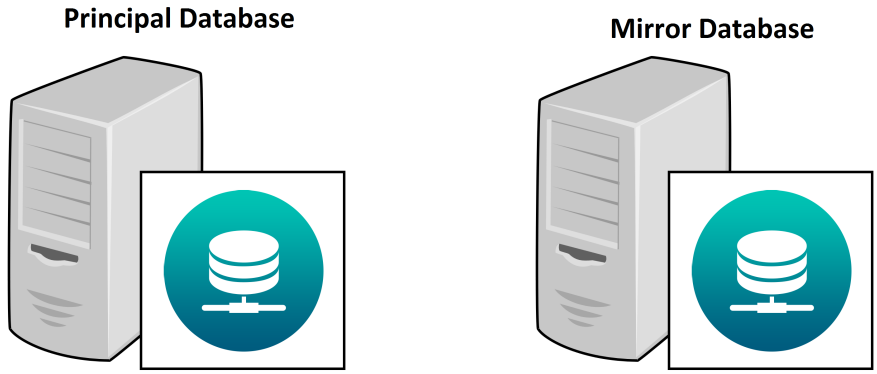

Image 1: Two 4D Servers Running On Two Separate Machines

Usually, two copies of the 4D Server are deployed to two separate machines and started up. The databases should have some code to allow the database to identify itself as either the principal server or as the mirror server. This can be accomplished in numerous ways. Some simple methods are to have the startup code search for the existence of a file or folder and if so it is the mirror.

Same Data from Reproduced Data Operations

The major feature of running a mirror is that there exist two copies of the database that maintains the same current set of data. When the data is touched in any way on the principal database the same actions will get pushed to the mirror database to reproduce the same actions resulting in the same data. In mirroring there are two methods of pushing the data hot standby and warm standby. Hot standby is when every action performed on the principal database is immediately pushed and reproduced on the mirror server in a fully synchronized way that allows full accuracy at all times. Warm standby is when the mirror is not fully synchronized to the principal database and at times the mirror may not contain all of the most recent data operations on the primary database.

In 4D, a Mirror is usually set up as a warm standby database. Because of 4D's features, there are no major detriments to setting the mirror up as a warm standby database.

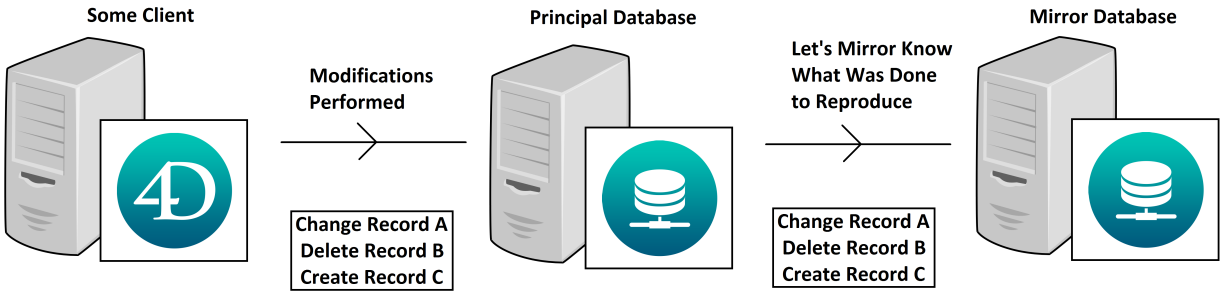

Image 2: Diagram of generic work flow of mirroring process

Functionally the data is pushed as follows:

1. On the principal database, data modifications are performed and are logged in the current log file as shown in the image below.

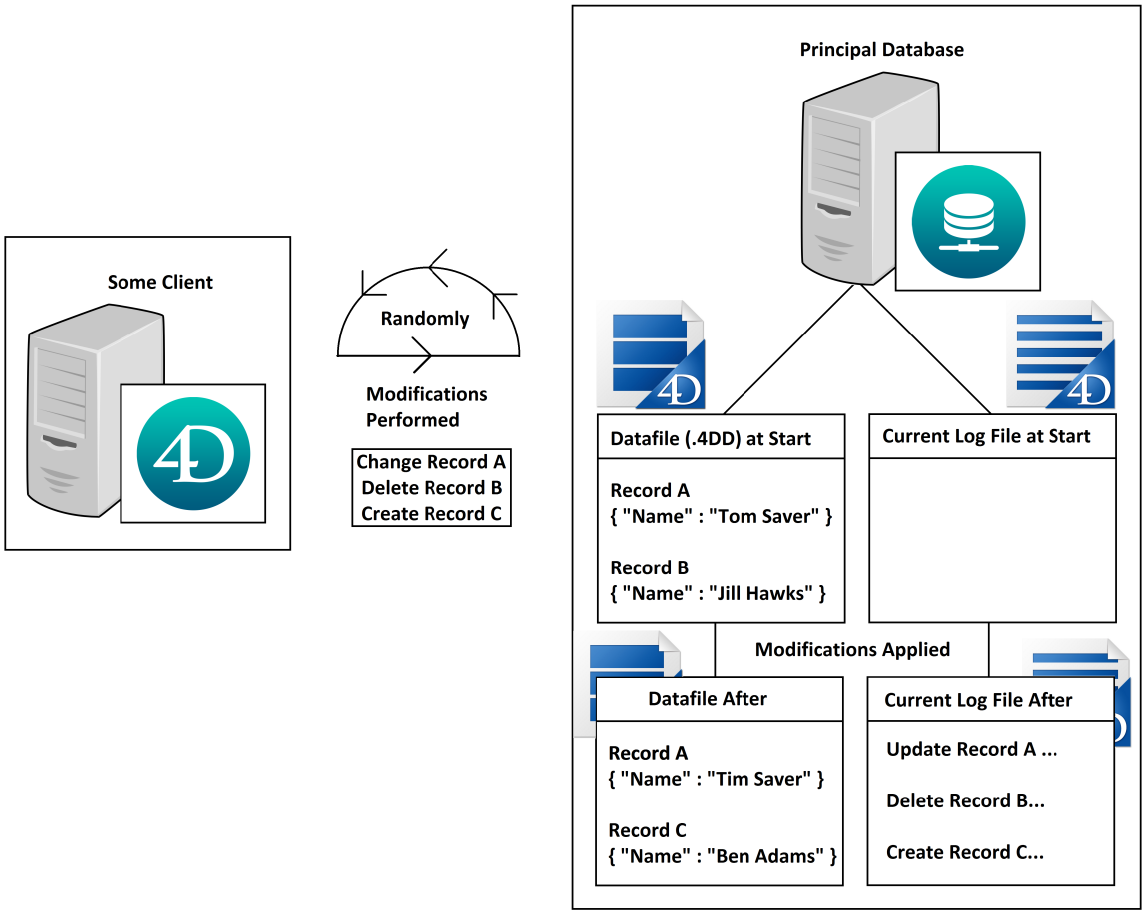

Image 3: Diagram of how logs and datafile are related with operations

2. After a set period of time, the current log file is closed and a new one is started using the **New log file** command. This is typically done in a regular interval in a stored procedure. The closed log file(s) is somehow transferred to the mirror database. This can be done in numerous ways from either the principal server sending it to a shared location or the mirror retrieving it through a file transfer.

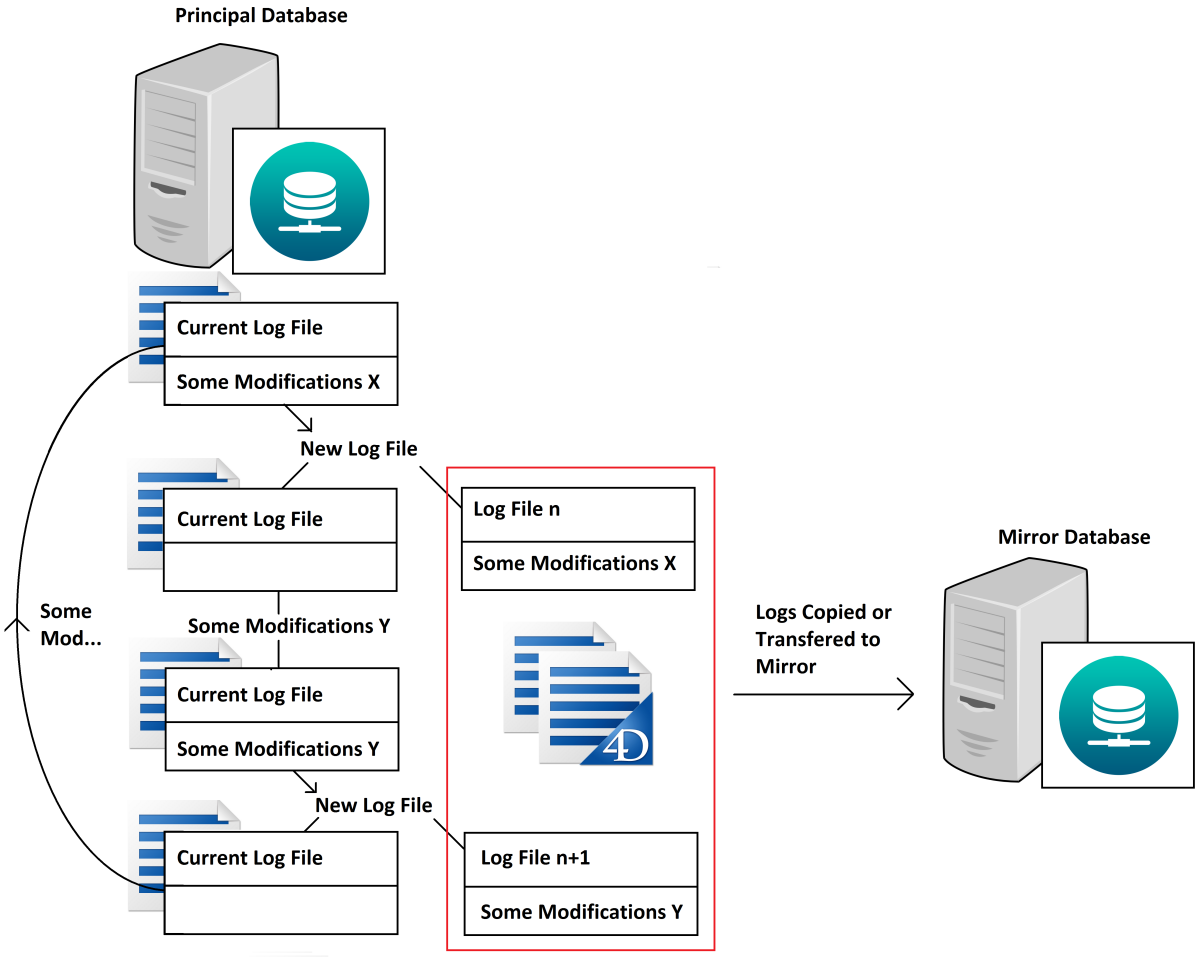

Image 4: Diagram of how logs are generated and sent to Mirror

3. The mirror server then regularly checks for new log files received and reproduces the actions stored in them using the **Integrate Mirror Log File** command.

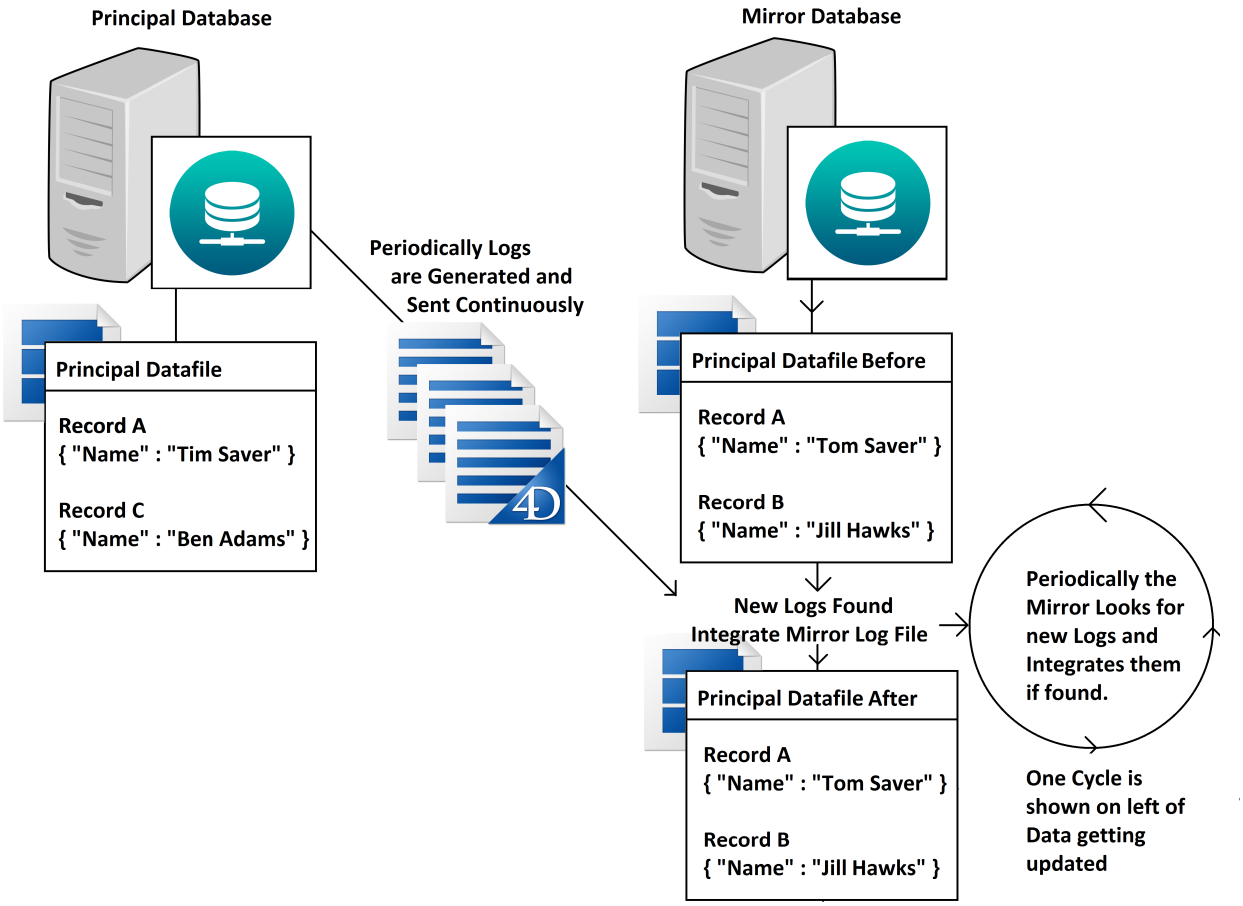

Image 5: Diagram of how the warm standby database maintains an accurate copy of datafile

As laid out, the process is fairly simple and allows the mirror to be mostly accurate based on the interval times applied to the processes for generating the log files, sending the log files, and integrating the log files. The mirror database, as a result of the process, this generates and maintains a live and most up to date copy of the data.

Recovery from a Mirror

As explained in Part I of this two-part Technical Note series, events can randomly occur that can damage a database. For example, a simple scenario would be the surge causing the machine hosting a database to not gracefully shut down. Then the database is started back up the machine and then the database shows that the database is damaged and will not run. In this scenario, if a mirror was set up for the database the recovery process can be very short, assuming that the mirror was not impacted by any issues too.

Use Datafile from Mirror

The first step is to forgo the datafile from the principal database and instead shut down the mirror database and work with its datafile. Since the datafile from the mirror is damaged, it can no longer be trusted when compared to the fairly accurate datafile maintained by the mirror.

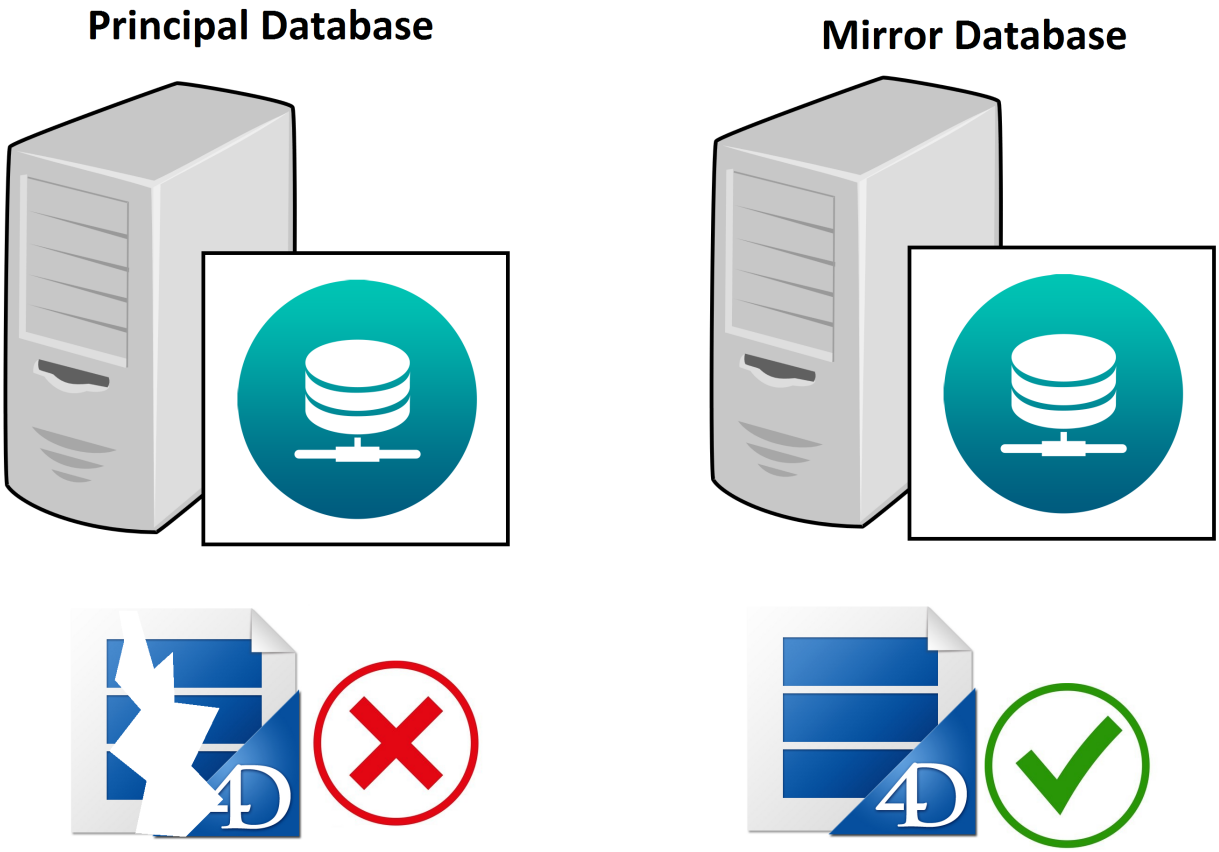

Image 6: With a mirror a fairly accurate copy of the datafile is available at all times even when principal datafile is damaged.

Integrate Current Log File

The next step is to use the current log file from the primary database with the datafile from the mirror and integrate the log. Because the mirror was set up as a warm standby database it may not be 100 percent accurate and contain every single operation performed. This can be remedied due to 4D's logging system. With 4D's logging system a live current log file is maintained that logs every operation performed on the datafile. With the datafile from the mirror and the current log file from the primary database, the operations from the current log file can then be integrated into the datafile on either database. Based on the interval set for closing and generating the log files to push to the mirror, the number of operations should be fairly small meaning the integration should not take long.

Handling the datafile and log must be done carefully, done wrong and it can cause the process to take longer than it should. The log and datafile can be integrated in two ways and depending on the method used, it can be done at any location.

Manually Integrating the Log

One method is to continue to use the Integrate Mirror Log file command to integrate the log file from the principal database to the mirror's datafile. This is an efficient method if performed on the mirror database. After an event, has occurred and the mirror is still running.

After starting back up the principal machine the log file is transferred to the mirror. A method will have needed to be implemented to integrate the log if found or the log will need to be renamed to the next log in sequence after the last one that was created from the New log file command. Then the operations in the log will be integrated into the datafile and updated. The next part is important. The log file and the new datafile are a pair that must be maintained together when transferring the files to the primary database. The log file should be renamed back to its original of "{databaseName}.journal". If the datafile is placed in the primary database logging will not occur unless a backup is performed, which will cause more downtime and can be significant if the database is large.

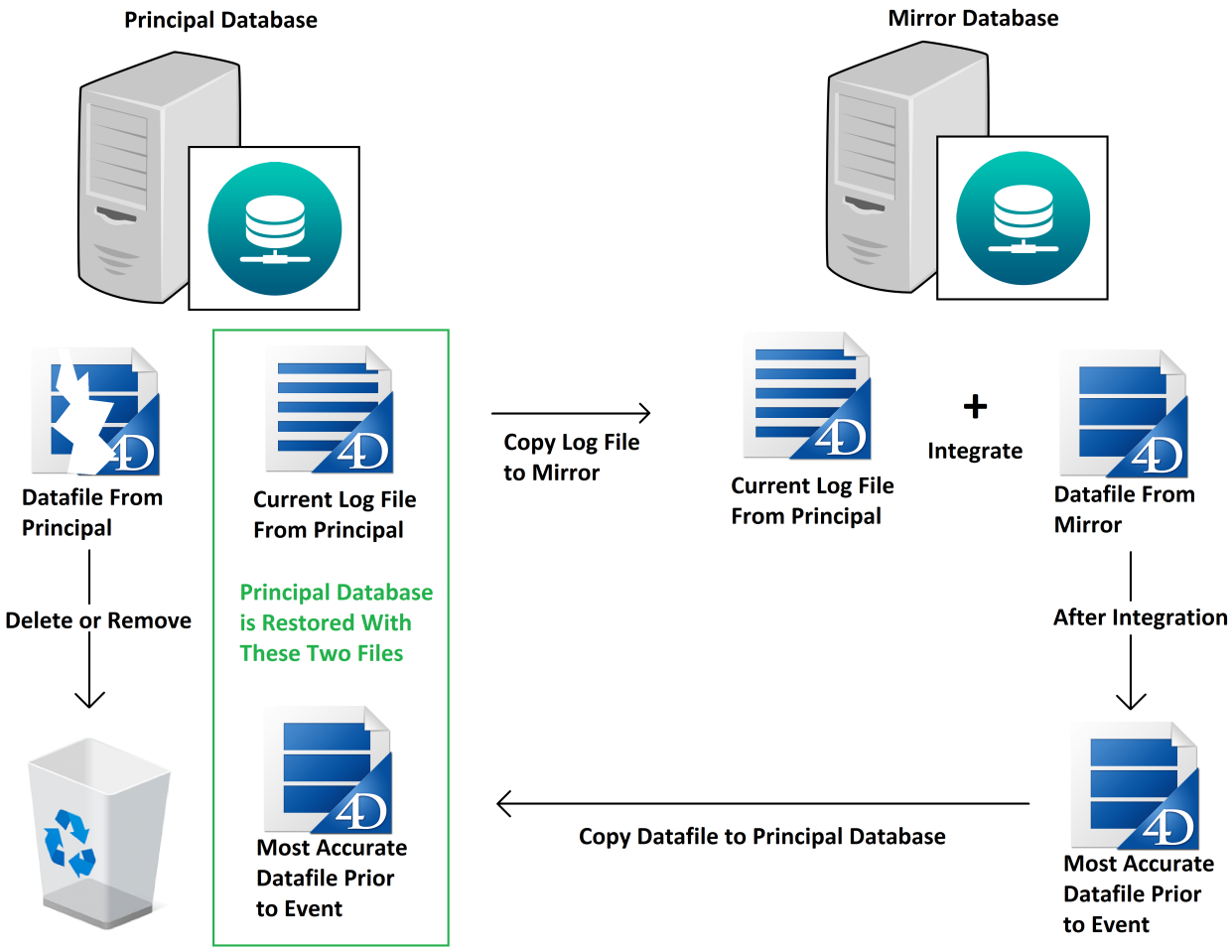

Image 7: Diagram of manual integration process

The datafile to run on the Mirror requires consideration and planning. Since the principal database's current log file's operations are already integrated into the new datafile, when the log gets closed and sent to the mirror an error can occur if strict mode integration is used. So some options to prevent this are as follows:

- Maintain and use a copy of the Mirror's datafile prior to the integration process

- Immediately close the log file on the Principal Database and do not integrate it into the mirror.

- The integration system must maintain the operation Number from the INTEGRATE MIRROR LOG FILE command and to start the integration of the next log from the operation Number

- The integration system must use the INTEGRATE MIRROR LOG FILE in auto repair mode.

Allowing 4D to Integrate the Log

The second method is to use 4D's *Automatic Restore>Integrate last log if database is incomplete function*. With this feature, if the datafile is synced with the log file, but is missing operations, these operations can be integrated automatically without needing to call any commands. This function is performed on startup. In this method, the process can be a bit easier than the first method.

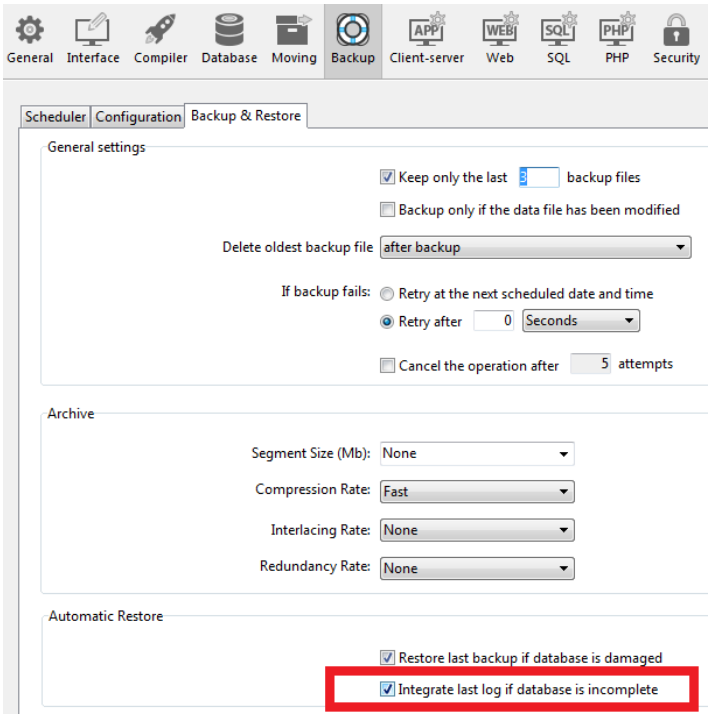

Image 8: Database setting to automatically integrate missing operations

After the event has occurred and the primary machine has been started up, shut down the mirror database and transfer the mirror's datafile to the principal database replacing the damaged datafile. Then just start up the principal database. If the *Integrate last log if database is incomplete function* is enabled (checked) then it should automatically integrate the log and the database will just seemingly start up with no issues seen and the database is good to go. If the setting is disabled, then the database will present a dialog informing that there are missing operations and allow the user a choice of allowing 4D to automatically integrate the operations or to create a new log file (in the case of restoring from a mirror, the user should integrate).

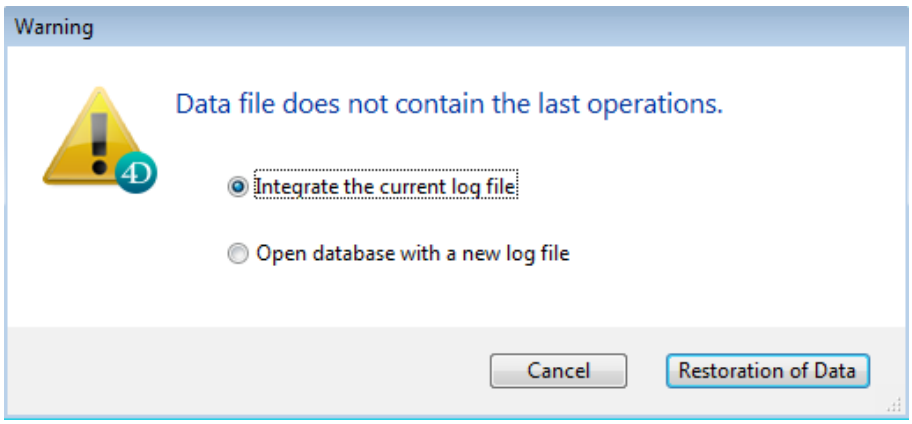

Image 9: Dialog if *Integrate last log if database is incomplete* is disabled

If the user does not integrate, it is not irreversible as long as the datafile is not touched, or a copy of the datafile from the mirror has been maintained. When the second option is selected the log is renamed and maintained before being replaced with a new one.

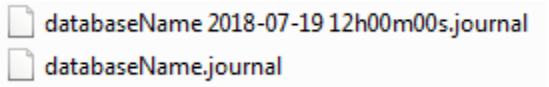

Image 10: Original log is renamed with timestamp and new log file added

With this method, the Mirror does not have to be touched in any way. After the principal server is started up from the process and business operations continue, the log will eventually get closed and sent to the mirror for integration.

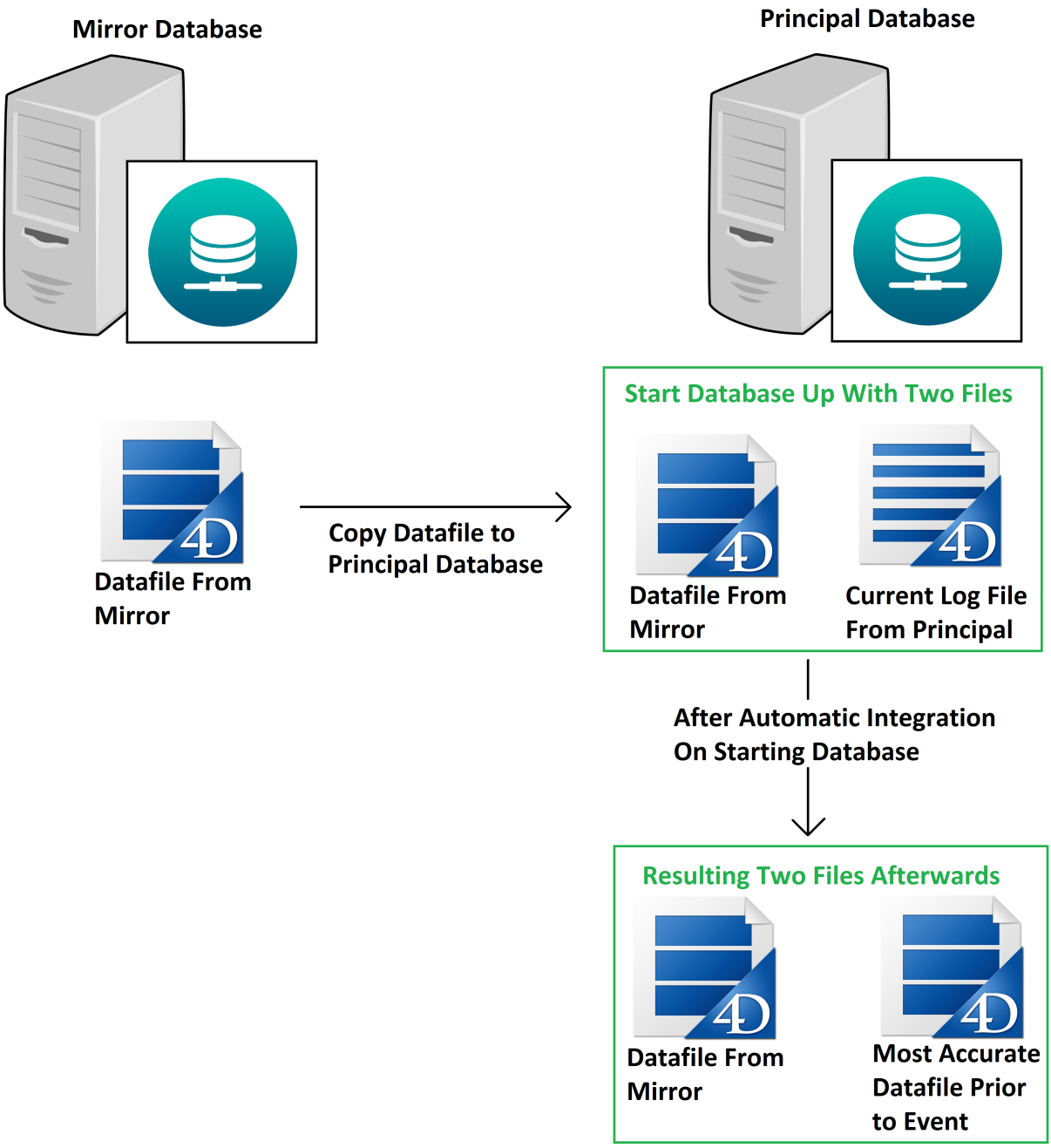

Image 11: Diagram of automatic integration process

It remedies the cons of both hot and warm standby databases that other database systems may encounter. With a hot standby database, the frequent updates can generate high latency that can hinder performance. With a warm standby, database the data may not be fully identical. The features of 4D allow a mirror to run as a warm standby database, which will not heavily impact

performance and allows for the datafile to be fully identical, achieved through integrating the current log file from the principal database.

Restart Production and Mirror Servers

With an undamaged up to date copy of the datafile and an undamaged copy of the database, the primary database can then be run properly and observed. If everything is good, the mirror database should then be started up back up.

The use of a mirror and its processes provides many benefits. The most beneficial impact of a mirror is the reduction in downtime in case of an unforeseen critical event. Compared to a repair or a full backup restore and integration of logs, obtaining the mirror's datafile and integrating the principal database's current log file is significantly faster. This also reduces the reliance on backups, meaning performing regular backups can be done in a less frequent manner and saves disk space maintaining the larger number of backups generated by a frequent process.

Conclusion

This Technical Note provided detailed information on mirroring in the scope of 4D and the recovery procedure. The mirroring process basically allows two copies of the same database running simultaneously where one acts as the primary database for database operations and the secondary "mirrors" (copies) the operations to maintain two copies of the same datafile. An in-depth look at how a mirror is set up with 4D was detailed in how mirroring with 4D is simplified due to the logging system. This allows for a fairly accurate copy of the datafile at all times. Recovering from a disaster, when using a mirror, is shown to be a simple process and quick process. The mirror's datafile and the principal database's current log file are integrated then the databases are restarted with the new datafile. Having a mirror is very beneficial in deployments in which high uptime is required and data protection is important while having a second machine is not that big of a detriment compared to the benefits.