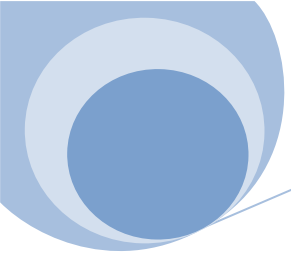




## MS SQL Server - An Overview



## MS SQL Server – An Overview

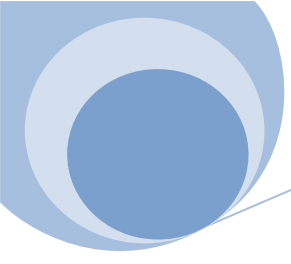
*Microsoft SQL Server's tight integration with Windows Server, automated self-tuning and management tools, and the wide availability of developers and compatible business applications can help small to medium-sized businesses achieve a positive ROI (Return-on-Investment).*

It's been said before but it bears repeating: Small and medium-sized businesses (SMBs) are the bedrock of American innovation and progress. Founded by entrepreneurs with dreams of running their own business, they are the American dream personified. As the marketplace begins to favor your company, a decision has to be made relative to how critical information is structured and accessed in order to manage every detail of the business.

Regardless of a business' size, in order to run a variety of applications — including enterprise resource planning (ERP), customer relationship management (CRM), supply chain management, product lifecycle management, and even basic e-commerce — it is necessary to use a relational database. The challenge for small and medium-sized companies is to find a database system that can be deployed with minimal cost and without the need to hire costly database administrators.

Many smaller companies utilize workstation-scale database products such as Microsoft Access, FileMaker Pro, or even Word or Excel for their database needs. While these solutions are acceptable for small-scale databases, growing companies generally consider a more capable, relational database system that can easily scale as business — and the amount of data it generates — expands. More sophisticated database systems from vendors such as Microsoft, Oracle and IBM offer a range of features; including back up and recovery; greater performance, scalability, and flexibility, as well as a higher level of security.

Some modern database management systems are becoming easier and less expensive to acquire, implement, and manage. Smaller businesses will find that if they choose the proper software, they can deploy relatively sophisticated relational databases using personnel with a minimum of technical knowledge or database experience. However, not all database systems are created equal. Certain database systems, such as those from Oracle and IBM, require dedicated database administrators (DBAs). Other systems, such as Microsoft SQL Server, feature a familiar point-and-click interface and a high level of automation that allow smaller companies to deploy relational databases without adding technical staff.



Some of the important advantages of a relational database for SMBs include:

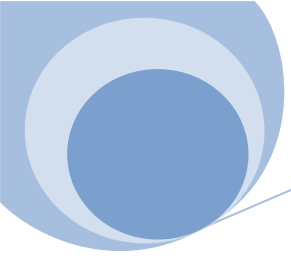
- **Scalability.** In the world of databases, scalability refers to both the maximum database size and the performance levels users can expect when adding, changing, or querying the database. Companies with heavy usage will want the performance levels and flexibility that a relational database management system (RDBMS) provides.
- **Data protection.** Protecting critical business data is the cornerstone of an advanced database system. Database software such as SQL Server features built-in automated backup and recovery tools to ensure that information is protected in case of a system failure.
- **Security.** Unlike a flat file system, an RDBMS allows companies to specify security policies at a granular level, right down to who can see which column of data and under what circumstances.
- **Data analysis and business intelligence (BI).** As companies grow, gaining knowledge from data becomes increasingly important. Only an RDBMS is powerful enough to allow sophisticated analysis services such as data warehousing and online analytical processing (OLAP).

Lower deployment costs also hold true for the design and development of applications that run on top of SQL Server. Many organizations find that the universe of developers who are well versed in programming applications for Windows servers and SQL Server is larger than that for competing databases from Oracle and IBM. Additionally, Microsoft's integrated development tools allow applications to be built without the need for dedicated database programmers. This can lead to reduced development costs and increased time to market compared with a competing database because Microsoft developers are plentiful, resulting in greater competition for projects and lower average fees.

Returns from lower deployment costs compared with competing databases include:

- Lower license costs
- Lower server software and hardware costs
- Lower training costs
- Lower IT personnel costs during deployment
- Lower developer hiring costs

Additionally, SQL Server's self-tuning and optimization routines require little, if any, IT staff intervention, making the day-to-day tasks of maintaining the database less costly than maintaining other database systems. Moreover, because SQL Server features an interface familiar to IT workers experienced with Windows systems, most companies will find that they can assign



workers with little specialized database experience to SQL Server management duties. And although it would be hyperbole to state that SQL Server “runs itself,” most companies will find that once it is up and running, a SQL Server database requires a minimum of day-to-day maintenance and intervention.

Ultimately, Microsoft SQL Server enables small to medium-sized business to realize a positive return on investment (ROI) because of its lower deployment costs, improved technology management and its ease of use, even for organizations without dedicated database developers and managers.

## What is Microsoft SQL Server?

Microsoft SQL Server is a relational database management system (RDBMS). Its primary query language is Transact-SQL, an implementation of the ANSI/ISO standard Structured Query Language (SQL) used by Microsoft and Sybase.

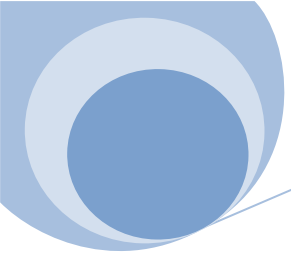
The architecture of Microsoft SQL Server is broadly divided into three components: *SQL OS* that implements the basic services required by SQL Server, including thread scheduling, memory management and I/O management; the *Relational Engine*, which implements the relational database components including support for databases, tables, queries and stored procedures as well as implementing the type system; and the *Protocol Layer* that exposes the SQL Server functionality.

Microsoft SQL Server 2005 is comprehensive, integrated data management and analysis software that enables organizations to manage mission-critical information reliably and confidently run today’s increasingly complex business applications. SQL Server allows companies to gain greater insight from their business information and achieve faster results for a competitive advantage.

*Note: a more detailed discussion of SQL Server can be found in the appendix of this document.*

## Aligning SQL Server to the Business Needs of the SMB

SQL databases emerged because spreadsheets were inadequate to meet the needs of users who not only needed to organize their data, but manipulate it as well. When you want to describe data in two ways, spreadsheets are perfect. “Column” and “row” are all you need. As the amount of data, and the complexity of its relationships increases, the spreadsheet structure becomes inadequate quickly; it becomes too difficult to track the relationships among columns and rows.



Either some relationships cannot be tracked, or multiple entry is required. The process becomes insufficient or cumbersome.

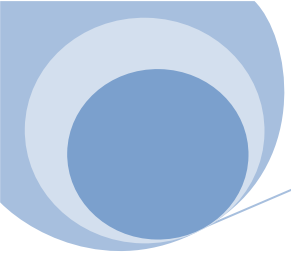
Consider a spreadsheet with customer contact information. You collect a variety of data points (name, address, phone number), but they all tie back to a particular record, a particular person. Now, take it a step further. There is other information that pertains to the contact, such as open orders or billing history. Quickly, you are into a level of complexity that a spreadsheet cannot support. Ad hoc reporting using a collection of spreadsheets would require hours to construct. Information would have to be pulled from a number of sources and aggregated into yet another spreadsheet. The cost of using your data escalates as a result. Essentially, you pay multiple times to use the information that you already have.

Databases were developed specifically to *describe* data in more than two ways. A relational database is a collection of two-dimensional spreadsheets. The tables have *keys* (automatically generated data) that are used to connect data from different tables. So, you could have your customer's contact information in one table, billing history in another and open orders in yet another. The open orders table might even connect to a table with a list of products. With a relational database, the data is tied together across as many tables as needed.

In this environment, everything you need is at your fingertips. Instead of having to assemble reports manually, a simple query can pull the data that you need when you need it. Customer contact information is linked to billing history and open orders, for example. If a customer is late in paying for an open order, you can access payment history at the same time to see if this is an isolated incident or part of an ongoing problem. Data pertaining to a number of different areas of your business is immediately available.

In order to pull data from different tables and make it meaningful, most databases have adopted a version of the Structured Query Language (SQL). SQL databases use a straightforward language to access, manipulate and present data. For most database users, the SQL queries are hidden. The average business user does not go directly to the tables. Instead, he uses an application (such as a payroll system) with a user interface that allows the user to do specific tasks, and the application creates the query that works with the database.

It is easy for a SQL database to become the technological nerve center of your business. The fact that it makes your information more available and useful is attractive. But, databases do require



an investment of time. Databases need to be maintained, with backups crucial to preserving the value and integrity of your data.

Because you are one step removed from your database (via the application), your data could be at risk. The database is not front of mind, while the application that accesses your database is. As a result, it is easy to forget to take regular backups of the database. In fact, many small businesses neglect their data, conducting backups infrequently, if at all. One unexpected computer failure could cost you years of accumulated information. Recovering from a data loss can take months, though many companies never recover.

When you move from spreadsheets to SQL databases, it may be time to hire a professional database administrator (DBA). The DBA's sole purpose is to manage your data, which involves the maintenance of the database, development of queries for rapid data access, and suggestions for improved business use. DBAs have specialized skill sets, and they would be responsible for scheduling, executing, and refining backups.

There are a number of ways to backup your data. In fact, most SQL databases include their own backup and recovery utility. But, these utilities can be difficult to use, especially without specific expertise in database management. Even if you master the use of your SQL database's backup utility, you must commit to monitoring it. Backups can fail for a number of reasons, including:

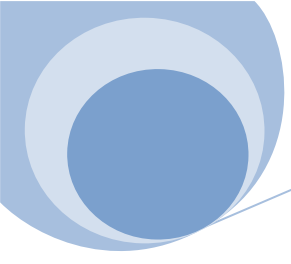
- Backup media is full
- Backup media is missing
- Changes to the server were not updated in the database

Ultimately, SQL database backups are not “set and forget.”

Even on the best of days, there are key issues to consider. Speed and security always top the list. Slow backups increase the likelihood that something will go wrong. For example, a longer backup increases the time during which a file is at risk to be corrupted. Also, slow backups increase the possibility that the database could “hang” on a particular row or record.

Security is crucial as well. Unencrypted backups almost invite impropriety from inside your organization as well as without. Most SQL databases, unfortunately, do not address these concerns. They might include some nascent functionality to address speed and security, but they are not default priorities.





Third-party solutions, though, can make the entire data backup process substantially easier, from ease of use through speed and security. A database backup and recovery solution should be easy to use, require absolutely SQL programming knowledge, be fast and secure. The right third-party solutions can improve how you store your data. As your business grows, you'll find that you are running out of space on your hard drive or have developed an unmanageable collection of CDs. A backup solution that compresses data will reduce the headaches around storage speed while further accelerating your backup. Additionally, the best backup solution will be just as concerned with allowing you to *restore* your files as it is to back them up. Backed-up files that cannot be restored are no better than data that was never backed up at all.

## Why Backup at all?

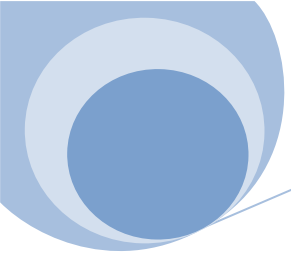
The concept of backing up data is based on copying it to a safe place to restore from when necessary. With an active database, you need to backup and protect more than just your database files and data. Databases include many components, such as transaction logs, that must be backed up as well to maintain a fully functioning database upon restoration that ensures the availability, continuity and integrity of the information stored there.

In evaluating today's recovery solutions, look for those that not only back up the necessary tables, data, and user-defined objects, but also treat the database as a more than a combination of files, you want the application to treat the database as a single, combined entity. Ideally, when the backup procedure starts, you want it to complete all the active transactions, makes a snapshot of the database and related files, and resume the transactions immediately. The database idle state is minimal and the backup will be written to the archive location while the database is online.

Using this process to back up the database ensures that the restored database will be operational upon startup. Since the backup is made at the point in time that the snapshot is created, no transactions made after the backup process starts will be included in the archive.

## Type of Backup Methodologies

There is a trio of popular backup methodologies available for optimal backup for comprehensive protection of your data from hardware failure, user errors or even natural disasters. These include:



A **full backup** contains all existing data at the moment of backup creation – a complete database or instance. You can recover the entire database by restoring the database from a full database backup to a chosen location. Enough of the transaction log is included in the backup to let you recover the database to the time when the backup finished. When the database is recovered, uncommitted transactions are rolled back. The restored database matches the state of the original database when the backup finished, minus any uncommitted transactions.

For a small database that can be backed up quickly, it is convenient to use only full database backups. However, as the database becomes larger, full backups take more time to finish and require more storage space. Therefore, for a large database, you might want to supplement full database backups with differential backups.

A full backup can form a base for further differential backups or can be used as a standalone archive.

A **differential backup** creates an independent file, containing all changes against the initial full backup.

A standalone full backup could be an optimal solution if you often roll back the database to the initial state. In this case, you do not need to re-create the initial full backup, so the backup time is not crucial and the restore time will be minimal.

Alternatively, if you are interested in saving the last data state to be able to restore it in case of a database failure, consider the differential backup. It is particularly effective if your data changes tend to be small in comparison to the full data volume.

**Transaction log** records all transactions and the database modifications made by each transaction. The transaction log is a crucial component of the database and, if there is a system failure, the transaction log might be required to bring your database back to a consistent state.

If you are backing up filegroups, the transaction logs will be backed up along with the full backup. Backing up transaction logs prevents data loss after the last backup and lets you restore the database state to an arbitrary point in time in order to undo the harmful changes.

To choose the appropriate backup strategy you should identify the requirements for the availability of your own business and technical requirements. Your overall backup strategy defines the type and frequency of backups as well as the type and capacity of the hardware required for the archive location.





## Which backup strategy is right for me?

Follow the recommendations below to define the best backup strategy for your organization:

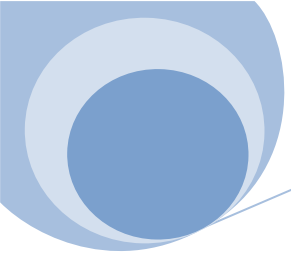
- Database activity is low to medium:
  - Full backup once a week
  - Differential backup once a day
  - Transaction Logs backup every two to four hours
- Database size is small to medium, but activity is high:
  - Full backup twice a week
  - Differential backup twice a day
  - Transaction Logs backup every hour
- Database size is large and activity is high, recovery model is Full or Bulk-Logged:
  - Full backup once a week
  - Differential backup once a day
  - Transaction Logs backup every twenty minutes
- Database size is large and activity is high, recovery model is Simple:
  - Full backup once a week
  - Differential backup twice a day.

## Recovery models

There are three database recovery models: Full, Simple and Bulk-logged. In most cases databases use the simple or full recovery models.

### Full Recovery Model

Full recovery model requires backing up logs, which allows you to restore to the point in time just before the data was corrupted and prevent data loss. The disadvantage of this recovery model is that it requires more storage space and makes restoration slower and more complicated.



## Simple Recovery Model

The simple recovery model does not store transaction logs, so you can recover your databases only to the point in time when the last backup was created. Therefore it is recommended that you create backups (full or differential, in accordance with the backup strategy you defined) often enough to prevent significant data loss. It is recommended to use the full recovery model for critical databases.

## Bulk-logged Recovery Model

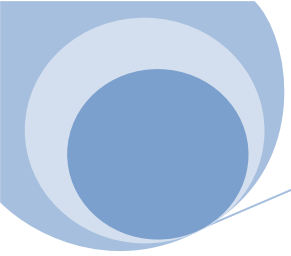
Bulk-logged recovery model uses transaction logs for non-bulk operations fully logged and bulk operations minimally logged. Data may be lost when restoring from bulk operations (e.g. SELECT INTO).

The bulk-logged recovery model requires log backups. It is an adjunct of the full recovery model and provides recovery to the end of any backup (but not to any point-in-time). If the log was damaged or bulk-logged operations occurred after the latest log backup, data changes will be lost. This is the least secure recovery model.

A word about **Compression**. While “Moore’s Laws” has more than proven its theoretical structures — *getting twice the amount for storage for half the price* — the cost of storage is indeed going down. Still, the amount of data is exploding at an exponential rate. To ensure your organization is saving space in your preferred storage destination, consider vendors that offer archive compression. Ideally, it is wise to choose products that promote compression “on-the-fly,” so that you do not need to move giant packets across the network and in turn will not consume large amounts of bandwidth, unnecessarily.

## Why Acronis® Recovery™ For MS SQL Server

Acronis Recovery for MS SQL Server offers a fast and reliable disaster recovery solution to protect your SQL database. Acronis Recovery for MS SQL Server provides proven database backup technology that will drastically reduce disaster recovery time so you can be running again in minutes instead of hours.



One-step Recovery and Automated Recovery to Point-of-Failure reduce downtime and assist your organization in improving your recovery time objective (RTO). Rolling Snapshot enables near-instantaneous recovery in cases of human error or logical data corruption.

Acronis Recovery for MS SQL Server is an excellent complement to the award-winning Acronis True Image™ suite of disaster recovery and system migration products that use patented disk-imaging technology. Together they deliver comprehensive server system backup and restore plus full SQL database protection — a winning disaster recovery plan combination.

Comprehensive recovery requires more than just data backup. Your MS SQL database contains tables, logs, and other components that structure the data. Using Acronis Recovery for MS SQL Server, a cohesive and intuitive backup solution, is critical to ensure a secure live database backup that can be quickly recovered.

Recovery is now made easy with automated system restore to point-of-failure. There is no need to walk through menus to get your system back online. This one-step process will return your database to the last known good state just before failure. Erroneous transactions will no longer cost you hours, or even minutes, in recovery time.

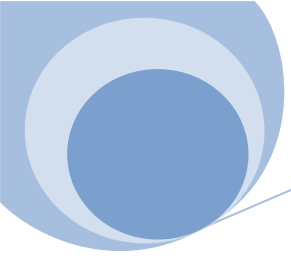
This powerful application includes an intuitive, wizard-driven GUI that guides you through the scheduling process and reduces the possibility for errors. The Backup Strategy Assistant creates a Disaster Recovery Plan for your environment, delivering step-by-step instructions for recovery. Anyone, whether an experienced data base administrator (DBA) or not, can schedule backup jobs and restore a system rapidly. Acronis even provides FTP capability for storing your backup on any FTP server, worldwide.

## Features Unique to Acronis Recovery for MS SQL Server

### FTP Support

For those organizations choosing to store data off-site to meet legal requirements, (such as Sarbanes-Oxley Act compliance, for example), or simply to insure the integrity (and recoverability) of a database during a crisis such as fire or floods, Acronis offers FTP support as an integral feature of its recovery methodology for MS SQL Server.

While the preferred mode of off-site recovery is most closely aligned to a tape-only or a disk-to-tape solution, by being able to recover to an FTP server off-site — as the data itself is being backed



up, for example — an SMB SQL administrator avoids having to include a third party in the process (such as a company contracted to take and store the company's tapes off site).

For SMBs in particular, whose databases are limited to a couple hundred gigabytes of data, Acronis Recovery for MS SQL Server will enable you to save, restore and recover data on the fly to an FTP server for each individual backup and in place of having to move an entire dataset to a given location all at once. For SMBs that prefer the flexibility of having off-site storage but who don't want to rely on tape or contracting with a third party for archival storage of tapes, FTP support is a compelling, as well as cost-effective solution.

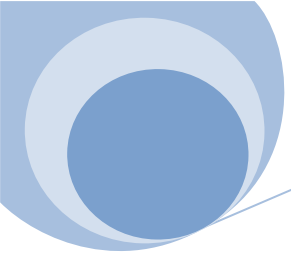
## Automated Recovery to Point-of-Failure

A popular approach to recovering data within the industry is the concept of recovery to point-in-time. By its very definition it suggests that you are restoring to a slice in time where you believe the corruption or database misfire took place.

By employing recovery to point-of-failure methodology found in Acronis Recovery for MS SQL Server, there is no need to walk through a menu or series of menus to get your system back online. This one-step process returns your database to the last known good state just before failure. Erroneous transactions will no longer cost you hours or even minutes in recovery time.

For example, as an SMB SQL administrator, it's likely you are continuously writing data with incremental saves set for every 15 to 30 minutes. However, if your MDF (Master Data File) or LDF (Log Data File) become corrupted (and it's not something you're immediately aware of), the time between the last good save and the time in which you were not writing is considered a *threshold for lost data* and everything that was entered between those two deltas in time is considered lost, particularly if you're only able to restore to a point-in-time. As a result, when you run your restore you're running it against your transaction logs and asking them, through commands, to restore to "this point in time" (e.g. wherever the transaction logs have defined that an error has occurred).

Automated Recovery to Point-of-Failure methodology goes beyond the typical Transaction Log backups. Consider: a Transaction Log backup isn't just about backing up an individual file, but backing up an actual log (e.g. this transaction happened at this time wherein, if I have a backup that covers two points in time — say, 11 a.m. and 11:15 a.m. for example — as long as I have a backup running through 11:15 a.m., I can restore back to the individual transactions or point of



time in that log). However, let's say you're doing backups, in an extreme case, only once an hour; in this case 11 a.m. and noon. At 11:59 a.m. your entire database shuts down due to a corruption in the MDF file. So long as your transaction log remains uncorrupted and functional, you have everything you need to recover your database. Simply task this feature to *recover to the point-of-failure* and recover all of the MDF and LDF back to 11a. m. A further advantage: Any transactions already incorporated into the backup are not "overwritten" through the restoration process

## Disaster Recovery Plan

While disasters by their very nature cannot be "planned for," having a process in place to address them when they do happen can and should be planned *and tested*. In SMB environments and full-fledged enterprises alike, the disaster recovery plan is often something that is manually documented. This includes detailed instructions, in writing, on such topics as where backup files are stored and a detailed process for running a full restoration.

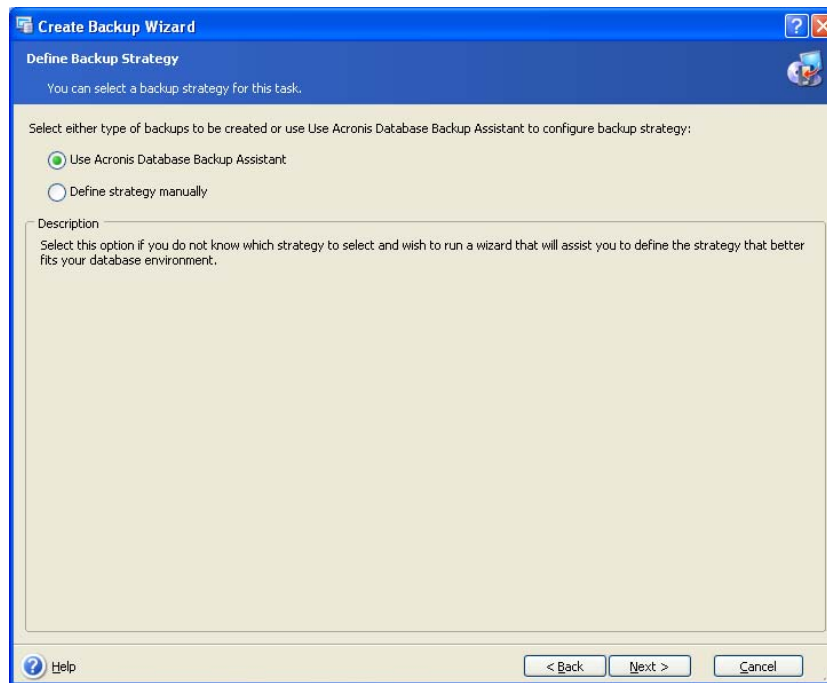
Acronis Recovery for MS SQL Server allows you to create an automated disaster recovery plan. This plan provides all the necessary information about the server and databases in the form of step-by-step instructions restoring the entire database in case of failure, disaster or data corruption. It automatically creates and e-mails disaster recovery plans, with step-by-step instructions for recovering databases, to a pre-determined distribution list. This guided process makes it possible for staff to restore databases quickly, even without DBA skills and helps to reduce issues during the recovery of databases.

This step-by-step process enables SMBs in particular, which might not have a dedicated DBA, to recover critical databases rapidly using existing IT staff who do not have the specialized training and skills of a DBA. In fact, using the step-by-step instructions, the person performing the database recovery need not have any technical background at all, for that matter.

The SMB SQL administrator can choose to back up one or more filegroups or individual files when the database size and performance requirements make it impractical to create full database backups. If you choose to back up an individual file instead of the full database, make sure that all files in the database are backed up regularly and create separate transaction log backups for the databases with files or filegroups you back up individually. After restoring a file backup, you have to apply the transaction log to roll the contents of the file forward to make it consistent with the rest of the database.

As with instances and databases, you can create full and differential backups for filegroups as well, but this is available for Microsoft SQL Server 2000 only.

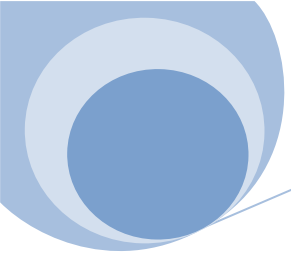
It is recommended to update and test the disaster recovery plan from time to time to be sure that the company's staff is really able to recover lost or corrupted data.



It is extremely important for all companies as well as for an individual user to have detailed instructions describing the way to recover lost data fast and efficiently in case a disaster occurs, a disaster recovery plan. This plan contains all the necessary information to guide you through the recovery process. While creating a recovery plan, include scenarios for different kinds of disaster (loss of database server, data corruption, complete loss of database etc).

In general a disaster recovery plan includes step-by-step instruction for each kind of disaster, server's hardware and software configuration.

Acronis Recovery for MS SQL Server generates disaster recovery plan automatically according to the settings you specified. Using **Create Disaster Recovery Plan** wizard allows you to generate and view the Disaster Recovery Plan immediately or schedule receiving it via e-mail after each



update. With the help of **Create Disaster Recovery Plan** wizard choose whether you want to receive a Disaster Recovery Plan right after finishing this wizard or after each new update.

Note: A typical database restoration plan allows the restoration of a full back up and leaving the database in a so-called “open state” where you are restoring to the latest differential or whatever differential falls before the point in time you want to restore (and then restoring the incremental). With Acronis Recovery for MS SQL Server, the application that supports the Disaster Recovery Plan knows it cannot get to that point in time without including each of the preceding backups — full, differential and incremental — so it will automatically roll those up for you as part of the Disaster Recovery Plan. As a result, you will have the ability to point to any file within – even to a point in time – and restore what you need in order to recover completely and intact from a database disaster.

Remember, and this is probably the most critical aspect of developing a plan, the best way to ensure the disaster recovery plan will work when a disruption occurs is to test it frequently. Make sure that the steps that you put into the plan have been tested by the people who will be required to implement them. If your plan calls for staffers without any technical experience to restore a SQL server, select one or two members of your administrative staff to perform the task. As noted earlier, backing up the database isn’t enough; you need to be able to restore it to a known, working condition.

## How Does Acronis Recovery for MS SQL Server Work?

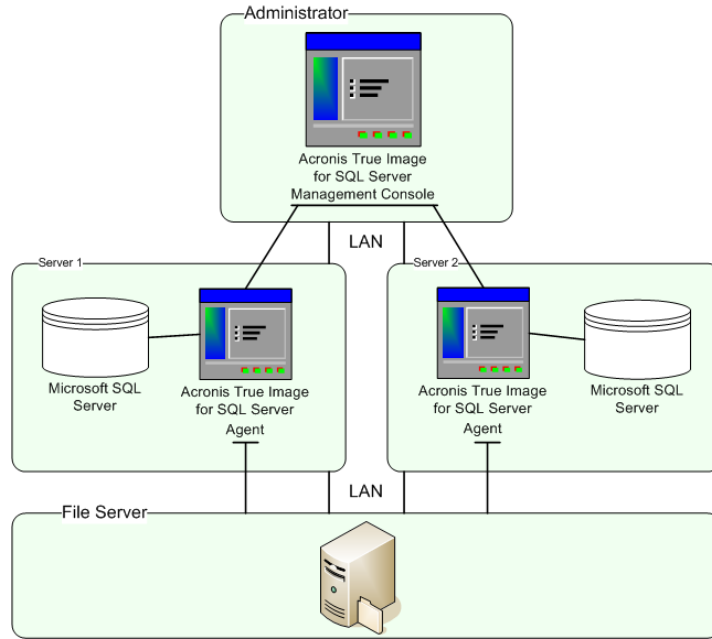
Acronis Recovery for MS SQL Server for SQL Server Agent is installed on the computers on which you want to backup/restore a SQL Server database.

Acronis Recovery for MS SQL Server for SQL Server Management Console is installed on the computer from which you plan to manage operations processes on remote or local database servers.

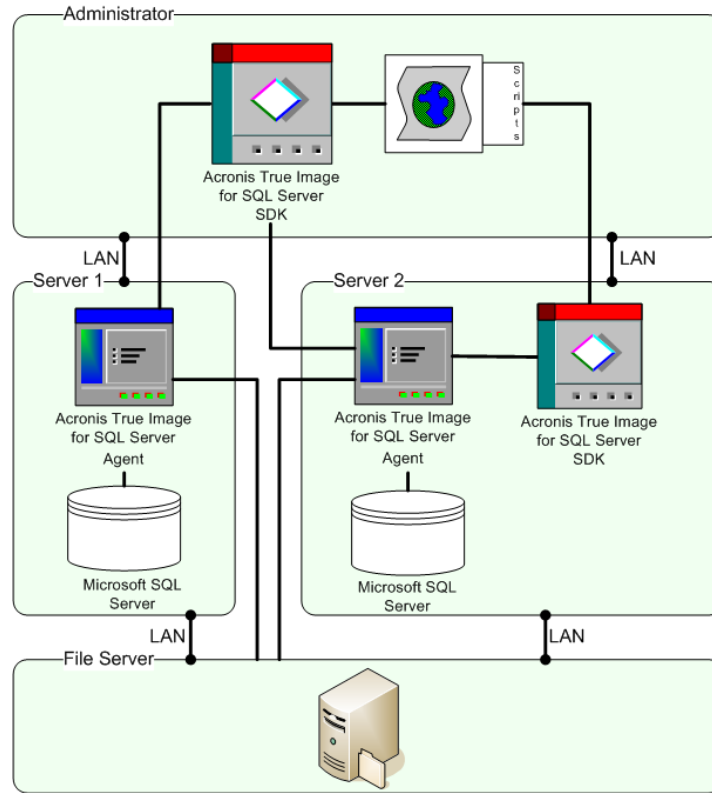
After issuing a backup or restore command from the Management Console, Acronis Recovery For MS SQL Server for SQL Server sends a request to the Acronis Recovery for MS SQL Server Agent to retrieve the required instance or database from Microsoft SQL Server and sends it to Acronis Recovery for MS SQL Server, which backs up the selected data.

Included below are diagrams describing interaction between Management Console, Agent and software development kit (SDK).





**Interaction between Acronis Recovery for MS SQL Server and Acronis Recovery for MS SQL Server Agents**

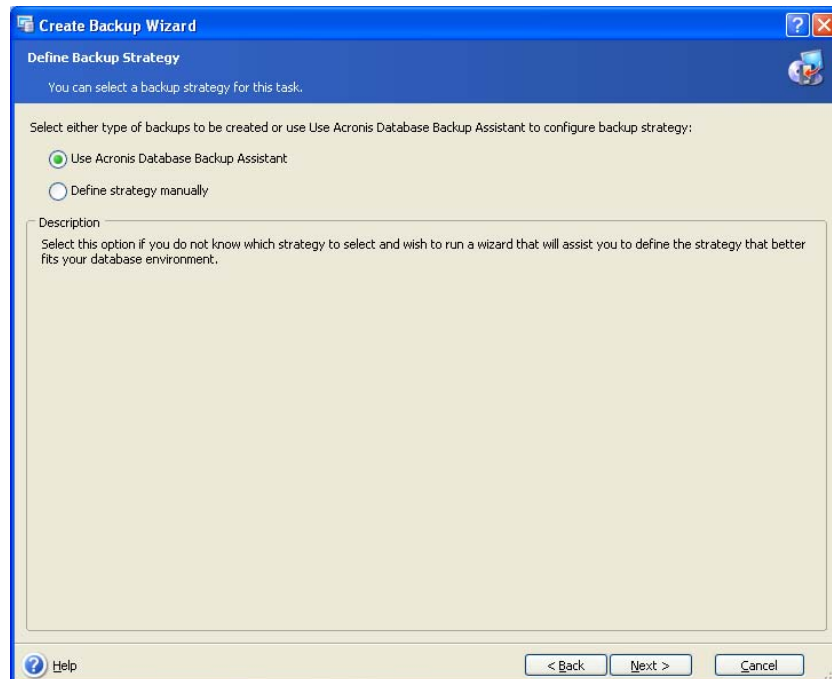


## Interaction between Acronis Recovery for MS SQL Server SDK and Acronis Recovery for MS SQL Server Agents

Backup is crucial to maintaining timely and consistent record of your databases for recovery in case of failure. Acronis Recovery for MS SQL Server offers an easy and flexible process for creating backup archives. Note: Acronis Recovery for MS SQL Server backs up databases as well as separate filegroups.

You can choose to back up one or more filegroups when the database size and performance requirements make it impractical to create full database backups. If you choose to back up an individual file instead of the full database, make sure that all files in the database are backed up regularly, and create separate transaction log backups for the databases whose files or filegroups you back up individually. After restoring a file backup, you have to apply the transaction log to roll the contents of the file forward to make it consistent with the rest of the database.

As with instances and databases you can create full and differential backups for filegroups as well, but this is available for Microsoft SQL Server 2000 only.

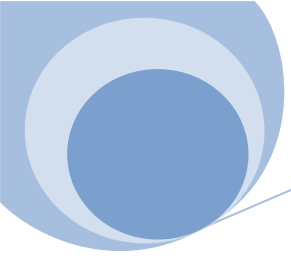


Thus, in case of disaster you only have to take the last version of Disaster Recovery Plan (printed or e-mailed) and follow the instructions to restore your databases.

## Summary

Whether you're running a small business that's ready to take the next step in its growth or an SMB that's ready to "grow into" an enterprise with database tools to match, Acronis Recovery for MS SQL Server offers a fast and reliable disaster recovery solution to protect your SQL database. Acronis Recovery for MS SQL Server provides database backup technology that will drastically reduce disaster recovery time so you can be running again in minutes instead of hours. One-step Recovery and Automated Recovery to Point-of-Failure reduce downtime and assist your organization in improving your recovery time objective (RTO).

Acronis Recovery for MS SQL Server is an excellent complement to the award-winning Acronis True Image suite of disaster recovery and system migration products that use patented disk-imaging technology. Together they deliver comprehensive server system backup and restore plus full SQL



database protection – a winning disaster recovery plan combination for the small and medium-sized business owner.

© Acronis, Inc. 2008. Reproduction in whole or in part without written permission is prohibited.

A portion of this white paper's research and findings are reprinted here with the permission of Nucleus Research, Wellesley, MA. For further information please visit [www.NucleusResearch.com](http://www.NucleusResearch.com).

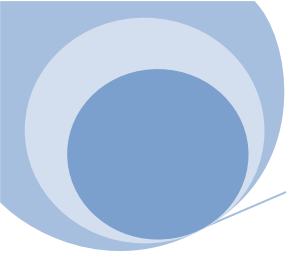
## Appendix

### Microsoft SQL Server (A Technical Overview)

The architecture of Microsoft SQL Server is broadly divided into three components: *SQLOS*, which implements the basic services required by SQL Server, including thread scheduling, memory management and I/O management; the *Relational Engine*, which implements the relational database components including support for databases, tables, queries and stored procedures as well as implementing the type system; and the *Protocol Layer*, which exposes the SQL Server functionality.

**SQLOS** is the base component in the SQL Server architecture. It implements functions normally associated with the operating system — thread scheduling, memory management, I/O management, buffer pool management, resource management, synchronization primitives and locking, and deadlock detection.

Because the requirement of SQL Server is highly specialized, SQL Server implements its own memory and thread management system, rather than using the generic one already implemented in the operating system. It divides all the operations it performs into a series of *Tasks*, both background maintenance jobs as well as processing requests from clients. Internally, a pool of worker threads is maintained, onto which the tasks are scheduled. A task is associated with the thread until it is completed, only after its completion is the thread freed and returned to the pool. If there are no free threads to assign the task to, the task is temporarily blocked. Each worker thread is mapped onto either an operating system thread which are user mode threads, which is used to implement co-operative multitasking. Using the latter, even though all the book-keeping jobs of thread management has to be implemented in SQLOS (in addition to the native OS implementation), it can optimize for the particular use. SQLOS also includes synchronization primitives for locking as well as monitoring for the worker threads to detect if they have entangled themselves into a deadlock and takes necessary measures to recover from the situation.



SQLOS handles the memory requirements of SQL Server as well. Reducing disk I/O is one of the primary goals of specialized memory management in SQL Server. As such, it maintains a buffer pool, handled by SQLOS, which is used to cache data pages from the disc as well as to satisfy the memory requirements for other components, including query processor, and other internal data structures. SQLOS also has to take care that the memory allocated is used efficiently, as such it monitors all the memory allocated from the buffer pool, ensuring that the components return unused memory to the pool, as well as shuffling data in and out of the cache to make room for newer data. For changes that are made to the data in buffer, SQLOS writes the data back to the disc lazily, that is when the disc subsystem is either free, or there have significant number changes made to the cache, while still serving requests from the cache. For this, it implements a *Lazy Writer*, which handles the task of writing the data back to persistent storage.

SQL Server normally supports up to 2 GB memory on x86 hardware, although it can be configured to use up to 64 GB if Address Windowing Extension is used with a supporting OS. For x64 hardware, it supports 8 TB of memory, and 7 TB for IA-64 systems. However, when running x86 versions of SQL Server on x64 hardware, it can access 4 GB of memory without any special configuration.

**The Relational engine** is the component that implements the relational data store using the capabilities provided by SQLOS, which is exposed to this layer via the private SQLOS API. It implements the type system to define the types of the data that can be stored in the tables, as well as the different types of data items (such as tables, indexes, logs etc) that can be stored. It includes the storage engine, which handles the way data is stored on persistent storage devices, as well as implement methods for fast access to the data. The storage engine implements log-based transaction so as to ensure that any changes to the data are ACID (Atomicity, Consistency, Isolation, Durability) compliant. It also includes the query processor, which is the component that allows data to be retrieved. The specification of what needs to be retrieved is provided in the form of a SQL query, which it optimizes and translates into the sequence of operations needed to retrieve the data. The operations are then scheduled on to the worker threads, which are scheduled for execution by SQLOS.

**The Protocol layer** implements the external interface to SQL Server. All operations that can be invoked on SQL Server are communicated to it via a Microsoft-defined format, called Tabular Data Stream (TDS). TDS packets can be encased in other physical transport dependent protocols, including TCP/IP, Named pipes, and Shared memory. Consequently, access to SQL Server is available over these protocols.



## Glossary

**Address Windowing Extension:** a Microsoft Windows Application Programming Interface that allows a 32-bit software application to access physical memory greater than 4GB. The process of mapping an application's virtual address space to physical memory is known as “windowing”

**ACID:** set of properties that guarantee that database transactions are processed reliably. In the context of databases, a single logical operation on the data is called a transaction. For more on ACID, see its Wikipedia entry at <http://en.wikipedia.org/wiki/ACID>

**API:** a source code interface that an operating system or library provides to support requests for services to be made of it by computer programs

**Buffer:** a region of memory used to temporarily hold data while it is being moved from one place to another

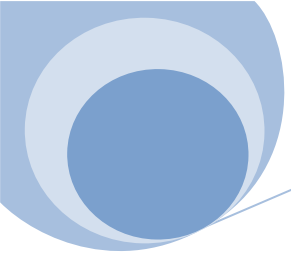
**Cache:** a temporary storage area where frequently accessed data can be stored for rapid access

**Cooperative Multi-Tasking:** Early multitasking systems consisted of suites of related applications that voluntarily ceded time to each other. This approach, which was eventually supported by many computer operating systems, is today known as cooperative multitasking. Although it is rarely used in larger systems, cooperative multitasking enables the running of multiple applications simultaneously

**Deadlock:** Deadlock refers to a specific condition when two or more processes are each waiting for another to release a resource, or more than two processes are waiting for resources in a circular chain

**Locking:** A lock is used when multiple users need to access a database concurrently. This prevents data from being corrupted or invalidated when multiple users try to write to the database. Any single user can only modify those database records (that is, items in the database) to which they have applied a lock that gives them exclusive access to the record until the lock is released

**Log-Based:** Records events in a certain scope in order to provide an audit trail that can be used to diagnose problems



**Named Pipes:** A traditional pipe is "unnamed" because it exists anonymously and persists only for as long as the process is running. A named pipe is system-persistent and exists beyond the life of the process and must be "unlinked" or deleted once it is no longer being used. Processes generally attach to the named pipe (usually appearing as a file) to perform IPC (inter-process communication)

**Relational Data Store:** A **relational database** is a database that conforms to the relational model, and refers to a database's data and schema (the database's structure of how that data is arranged)

**Shared Memory:** Shared memory is a memory that may be simultaneously accessed by multiple programs with an intent to provide communication among them. Depending on context, programs may run on the same physical processor or on separate ones

**SQL:** a computer language designed for the retrieval and management of data in relational database management systems, database schema creation and modification, and database object access control management. *SQL* now stands for Structured Query Language

**Tabular Data Stream:** A protocol to transfer data between a database server and client

**TCP/IP:** The Internet protocol suite is the set of communications protocols that implement the protocol stack on which the Internet and most commercial networks run. The acronym stands for Transmission Control Protocol/Internet Protocol

**Thread:** Threads are a way for a program to fork, or split, itself into two or more simultaneously (or pseudo-simultaneously) running tasks. Threads and processes differ from one operating system to another but, in general, a thread is contained inside a process and different threads of the same process share some resources while different processes do not

**Transaction:** a unit of interaction with a database management system or similar system that is treated in a coherent and reliable way independent of other transactions

To find out more about Acronis True Image products:

Call +1 877 669-9749

E-mail: [sales@acronis.com](mailto:sales@acronis.com)

For OEM inquiries:

Call +1 650 875-7593

E-mail: [oem@acronis.com](mailto:oem@acronis.com)