MICROSOFT SQL SERVER 7.0 ADMINISTRATORS GUIDE pdf

1: Read Microsoft SQL Server Administrator's Guide with CDROM Ebook Online - Video Dailymotion

This narrative reference covers all the major activities in which SQL Server administrators must engage. It also includes tutorials for complex issues. This book helps administrators avoid costly mistakes and dead ends by providing clear explanations, justifications for preferred practices, and comprehensive checklists for administering SQL (Structured Query Language) Server

One major enhancement to SQL Server 7. LazyWrtier and Read-Ahead Manager are self-tuning. This reduction in tuning requirements saves valuable administrative time that can be applied to other tasks. This allows SQL Server to automatically adjust the configuration of the database server as factors affecting the database server change. Manual tuning of LazyWriter was sometimes necessary. There is no longer the need to manually tune "free buffer" and "max lazywriter io. Manual tuning of Checkpoint was sometimes necessary. In versions of SQL Server prior to the 7. The SQL Server 7. This default setting will maintain recovery times less than one minute for all databases as long as there are no exceptionally long-running transactions present on the system. Manual tuning of the Log Manager was sometimes necessary. This separating of the log file management from the data cache management brings enhanced performance for both components. This option has been removed from SQL Server 7. This helps SQL Server run at peak performance even as user load and queries change over time. RAM is a limited resource. A major part of any database server environment is the management of random access memory RAM buffer cache. Access to data in RAM cache is much faster than access to the same information from disk. But RAM is a limited resource. Too much unneeded data and index information flowing into buffer cache will quickly push out valuable pages. Create and maintain good indexes. Disk queuing results in bad performance. Application and Query Tuning. This becomes especially important when a database server will be servicing requests from hundreds or thousands of connections through a given application. This logged workload can then be submitted to SQL Server Index Tuning Wizard so that index changes can be made to help performance if necessary. This document describes key Performance Monitor counters to watch. Then watch Performance Monitor for signs of disk activity or queuing issues. This configuration option is dynamic that is, it does not require a stop and restart of SQL Server to take effect. If the number of connections actively submitting batches is greater than the number specified for max worker threads, worker threads will be shared among connections actively submitting batches. The default of will work well for many installations. Note that the majority of connections spend most of their time waiting for batches to be received from the client. Worker threads take on most of the responsibility of writing out dirty 8-kilobyte KB pages from the SQL Server buffer cache. Buffer Managerâ€”Free Buffers" to see if this value dips. Optimally, LazyWriter keeps this counter level throughout SQL Server operations, which means that LazyWriter is keeping up with the user demand for free buffers. Compare drops in free buffer level to any disk queuing to see if this is true. Monitor the current level of disk queuing in Performance Monitor by looking at the counters for Logical or Physical Disk: It is very important to be aware of the number of drives associated with the RAID array controller in order to properly interpret the disk queue numbers that Performance Monitor is reporting. For more information, search for the strings "freeing and writing buffer pages" and "write-ahead transaction log" in SQL Server Books Online. Dirty pages are any buffer cache pages that have been modified since being brought into the buffer cache. A buffer written to disk by Checkpoint still contains the page and users can read or update it without rereading it from disk, which is not the case for free buffers created by LazyWriter. Checkpoint aims to let worker threads and LazyWriter do the majority of the work writing out dirty pages. Checkpoint does this by trying an extra checkpoint wait before writing out a dirty page if possible. This provides the worker threads and LazyWriter more time with which to write out the dirty pages. To make Checkpoint more efficient when there are a large number of pages to flush out of cache, SQL Server will sort the data pages to be flushed in the order the pages appear on disk. Be sure to watch disk read queuing in addition to disk write queuing. One thing that helps guarantee recoverability is the SQL Server logging process. Reading the transaction log and applying the transactions to SQL Server after a server stoppage is referred to as recovery. For more information on caching

controllers, refer to the section later in this document titled "Effect of on-board cache of hardware RAID controllers. Log Manager Object," and "optimizing transaction log performance. Query Processor communicates situations that would benefit from read-ahead scans to Read-Ahead Manager. Large table scans, large index range scans, and probes into clustered and nonclustered index binary trees or B-trees are situations that would benefit from a read-ahead. When it is necessary to retrieve a large amount of data from SQL Server, read-ahead is the best way to do it. The IAM is an 8-KB page that tightly packs information through a bitmap about which extents within the range of extents covered by the IAM contain required data. Combining the query information from Query Processor and quickly retrieving the location of all extents that need to be read from the IAM page s , Read-Ahead Manger can construct multiple sequential read requests. The solution is a general performance tuning goal, and that is to make sure that all SQL queries are tuned such that a minimal number of pages are brought into buffer cache. This would include making sure you use the right index for the right job. Save clustered indexes for efficient range scans and define nonclustered indexes to help quickly locate single rows or smaller rowsets. Note that if you only plan to have one index in a table and that index is for the purposes of fetching single rows or smaller rowsets, you should make the index clustered because clustered indexes will be faster than nonclustered indexes but not by the same dramatic scale as is the case for range scans. SQL Server is certainly no exception to this philosophy. This is illustrated by the following calculations: This is one reason why it is important to eliminate and prevent page splitting. Log Manager will write sequentially to the log files in sizes ranging up to 32 kilobytes. It is worthwhile to take a moment to explain what these terms basically mean in relation to a disk drive. A single hard drive consists of a set of drive platters. With respect to SQL Server, there are two important points to remember about hard drives: The time difference between the nonsequential versus sequential case is significant, about 50 milliseconds per nonsequential seek versus approximately milliseconds for sequential seeks. Note that these times are rough estimations and will vary based upon how far apart the nonsequential data is spread around on the disk, how fast the hard disk platters can spin RPM , and other physical attributes of the hard drive. It is important to remember that it takes almost as much time to read or write 8 kilobytes as it does to read or write 64 kilobytes. So, mathematically speaking, it is beneficial to try to perform KB disk transfers as much as possible when more than 64 KB of SQL data needs to be transferred, because a KB transfer is essentially as fast as an 8-KB transfer and eight times the amount of SQL Server data is processed for each transfer. Readers interested in more detail about physical hard drives should refer to the Compaq white paper "Disk Subsystem Performance and Scalability," the location of which is mentioned in the "Finding More Information" section at the end of this document. Typical RAID controllers have an advertised transfer rate of about 40 megabytes per second or very approximately 2, disk transfers per second. Peripheral Component Interconnect PCI buses have an advertised transfer rate of about megabytes per second and higher. The actual transfer rates achievable for a device will differ from the advertised rate, but that is not important for our discussion here. Assuming a RAID controller can handle 40 MB per second, it is possible to roughly calculate the number of hard drives that should be associated with one RAID controller by dividing 40 by 9. This means that at most 8 hard drives should be associated with the single controller in the nonsequential KB scenario. The random 8-KB data transfer scenario requires the most drives. Divide 40 by 0. Another way to figure out how many drives should be associated with a RAID controller is to look at disk transfers per second instead of looking at megabytes per second. Moving onto the PCI bus. Note that most large servers come with more than one PCI bus, so this would increase the number of RAID controllers that could be installed in a single server. This is because the calculations assume all sequential or all nonsequential data access, which is not likely to ever be the case in a production database server environment. But hopefully this section has helped to foster some insight into what advertised transfer rates really mean to SQL Server. The benefits of RAID are: RAID provides protection from hard disk failure and accompanying data loss with two methods: Mirroring is implemented by writing information onto two sets of drives, one on each side of the mirrored pairs of drives. Most RAID controllers provide the ability to do this failed drive replacement and rebuilding from the other side of the mirrored pair while Windows and SQL Server are online commonly referred to as "Hot Plug" capable drives. One advantage of mirroring is that it is the best-performing RAID option when fault tolerance is required. The other

advantage is that mirroring provides more fault tolerance than parity RAID implementations. Mirroring can sustain at least one failed drive and may be able to survive failure of up to half of the drives in the mirrorset without forcing the system administrator to shut down the server and recover from file backup. The disadvantage of mirroring is cost. The disk cost of mirroring is one drive for each drive worth of data. Parity is implemented by calculating recovery information about data written to disk and writing this parity information on the other drives that form the RAID array. If a drive should fail, a new drive is inserted into the RAID array and the data on that failed drive is recovered by taking the recovery information parity written on the other drives and using this information to regenerate the data from the failed drive. RAID 5 and its hybrids are implemented through parity. The advantage of parity is cost. To protect any number of drives with RAID 5, only one additional drive is required. Parity information is evenly distributed among all drives participating in the RAID 5 array. The disadvantages of parity are performance and fault tolerance. Also, RAID 5 can sustain only one failed drive before the array must be taken offline and recovery from backup media must be performed to restore data. General Rule of Thumb:

MICROSOFT SQL SERVER 7.0 ADMINISTRATORS GUIDE pdf

2: Database Administratorâ€™s Guide to SQL Server Database Engine .NET CLR Environment

I'm also ordering MS SQL Server Admin Train Kit(ISBN: as the Transcender SQL refers to the "SQLSBO"(SQL Server Books online) for almost all the questions; and the training kit is pretty much the same as SQLSBO but I like the written books better--and Amazon is selling at a great price %50 off! the buck retail price.

They are not specifications for this product and are subject to change. There are no guarantees, implied or otherwise, that these features will be included in the final product release. For some features, this document assumes that the reader is familiar with SQL Server features and services. For background information about SQL Server features and services, see the official product Web site at http: This white paper provides information that helps database administrators ensure successful, risk free, and stress-free adoption of Microsoft. Thus, the audience for this white paper is the database administrator. For a developer perspective of the. Along with this breadth of programming options comes the need to consider which set of tools is appropriate for each task. Although many tasks can be accomplished in multiple ways, each has pros and cons. Thus, finding the best tool for the job is critical for an application to perform and scale with load and growing business usage. Some of the questions the DBA needs to ask include the following: Should the system handle this data as XML or should it be shredded and stored relationally? Should this process, and all its complex pieces, be handled synchronously or asynchronously? Should this business logic, this calculation, or this added security option be handled in the client application, the middle-tier, or the back-end database? Should data analysis be handled in the relational database or through the Business Intelligence engine? Should complex business logic, traditionally running on middle-tier servers, remain in the middle-tier or migrate to the SQL Server platform? What mix of clients and servers are running in the infrastructure. Is there a need to support Windows clients, Unix clients, or both? In most database development projects, the role of technology selection and the structural design of components that interact with the database falls on the database administrator DBA. This is the person with final responsibility for managing and recovering that business data. Most DBAs adopt conservative attitudes to new technology. This is a natural instinct because, along with the benefits offered by new functionality, new technology can introduce new risks to stability and integrity. And, further, by taking the time to identify where it adds the most value and, perhaps more importantly, where it should not be used. Instead of leaving all features off permanently, a prudent DBA will tend to learn enough about the technology to determine where its use is appropriate and where its application makes the most sense. There is no need to understand every line of code in every language that the developer might use, but there needs to be enough confidence to be able to provide great operational support, maintenance, and troubleshooting. Across many of these new features, the key to proper usage is understanding, impact isolation, and strong control. Top Of Page Introduction to. Use Transact-SQL to write code that runs within the database. Code that is written as an extended stored procedure appears to users as a stored procedure and is executed in the same way. Parameters can be passed to extended stored procedures and they can participate in transactions and return both results and return status. NET, to write code that executes outside of the database and that passes in queries or invokes stored procedures and functions to access data. Each of these options has issues when the solution demands that data be integrated with functionality supplied by external libraries. For example, such as those provided with the. NET Framework, or that nontrivial mathematical operations be applied to the data, or if the requirement is for something more complex, such as a custom aggregation of data or a true user-defined data type. Each of the four options has limitations: Transact-SQL is excellent for set-based operations such as comparisons between tables but, due to the interpreted nature of the language, it can struggle to deliver good performance for computationally heavy tasks. Extended stored procedures are by their nature written in unmanaged code and execute within the context of the SQL Server process. A greater level of programming competency is required to create code that does not inadvertently leak memory or generate unhandled exceptions that can crash the entire SQL Server process. Extended stored procedures cannot provide in-process access to the Microsoft. NET Framework libraries without placing the server in an unsupported state. This requires that its interface be implemented in a compatible way and have further

restrictions on the amount of data that can be passed to the COM object in a single call. They can encourage inappropriate use of components that are not designed to be used in high-throughput scenarios, or that do not support multiple invocations by a single process. In the worst case, the component can attempt to display an error message window or other dialog on the SQL Server. External code can cause performance problems because data must leave the SQL Server process space and flow to the calling application. This data marshalling can be expensive for large volumes of data. None of the current options can be used to create first-class, custom aggregate functions or custom data types where first-class means running within the database as if it were a SQL Server primitive function or data type. With these limitations in mind, SQL Server integrates the. Thus, it enables database developers to place managed application code inside the SQL Server that is safe, secure, scalable, and feature rich. Code can be written as follows: User-defined functions scalar or table valued Stored procedures User-defined aggregates User-defined types The mapping of user-defined functions, stored procedures, and triggers to objects written in managed code is fairly intuitive. However, user-defined aggregates and types are less intuitive and extend the options of the database programmer in new ways: This enables complex statistical and data analysis in the database engine. User-defined types provide the programmer with the ability to define new types with custom behaviors. Combined with the power of the. NET Framework and third-party libraries, this new capability will allow strongly typed objects to be created instead of forcing a relational representation. High Performance Implementation SQL Server delivers high-performance access to managed code that runs inside the database server process. Unlike other database technologies that have provided a degree of integration with the. The integration is designed to avoid conflicting memory and CPU demands between database queries and programs. Additionally, the SQL Server and. Secure By Design, Default, and Deployment Microsoft strives continuously to deliver secure products to its customers. This change in philosophy ensures that potentially unused features are not enabled and left in an unprotected state. This will open the dialog below that enables the selection of a SQL Server instance and then the selection of each locked down option. Enable the Database Engine. Two views of the options are available: Before enabling features that increase the surface area of the SQL Server, it is recommended that the DBA ensure that their systems are: At the latest service pack and critical hotfix level obtained from Microsoft Update Configured according to their recommendations for secure systems these may be informed by Microsoft and other third-party vendor advice on server and infrastructure configuration Figure 2 The Database Engine. The user interface provides an easy way to view and set SQL Server instance level permissions. The SQL Server instance level features can also be controlled programmatically by: Figure 3 The illustration depicts both the dilemma and the choice that most DBAs will face. Some decisions are simple while other decisions are extremely complex, especially where there are many ways to implement the functionality and there is no absolute direction between the technology options. In these cases, prototyping becomes more important. A quick implementation using two or more competing options can make the choice much clearer. As previously stated, the professional DBA is conservative. As such, new technologies are not implemented lightly; they must first be understood and then carefully tested before they become integrated into production environments. To do otherwise would risk destabilizing the database and lead to user distrust of the integrity of the systems that the database hosts, and possibly threaten the job safety of the DBA. A further assumption is that developers are more willing to take risks and to adopt new tools that increase their productivity and extend the functional domain of the solutions they can offer their users. There is no doubt that the Database Engine. NET Framework programming model is powerful and that many developers will appreciate the ability to write in-database code in first-class programming languages. So, where do these two conflicting character types meet in order to ensure that their systems remain stable and that new productivity and functionality gains are realized? The answer is in finding balance, understanding the opportunities, and more importantly defining choices rationally, based on strict criteria for technology selection. This section suggests some guidelines that can be followed to achieve this middle ground. The first set of points provides guidance about where it might be a mistake to use the new functionality: NET Framework, especially if the set-based query is replaced by cursor-like behavior in the program. The section, Troubleshooting, contains information about this potential cost. Note that the opposite applies if complex

computations are taking place within the query. In this case, moving the logic into a. NET Framework program where the computation will be fully compiled can improve performance. The transition overhead between the Transact-SQL and. NET Framework execution environments is more noticeable for simple computations and basic relational data access. Long running, external calls While it is tempting to use the new functionality to further integrate existing business systems, it is important to take the time to ensure that the end-user experience does not get negatively impacted by calls to external APIs and external systems. These impacts can be especially visible in a user-defined function that might get called for every row of a table within a query. An external call that costs one second per row suddenly becomes unusable in an online system when applied to a modest 10,row table. Be aware that user-defined types have the following: An 8-K size limitation. They must fit on a single SQL Server data page. All data within the UDT is read and then rewritten if updated. The same size restriction applies to user-defined aggregates so care should be taken especially when concatenating large string objects. User-defined aggregates and online reporting User-defined aggregates cannot be used in combination with the SQL Server Indexed Views, so it is not possible to automatically pre-aggregate data for online report performance. If stale data is acceptable, then a separately created and maintained table that periodically caches the aggregated results can be used in place of an indexed view.

3: SnapManagerÂ® for MicrosoftÂ® SQL Server : Installation and Administration Guide

SQL Server offers many new possibilities and opportunities for SQL Server administrators, developers, and IT decision makers. But to make use of the new features found in SQL Server , companies must first upgrade their existing database environments.

Cannot read any data in user tables within a database. User-defined roles apply only at the database level, and are local to the database in which they were created. Application roles allow the application to take over the responsibility of user authentication. The existing permissions assigned to the user are dropped, and the security context of the application role is assumed. If the Multiprotocol Net-Library is used, the packet containing the password can also be encrypted. The best way of storing the username and password in an application is probably to store the information in a registry key. The key should be encrypted, and only the application should have the key to decrypt it. After application roles are activated, they cannot be deactivated. Application roles work with both authentication methods, and contain no members. Like user-defined roles, application roles exist within a database only. If, while an application is in the security context of an application role, another database is accessed, the access to the other database is, by virtue of permissions, granted to the guest account in that database. If the guest account has not been specifically granted access to the data, or does not exist, the objects cannot be accessed. In other words, application roles provide the security context within which the database object permissions are checked, but the identity of the actual user is not lost. Here is an example of an implementation using application roles. Permissions System There has been a change in the way that permissions are implemented. The public group was overridden by user-defined groups, which were overridden by permissions assigned to individual users. The hierarchy was clearly defined: If a user is a member of the sales, marketing, and research roles multiple group memberships are now possible , the user gets the sum of the respective permissions of each role. The most restrictive permission DENY takes precedence. This can be very useful. For example, if one finds that some database users are frivolously changing data, it would not be fair to remove permissions from all users, as the majority of the users are using the data responsibly. Therefore, Steps 1 and 2 are often repeated to group the users by access permissions required. The other approach to security is based on the use of roles, and is usually implemented in the manner illustrated in figure 2 below: Follow Steps 1 through 4 listed in the procedure above that accompanies figure 1. Object permissions are then assigned to the roles. However, after a user gains access to the server, the security mechanisms are identical. In each user accounts domain, global groups should be created to group users by job requirements. The users should then be placed into the appropriate global groups in their domain. The global group and local group requirements outlined earlier may seem like a lot for small, single domain networks; however, experience has shown that there is usually great value in doing this. The following Transact-SQL script creates a login for a non-trusted connection: The default database becomes pubs. The default database is the database to which the user is switched when she or he attempts to log in. Alternatively, the above can also be achieved using Visual Basic as follows: Permissions must be granted to allow users to access a database. The MMC creates a list of all accounts that have been granted the permission to log on to the server, and a selection needs to be made from this list. Although not a technical requirement, if you are using trusted connections it is strongly recommended that you create users with the same username in each database as the logon name. Some examples for the Transact-SQL statements required to grant permission to use a database are: Object permissions restrict access to objects such as tables, views, or stored procedures. Object permissions are dependent on the object being referenced. Permissions can be granted to roles and users. User-Defined Database Roles In an ideal environment, roles would not be necessary. In this case, roles can be used to group users by their permission requirements. Fixed server roles and fixed database roles are predefined and cannot be modified. Roles can be created with the following Transact-SQL code: Add oDbRole After a user-defined database role is created, it makes sense to add users, groups or other roles to it. Roles can be nested, although not in a circular manner, as this would not be productive. There are three methods used to set the appropriate permissions for users or roles within a database: The same example would

be coded like this in Visual Basic: Due to these similarities, the following examples will show the code only for existing database users. Revoking permissions deletes a previous grant or deny, while denying permissions prohibits access even when access permissions have been granted. The concept of ownership chains is established when permissions on an object are checked. For example, when a user accesses a view, the permissions on the view should be checked. But what about the permissions on the underlying table? A broken ownership chain is when an object does not have the same owner as its underlying objects. For example, if Bob creates a table, and Mary creates a view based on that table, there is a broken ownership chain. In relation to security, broken ownership chains specify where permissions should be checked over and above the original object accessed. This makes for a very practical model. The concepts of ownership chains are best explained with a detailed example. Assume that Bob owns a table. One day Sue sees Mary using this view, and exclaims how brilliant it is. Mary agrees to give Sue access to the view. Fortunately, there is a broken ownership chain, as Bob owns the table and Mary owns the view. The owner of the view does not own the underlying objects. If Sue has not been granted access to the table, she cannot use the view. This is because of the broken ownership chain. Effectively, a broken ownership chain guards against a user hijacking the access permissions. Conversely, if Bob decided to create the view and to deny Sue access to his table, but grant her access to the view, Sue would be able to access the view. This is because the permissions are checked only when Sue accesses the view. There is no broken ownership chain, so permissions for the underlying table are not checked. Because Bob has created both objects, he should understand that giving access to the view requires implicit access to the underlying objects. An example where Server 7. Users are not allowed to update the system tables directly, especially not those found in the master database. Users should be given the opportunity to change their passwords on a regular basis. At the same time, performance is increased because permissions do not require checking if the ownership chain is not broken. Securing Access Through Application Roles This section outlines how an application can make use of application roles. To use application roles, perform the following steps: Application role must be created. Application role must be assigned permissions. Client application activates the application role. The first two steps of this process are usually separated from the last two. The Transact-SQL script is as follows: And the same again in Visual Basic: Add oDbRole And to use the role: All other data sources cannot explicitly encrypt the password. In these cases, it is imperative to use an encrypted communications protocol with the server. A point often overlooked is that application roles are implemented per session. If your application opens multiple sessions and all sessions are required to use the same role, each session must first activate the role. This can be used to provide much more granular security than ever before. If the identity of the logged-on user is required, use the following SQL statement: Permissions on a registry key no longer control access to the server. If so, the user will not be granted access to the server. If the user is not specifically denied access, the server checks whether the user has been granted access directly or by virtue of a group membership. Then, the user proceeds to the appropriate default database where the user must also have been granted access. If access has not been granted for a particular set of logon credentials, the connection to the server is terminated. In this way, security can function identically in a trusted and non-trusted environment. This is particularly true when a lot of name lookups are done or the domain controller is connected over a slow WAN link. Another effect of this is that the usernames in system messages may not report an up-to-date name. This system table exists only in the master database. This view should not really be necessary, as system tables should not be accessed directly.

4: Microsoft SQL Server Performance Tuning Guide

This performance tuning guide is designed to help database administrators configure MicrosoftÂ® SQL Serverâ„¢ for maximum performance and help determine the cause of poor performance in a SQL Server environment.

Microsoft Corporation June 8, Summary: For example, suppose your corporate headquarters is in Chicago and you have regional offices in Seattle, Phoenix, Atlanta, and New York City. You want to keep a master price list for the products you sell at your home office, but you also want each regional office to have its own copy of the price list. Furthermore, each regional office tracks the orders it receives, and you want the corporate headquarters to have a master list of the order history from each regional office. One method you might use to accomplish this is to keep a price spreadsheet and an order history spreadsheet at the Chicago headquarters. You periodically copy the price sheet to each regional office and update the order history sheet at the home office with all the orders from each regional site. Although this method accomplishes your goal of keeping all the information current at each site, it can become difficult to manage. A better way would be to automate these tasks so that the consistency of the information is ensured while minimizing the administrative burden. A situation such as the one described above is called a distributed environment, in which multiple copies of the same information exist at multiple locations. This environment may have been created deliberately or it may have evolved unintentionally. A distributed environment may become more and more difficult to manage as it grows, sites are added, and the information being retained becomes more complex. At some point, you will probably need to find an easier way to manage data in a distributed environment. In SQL Server, replication refers to copying data in a database from one location to another. Publishers and Subscribers SQL Server uses a publisher-subscriber metaphor for its data replication feature. The publisher makes information data available. A distributor forwards any changes to the appropriate subscribers. The subscribers receive the information. As an analogy, think of a local newspaper organization. The news organization or company is the publisher, your delivery boy is the distributor, and you are the subscriber. Similarly, in SQL Server replication, the publisher maintains the source data and makes it available for replication. The distributor receives the data and any changes, and forwards the information to the subscribers, who receive the data and hold a copy of it. The data being replicated is referred to as a publication. The publication contains one or more individual articles. An article is a subset of the data that is being replicated. As with a newspaper subscription, in SQL Server replication you must subscribe to an entire publication. That is, you cannot subscribe to individual articles of a publication. Information in databases is stored in tables. When you designate a table to be a publication, you do not need to publish the entire table. This is called filtering the data to be published. You can apply either a vertical filter or a horizontal filter or both to a table. In SQL Server replication, there are two types of subscriptions: A push subscription is what we usually think of as a subscription. The publisher pushes the subscription to the distributor, who then sends it to the subscribers. The newspaper delivery model uses a push subscription. A pull subscription may sound like an oxymoron because we tend to think of subscriptions as being push by default. In a pull subscription, the subscriber initiates the subscription. For example, if you go down to the local news stand to get the newspaper, that would be considered a pull subscription. Replication Models There are also different replication models. In this model, one server can be set up as the publisher and distributor. The following picture illustrates this model: You have a central subscriber who receives data from multiple publishers. This model is helpful when you need to consolidate data at a central site. The following is diagram illustrates this model: Central subscriber replication model To go back to the scenario at the beginning of this article, you now know that an easy way to have each regional site maintain a current copy of your product price list and have your corporate headquarters receive the orders from each regional site is to use SQL Server replication. You can have your headquarters in Chicago set up as a publisher and distributor to send the product price table of your database to the regional offices which will be the subscribers , using the central publisher model. Considerations When planning to implement replication, you should carefully consider the following questions: What information do you want to publish? Replication will have an impact on your network. To save disk space, processing time, and

network bandwidth, you should replicate only the necessary data. If you do not need to replicate all the information in a table, you can use a horizontal or vertical filter or both to replicate only a part of the table. How up-to-date does your data need to be? For example, some people like to receive their newspapers daily, while others choose to receive only the Sunday paper. In SQL Server replication, this is referred to as transactional consistency. How independent do your subscribers need to be? When you go out of town for an extended period of time, you probably stop your newspaper subscription temporarily. Because replication happens across your network, you must consider cases where the subscribers are not connected to the network. Decide whether a subscriber will always be connected to the publisher or if it will only be connected some of the time. Also, decide whether or not subscribers will be able to modify the published data after they receive it from the publisher. This is referred to as subscriber autonomy. How often do you want the data to be synchronized? You can schedule replication to occur at predefined intervals, or you can specify that replication will happen on demand. What type of replication model do you want to use? Who will initiate the replication? There are other considerations to keep in mind as well, but these should help to get you started. The SQL Server 7.

5: Download SQL Server Service Pack 4 from Official Microsoft Download Center

Microsoft SQL ServerÂ¶. To be able to use the Microsoft SQL Server API to perform bulk data loads, first you need to follow these steps: Download the Microsoft Command Line Utilities for SQL Server.

6: Veritas NetBackup (tm) for Microsoft SQL Server Administrator's Guide

Tai Yee Microsoft Corporation. June 8, Summary: This article gives you a broad general view of the replication feature of MicrosoftÂ® SQL Serverâ„¢ and how you can use replication to reduce the administrative burden of keeping multiple copies of information current and consistent at different sites.

7: Download SQL Server Upgrade Technical Reference Guide from Official Microsoft Download Center

This was the first book I read in preparation for the Implementing a Database Design on Microsoft SQL Server (Exam). The first thing I noticed was that a good portion of what I had been tested on for the System Administration for Microsoft SQL Server (Exam) was covered in this text.

8: SQL Server on Windows and Linux | Microsoft

8 | SnapManager for Microsoft SQL Server Installation and Administration Guide to the same federated group. Backing up a federated group backs up all databases in that group at the.

9: Microsoft SQL Server â€" Virtual DataPort Administration Guide

Finding the Right Tool for the Job. MicrosoftÂ® SQL Serverâ„¢ provides a broad set of programming interfaces that enable developers to build robust database applications with greater ease, performance, and reliability than before.

MICROSOFT SQL SERVER 7.0 ADMINISTRATORS GUIDE pdf

Laboratory chemistry for health science Convex Analysis and Variational Problems (Classics in Applied Mathematics) Horatio Alger, Jr. There She Goes Again The life and times of Cicero Us citizenship practice test 2018 Inside the forties Foundations in human development Illustrated books 2. The Brahmins Son Hermann Hesse The Athenian captive [a tragedy in five acts] Democracy and Social Ethics (Psychoanalysis for Beginners (Psychoanalysis for Beginners) Open letter to a woman of today. The concert as a literary genre: Berliozs / Open in play books Hawaiis hidden treasures Womanly dominion deceitfully assaulted Jesse Owens (On My Own Biographies Foundations of aural rehabilitation Time saver standards book Insurance operations, regulation, and statutory accounting Out of the house of life Animal origami for enthusiast Indiana Jones and the Peril at Delphi (Indiana Jones, No. 1) Health, Behaviour and Ageing (Health and behavior, a research agenda) The body of the Father Christian Rosencrux Part II: Foreskin metaphors Preserving our heritage: a story of the struggle to preserve an historic building Planning for standby Spiritual crumbs from our masters table Sovietization of Ukraine, 1917-1923 My Little Pony Color Poster Book (My Little Pony) Echoes from Niagara Real Estate Selling Magic Effect of food packaging material on the environment How To Go Into The Silence Pamphlet Edit uments trial Gerald Fitzgerald, the chevalier Introduction Jay McInerney Meghan Daum Dirk Wittenborn Glenn OBrien Emma Forrest Bruno Maddox Alain de Bo Mapping the diversity of nature