

Ensuring Data Integrity with Asynchronous Replication

A Technology Overview

By Claus Mikkelsen and Roselinda R. Schulman

July 2005

Executive Summary

Worldwide commerce and industry have become increasingly dependent on IT systems to conduct business. Even a temporary loss of critical applications can cause serious economic impact to a company, and an extended outage can threaten a company's existence. Laws and regulations increasingly focus on the creation and retention of business records in electronic form. The expectations of regulators and stakeholders are continually growing, which means there is less and less tolerance for any failure to protect important records from the recognized risks of site outage or business interruption. In sum, regulatory and competitive pressures, coupled with the potential financial impact of unavailable systems, have motivated IT executives to address the issue of real-time protection of critical data assets in the event of a disaster or an extended outage.

The result is that requirements for ultrahigh availability, 24/7 operations are the norm. All companies will be required, through regulation or best business practices, to have disaster recovery solutions in place at some point. How does a business protect its critical applications and data against a disaster or an extended outage? Although remote replication has become quite commonplace as a vehicle to provide such protection, much is still misunderstood about the technology and how it protects an organization's vital data assets.

Over the past ten years, numerous solutions have been brought to market with the promise of protecting enterprise data in the event of a disaster. With dozens of solutions on the market today, is the proper response just a matter of selecting the most economical solution? Unfortunately, it is not, since there are many underlying technical implications that must be considered. Obviously, the performance impact of the solution is important. But the most important consideration should be whether the solution maintains *data integrity* in a manner that guarantees a *successful* recovery.

Why should IT executives care about data integrity? Isn't just copying as much data as possible to the remote site sufficient? As with the question regarding selecting the most economical solution, the answer here is also "No." When a disaster or outage occurs, some number of transactions will not be completed or will be completed out of order. Either of these conditions leads to a data integrity problem, possible database corruption, and unsuccessful recovery. One might say that a seat belt in a car works most of the time, but if it fails at the moment of impact, it is of little value. Data integrity in a remote-copy solution may be considered in a similar fashion: the key question is whether it proves effective when a major failure or disaster occurs.

In fact, many industry analysts and experts agree: when it comes to business continuity and asynchronous data replication in particular, the key is not so much how far behind your data is on the replicated site. It's really about data integrity: if the data is not good, i.e. corrupted by the replication process, then why bother recovering it? The question of data integrity is therefore absolutely key to ensuring recoverability of data: without integrity, there is no recoverability.

This white paper outlines the key considerations for preserving data consistency and integrity during a worst-case disaster recovery scenario, and the essential capabilities that the data storage infrastructure must provide to ensure successful recovery with full data integrity.

Contents

Preface	1
Introduction	1
Historical Perspective	1
Focus of This White Paper	2
What is Data Consistency?	2
I/O or Crash Recovery Consistency	3
Transaction Consistency	3
Application Consistency	3
Data Loss versus Data Consistency	3
Types of Disasters and Recovery Scenarios	4
Predictive Disasters: the Planned Outage	4
Lingering Disasters	4
Rolling Disasters: Waves of Change	4
Replication Solutions for Data Recovery	5
Synchronous Replication	5
Asynchronous Replication	5
PiT Mediated Copy	6
Three Rules for Data Consistency During Replication	6
Update-sequence Validation	6
Missing-record Detection	6
Subsequent and Successful Settling (SASS)	7
Platforms for Reliable Recovery	7
Servers, Networks, and Storage	7
The Two-copy Model: Mirror, Mirror, Everywhere	8
Availability 24/7: Nondisruptive Maintenance	8
Destructive Resynchronization Revisited: New Approaches	8
Application Recovery and Data Structures: Consistency in the Real World	9
What is Application Recovery?	9
Database Technology	9
The DBMS Log	9
Conclusions	10

Ensuring Data Integrity with Asynchronous Replication

A Technology Overview

By Claus Mikkelsen and Roselinda R. Schulman

Preface

Over the years, there have been many white papers, articles, and books written on Business Continuity, Disaster Recovery, and Data Center Resiliency, including product-specific, technology-specific, and industry-specific information. In fact, if all of the literature on these related topics were stacked, it would make for a very impressive stack indeed!

This white paper is taking a slightly different approach by covering a more technical aspect of disaster recovery solutions, including the necessary technology required to ensure data integrity in the event of a disaster. Data integrity is paramount in disaster recovery solutions and, as such, it is important to understand how various solutions work. Although the explanations within this paper are primarily targeted to storage system-based replication, the lessons can just as easily be applied to network-based or server-based techniques.

It is assumed the reader of this white paper has a basic knowledge of disaster recovery and business continuity topics.

Introduction

What does data integrity really mean? More importantly, why do we care about it? Isn't it most important to have all the data, even if we have to fix any problems with it later? Or, is some other aspect more important? These questions transcend both disaster recovery techniques and the more traditional backup technologies.

Historical Perspective

In the early nineties, faced with rapid data growth and a need to improve availability, companies started using techniques like “backup while” open to improve their application uptime. Taking backups with the systems shut down was just not practical anymore: for organizations supporting 24/7 operations, the backup window simply went to zero. Backup while open sounded like a good alternative, until people tried to use the recovered data and discovered that it was “fuzzy”: the restored data was not fully consistent. Because of the way it had been backed up, it either proved unusable or required an extensive amount of time to fix it. The difference between simple data loss and data consistency loss is analogous to the difference between losing a few entries in a phone book (data loss) and having many of the addresses and phone numbers in the phone book randomly re-assigned to different names. In the latter case, all of the data is present—but making it useful again would require many hours of work, and it might not be possible at all.

The key objective of a remote disaster-recovery solution is to produce an I/O-consistent image of all data at the secondary location, as it existed at a point in time prior to the beginning of the disaster. Such a capability makes the data more quickly useful, and avoids lengthy recovery processes.

Later in this paper we will describe three rules or required capabilities for achieving data consistency: update-sequence validation, missing-record detection, and subsequent and successful settling (SASS). While this paper is focused mainly on storage system-based capabilities for ensuring data integrity in a rolling disaster, these same principles also apply to other approaches to remote replication, such as network- or software-based data replication solutions.

When considering data replication or backup technologies, it is important to ask the vendors how their techniques address these three fundamental rules.

Focus of This White Paper

The main topics of this paper include:

- :: Background on Data Consistency
- :: Data Consistency and Data Loss
- :: Types of Disasters and Recovery Scenarios
- :: Replication Solutions for Data Recovery
- :: Three rules for Data Consistency During Replication and Recovery
- :: Platforms for Reliable Recovery
- :: Application Data Structures and Their Recovery Implications

This paper focuses on the concept of data consistency. This is an important aspect of data integrity, and it must be ensured by any effective disaster-recovery solution that protects against site outages and regional disasters.

What is Data Consistency?

When assessing the ability to recover from a failure or disruptive event, one key factor is the integrity of the data when the recovery is complete and the application is ready to resume or restart.

One concept important to understand is the “dependent write” process that occurs within applications, systems, and data structures. A dependent write is a data update that is not to be executed until a previous “write”—on which it is dependent—has been executed. The logic that determines the sequence in which systems issue “writes” is pervasive in the complex data structures comprising databases, file systems, etc. Forgetting about replication for a moment, it is this logic that preserves the integrity of the data and allows systems and applications to restart after a sudden failure.

Data consistency can have different implications at different levels within the application and data architecture, since the meaning of the data can have different logical dependencies. Some key concepts include I/O consistency, transaction consistency, and application consistency.

I/O or Crash Recovery Consistency

This is a concept that refers to data that is not necessarily transaction consistent, but is still in a restartable state. For example, if the writing of data is interrupted in the middle of a transaction, the failure to complete the activity will leave the resulting data in an I/O-consistent state as long as the sequence of dependent writes has been maintained. The data is certainly recoverable, it just means that when the application (for example a database) is restarted, the data will be rolled back or rolled forward to a transaction-consistent state.

Transaction Consistency

A transaction is a logical unit of work that may include hundreds or thousands of updates. Transaction consistency is achieved either when the application (typically database) is shut down (quiesced) or when the database or other system component rolls back or rolls forward after a restart. That is, if a sudden failure leaves a database in an I/O-consistent state, transaction consistency is achieved when that database application is restarted.

Application Consistency

An application may be made up of many different types of data, such as multiple database components as well as flat files. Application consistency is the state in which individual components have each been recovered to a transaction-consistent state and, subsequently, those components collectively are brought into some sort of synchronization based upon the application requirements. We will explore these issues in more detail near the end of this paper (see section 10).

Data Loss versus Data Consistency

How does data consistency relate to the concept of data loss? Is it better to capture as many data blocks as possible, even if the result is not consistent? Or, are you better off having consistent data, even if a few partial transactions are lost or rolled back in the process?

Data loss can be defined as data that is lost and cannot be recovered by any other means. Often, transactions or files can be restored or recreated, which, therefore, is not true data loss. Applied to an asynchronous disaster recovery solution, for example, where updates may be lost in flight, many times the transactions comprising those lost updates can be recreated or reentered.

Data loss does not imply a loss of data integrity. However, given a choice, most organizations would protect data consistency—for example, ensuring that bank balances in different accounts all reflect a consistent picture at one point in time—rather than applying related updates to some locations and not to others.

Types of Disasters and Recovery Scenarios

There are many types of disasters or events to consider, and data integrity plays a big part in this. So let us think about what risks or events we need to evaluate.

First we must consider local events or risks, such as accidental or willful deletion of data, corruption, virus infection, etc. Typically an organization recovers from such events by using a local backup. The copy you recover from may be aged—for example, a 24-hour-old backup. But the most important requirement is for the recovered data to be I/O consistent—"crash recovery" consistent, as previously discussed. There are many techniques today that allow an organization to take consistent backup copies while open. What is known as consistent split technology allows you to maintain the sequencing necessary to achieve this. When looking at backup technologies and in-system replication solutions for backup, this should be a consideration.

Next let us consider the approaches we more commonly think of when we talk about disaster recovery. These can be divided into three subcategories—predictive, rolling, and lingering disasters.

Predictive Disasters: the Planned Outage

This is typically a preventive action that an organization takes when a local or regional outage is predicted well in advance. In many ways it is similar to a planned event such as a site failover to permit a major upgrade or scheduled building maintenance. From a data integrity standpoint, there should be no issues if the applications are quiesced and systems are shut down cleanly, prior to switching over to an alternative site. An example of this could be a hurricane warning: a company heeding the warning could fail over to its disaster recovery site ahead of the event, avoiding any problems with data consistency.

Lingering Disasters

This typically refers to events like a power failure, and its lingering nature is more a procedural or decision-making issue than anything else. Consider the power failure in which the power company tells you that the power grid will be back up in four hours. Now if it takes eight hours to recover at a disaster site, a company may elect to wait it out. However if the outage then continues for another eight hours, your downtime is increased beyond what it may have been had you failed over immediately at the occurrence of the event.

Other terms have come to the forefront recently, such as the "smoking-hole disaster." However, that generally falls into the rolling-disaster category, and raises the same issues.

Rolling Disasters: Waves of Change

Rolling disasters are unplanned events that occur over a span of time, typically from minutes to hours. They are called rolling disasters because not all systems, storage, or network connections fail at precisely the same moment. During such a disaster, components will fail independently, resulting in corrupted and unusable data that often requires difficult and very lengthy recovery. Terrorist attacks such as those of September 11, earthquakes, tornadoes, and floods all represent rolling disasters.

In the simplest case—that is to say, one in which no advanced disaster recovery solution is in place—a rolling disaster can involve lengthy recovery actions, including data recovery from consistent tape backups and forward recovery based on log files. When remote replication technologies are deployed, the impact of a rolling disaster can actually be even greater. Experience shows that updates occurring within the rolling-failure window will be unpredictable and chaotic, and not all of them may be successfully written to the disaster site. Therefore, it is up to the replication technology to ensure that consistency of the data is maintained in order to produce a restartable image.

Replication Solutions for Data Recovery

Replication approaches include a number of topologies that support local and/or remote copies of critical data, as well as different modes of synchronization between the primary and secondary copies.

Synchronous replication is often used for local or in-region replication, while asynchronous replication approaches are most useful for recovery from site-wide or regional outages or for when the recovery site is located too far from the production site. Related white papers compare the benefits of local and remote mirroring topologies, so this paper will focus primarily on the data-consistency implications of the different replication modes.

Synchronous Replication

In this technology, an update to the primary data is not allowed to complete to the application until it has also been successfully secured at the secondary location. This has a performance impact, and, therefore, this technology cannot be used over long distances. It is widely but erroneously believed that with synchronous copy you are protected against a rolling disaster and that there is no data loss. However, if the update does not complete successfully at both locations, the action taken is dependent on how the environment has been set up and what controls are in place (some controls could result in additional performance impact at the production site). In fact, the very act of protecting against a rolling disaster may cause some data loss, since the image at the remote site must be frozen when the first failure or error occurs to avoid the potential of out-of-sequence writes being transmitted during the failure period.

Asynchronous Replication

This mode of replication is being used more and more these days, as out-of-region replication is recognized as a key requirement for successful disaster recovery. The major difference is that asynchronous replication technology is not limited by distance. The data updates are immediately posted as complete to the application, and they are subsequently replicated to the remote site. The very nature of this technology means that data integrity plays a critical role in its design and in its usefulness during recovery. A successful and reliable solution will adhere to the key data-consistency rules to ensure that a restartable image will be available. Otherwise, recovery time may be elongated, or, in some cases, recovery may not even be possible.

PiT Mediated Copy

With point-in-time (PiT) copy approaches, the storage infrastructure makes a full copy or snapshot of the data on a regular basis. To maintain data integrity, a consistent split or freeze process must be used to create these frozen copies of the data. When used in disaster recovery solutions, enterprises typically create multiple copies of the data. At least two copies are needed, because the process is destructive during the resynchronization cycle between two sites: the older snapshot is rewritten or updated, and then frozen again. (See additional notes on destructive resynchronization below, under "Platforms for Reliable Recovery.") Because resynchronization is destructive, the enterprise will always need to maintain a separate, static copy that it can use if needed to restart the affected applications.

Three Rules for Data Consistency During Replication

As mentioned, the most important aspect of replication is maintaining data integrity and consistency. Debates will continue over the virtues of out-of-region replication versus near-site replication, or the merits of synchronous versus asynchronous and PiT copy approaches, but no one debates the importance of data integrity.

So how is data integrity maintained during replication, and, more importantly, how is it guaranteed following a disaster? The exact technical details will vary from solution to solution, but there are three essential data consistency rules that must be followed by any successful and reliable solution: Update-sequence Validation, Missing-record Detection, and Subsequent and Successful Settling.

Update-sequence Validation

This means the solution must, in all cases, apply the updates at the secondary site in exactly the same sequence in which they occurred at the primary site. Especially in asynchronous solutions—where updates most likely are arriving out of sequence at the secondary site (due to multiple links over different topologies and distances)—great care must be taken to assemble the updates in their original order. (Consider the discussion below on DBMS technology, and imagine the impact on data integrity if log updates and database updates were applied in the incorrect sequence at the secondary site.) Furthermore, applying updates out of sequence can violate the basic tenet of any disaster recovery solution, as it cannot provide a consistent recoverable image at a point in time prior to the disaster occurring.

Missing-record Detection

This requirement addresses the issue of detecting what is not known. Consider five updates executed at the primary site. (We'll call these five updates Erik, Kristy, Kelly, Danny, and Terry.) Now suppose the secondary site receives only Erik, Kristy, Kelly, and Terry. What happened to Danny? There should be sufficient logic and intelligence at the secondary site to detect the fact that Danny was an update executed at the primary site, and was just not received at the secondary site.

Again anticipating our discussion of complex data structures, it is possible that data may have been written—but a pointer to that data (for example, in the file system) was not received at the secondary site.

Subsequent and Successful Settling (SASS)

The previous two “rules” ensure that, at any given moment, the image at the secondary site is integral and represents a valid and usable image at that specific point-in-time. But the two prior rules do not, by themselves, provide a solution that can survive a rolling disaster. The challenge here is to ensure that the recoverable image is “frozen” at a point in time prior to the beginning of the disaster. Even more than the previous two rules, this requirement is difficult to address in a real-time asynchronous implementation.

How this is accomplished may vary by vendor and solution, but the process must satisfy the following statement: No update shall be applied at the secondary site until a subsequent and successful update is received from every other volume in the same consistency group (CG).

To illustrate this point with an example, let’s assume there are three volumes (A, B, and C) involved in a single CG. Now we receive the first update to A (call this update A1), but we cannot settle this one right away, since we’ve yet to receive the subsequent and successful updates from B and C. Now the second update is received to C (call this update C2). Still, neither A1 nor C2 can settle since we have yet to receive an update from B. When a third update is received, also to A (A3), still nothing can be applied. However, if the next update is to B (call it B4), now update A1 can actually settle to the secondary volume since it is the only update that satisfies the “subsequent and successful” requirement.

Why the convoluted logic? Well, let’s use this same example but without the SASS logic. A storage system at the secondary site receives A1, A3, and B4 (and does not receive C2) and decides to apply them. But let’s say in this example C2 never arrives because it resided on a storage system that had failed during a disaster. Had we continued to settle the updates to A and B, we would have been potentially applying these updates at a point in time after a disaster had in fact occurred, a process that would certainly have corrupted the data.

These three rules are basic to the maintaining data integrity. They do not necessarily make much difference during normal operations, any more than a seat belt improves normal driving success, but they are crucial to an organization’s ability to recover its data and applications during a rolling disaster.

Platforms for Reliable Recovery

Sometimes when we focus on data integrity, we forget about important considerations in platform selection and configuration.

Servers, Networks, and Storage

After investing in the disaster recovery component of an overall business continuity solution, it would make sense that the platform executing the replication has the resiliency to maintain a persistent session. The purpose of disaster recovery is to ensure your critical data is always available and never exposed to corruption, even during and following a rolling disaster. To achieve this goal, we should conclude that all components of the solution must be of equal resiliency to avoid the “weak link” problem. This means not only that the replication component must be capable of preserving data integrity, but also that the servers, networks, and storage systems must support high resiliency for critical applications.

The Two-copy Model: Mirror, Mirror, Everywhere

Enterprise storage systems generally (not always) follow the “two-copy” model (an engineering concept), which is defined as always maintaining two copies of all data in nonvolatile form as soon as data is written to the storage system. For example, RAID protection for the data as it resides on disk follows the two-copy model since it can survive a single failure and keep data preserved. Similarly, cache should always be mirrored and data updates should always be written to separate physical cache cards (on separate power supplies) so that if one cache card fails before the data is destaged to disk, the data is preserved. Also, cache should be nonvolatile or backed by sufficient battery in case power is suspended.

Availability 24/7: Nondisruptive Maintenance

Not all enterprises doing data replication are currently operating around the clock, but those that do need the assurance that all the components of their solution are also available around the clock. Operations should not have to cease because of any configuration change, microcode upgrade, or repairs to the storage system or servers. Similarly, a component failure should not have serious impact to running applications and the storage system should contain redundant components throughout. The goal should be continuous operations.

Destructive Resynchronization Revisited: New Approaches

In the majority of replication solutions, when following a remote link outage or some other event that causes the replication to cease, the replication solution continues by logging which tracks have changed. That is, if for any reason the solution is unable to transmit changed data, it simply marks (generally in a bit map) which tracks are being changed. This allows for subsequent catch-up, once the links become available or the problem is resolved. The replication solution does not have to copy all of the data again—it just copies the changed tracks to the remote site. During this resynchronization, the integrity of the secondary copy is compromised, as none of the rules of data integrity are maintained.

To mitigate this risk, a PiT copy from the secondary volume should be snapped before the “destructive resynchronization” process begins. This approach will ensure that a consistent copy is available if a disaster should occur while the resynchronization is in progress or in case the resynchronization process fails. However, the “known good copy” may significantly lag in data currency, depending on how long the event lasted and the locality of reference of the data that was changed during this period. These factors impact the time required to copy all the changed tracks, to produce a secondary copy that is both current and consistent.

Newer technologies that deploy advanced buffering or journaling techniques can avoid these problems by maintaining the rules of data integrity during the event period. Such approaches also maintain integrity and consistency during the catch-up period, while improving the currency of the secondary copy with every record being sent during this time.

Application Recovery and Data Structures: Consistency in the Real World

What is Application Recovery?

An application may comprise many different data structures from a technical perspective. It may include some database components, including pieces in multiple database formats such as DB2, Oracle, and SQL as well as flat files. Such situations can make for complex recovery scenarios. From a data consistency point of view, it is widely thought that all components need to be at the same time point in order to be able to recover. That is not necessarily true. Think of it this way: under normal operating conditions certain components may fail (for example, DB2 may fail while IMS stays up in a mainframe environment). How that is handled is based on the way the application is written, and is an application recoverability issue. The same holds true in a disaster recovery situation: even if multiple components are not at exactly the same point in time, it may not be an issue—as long as those individual components are I/O consistent or crash recovery consistent. Each component (for example, Oracle or SQL) will roll back or forward based on its own data. As long as sequencing of the data within that database is maintained, application recovery is enabled. Recovery would be handled as if a component had failed during normal operation.

In general, storage system–based replication simulates a sudden outage (such as a sudden power failure) at the primary site. The difference is that, in the disaster recovery scenario, recovery is performed at the secondary site.

Database Technology

Recovery approaches and replication solutions generally leverage the technology provided by the leading database management system (DBMS) products to enable restart from a sudden outage. Forgetting about remote copy for a moment, if an organization experiences a sudden power failure in the data center, it can generally recover DBMS operations fairly quickly once power is restored.

DBMS solutions provide features that enable faster restart after a sudden outage. Some of the most important metrics sought by DBMS vendors include performance and throughput, in terms of transactions per second. Although these motivations will result in differing designs from product to product, the underlying architectures remain quite similar.

The “magic” in database products is in the relationship between the two primary data structure components: the actual database, and the DBMS logs.

The DBMS Log

A DBMS log is a file that contains the contents of every transaction as well as the status of each transaction. For example, if you make a withdrawal at an ATM machine, the log entry for that transaction might include your account number (from the debit card), the amount of the transaction, the date and time, the location of the ATM, and much more. All of this information will be written as a log entry before the database is actually updated. Once the log is successfully updated (and *only* after the log is updated!) the database tables are modified to reflect the transaction. Once the database updates complete, the log is updated once again to indicate that the previous transaction is “complete.”

To summarize this logic, the following three phases are executed in sequence, with no step starting until the previous step completes successfully:

1. The log is updated with all of the contents of the transaction.
2. Once the log is successfully updated, the database is modified to reflect the transaction.
3. Once the modifications to the database have completed, the log is updated once again to indicate the transaction has completed.

Note: The previous is a very high-level description of how DBMS products logically are structured. Actual implementation varies dramatically of course, but the underlying concept remains quite similar.

It is important to note that the transaction is not considered complete until step 3 above has successfully completed. This three-step sequence is critical to maintaining the integrity of the database.

There is another relationship between these two data structures (again, allowing for differences among the actual products): that is, their synchronicity to the actual storage system. Updates to the log are generally “synchronous” to the storage system. This means that when the DBMS makes an update to the log, nothing else happens with regard to that transaction until the update is secured in the storage system. Conversely, the updates to the database are not written directly to the storage system but buffered instead in processor memory for performance reasons. If a random disaster occurs, these database “updates” are lost since this memory is volatile. However, upon recovery, since the log writes are synchronous, the transactions can be recovered by replaying their corresponding entries from the log.

Conclusions

Enterprise data managers and IT professionals need to take a close look at data protection in a variety of failure and recovery scenarios, and put measures in place that will enable recovery of critical data assets and applications in any remotely likely scenario. The expectations of regulators, customers, and investors will increasingly demand that enterprises protect critical applications and data sets with remote replication and recovery solutions.

When considering alternative replication and recovery solutions, enterprises must understand the recovery requirements of their applications and data sets. And they must ensure that those solutions adhere to the three rules for ensuring data consistency in any remote replication and recovery scenario—up to and including a rolling disaster with failover to a remote site.

To be sure, there are many other practical, procedural, and human issues that a disaster recovery plan must address. But from a data-protection point of view, the most important goal is to ensure the integrity and consistency of your critical data.

 **Hitachi Data Systems Corporation****Corporate Headquarters**

750 Central Expressway
Santa Clara, California 95050-2627
U.S.A.
Phone: 1 408 970 1000
www.hds.com
info@hds.com

Asia Pacific and Americas

750 Central Expressway
Santa Clara, California 95050-2627
U.S.A.
Phone: 1 408 970 1000
info@hds.com

Europe Headquarters

Sefton Park
Stoke Poges
Buckinghamshire SL2 4HD
United Kingdom
Phone: + 44 (0)1753 618000
info.eu@hds.com

Hitachi Data Systems is registered with the U.S. Patent and Trademark Office as a trademark and service mark of Hitachi, Ltd. The Hitachi Data Systems logotype is a trademark and service mark of Hitachi, Ltd.

All other company names are, or may be, trademarks or service marks of their respective owners.

Notice: This document is for informational purposes only, and does not set forth any warranty, express or implied, concerning any equipment or service offered or to be offered by Hitachi Data Systems. This document describes some capabilities that are conditioned on a maintenance contract with Hitachi Data Systems being in effect, and that may be configuration-dependent, and features that may not be currently available. Contact your local Hitachi Data Systems sales office for information on feature and product availability.

Hitachi Data Systems sells and licenses its products subject to certain terms and conditions, including limited warranties. To see a copy of these terms and conditions prior to purchase or license, please go to http://www.hds.com/products_services/support/warranty.html or call your local sales representative to obtain a printed copy. If you purchase or license the product, you are deemed to have accepted these terms and conditions.

©2005, Hitachi Data Systems Corporation. All Rights Reserved.

WHP-200-00 July 2005