HPE Shadowbase Data Recovery Software — UNDO and REDO

A Gravic, Inc. White Paper

Executive Summary

HPE Shadowbase software provides real-time data replication to target databases and to application environments throughout the enterprise. Uses for HPE Shadowbase products include achieving high or continuous availability by synchronizing the databases of active or passive redundant systems, integrating applications, feeding data warehouses and real-time business intelligence facilities, and driving Extract, Transform, and Load (ETL) solutions.

In today's online economy, a company's data is mostly stored in databases in its datacenters. The information stored in these databases is fundamental to the operation of the applications that drive business decisions and the services that the company provides to its stakeholders – employees, customers, vendors, regulators, and shareholders.

Erroneous information in a company's databases can wreak havoc on the recipients of application services, and to the operations of the company (significant costs may be incurred, audit compliance may be compromised, embarrassment to the organization, etc.). Hence, if a database becomes corrupted, then it is important to be able to restore it quickly and completely to a consistent (and correct) state to minimize the impact of erroneous data and to restore correct business operations. This capability is provided by the ***HPE Shadowbase Data Recovery Software***, ***HPE Shadowbase UNDO and REDO***.

Shadowbase REDO maintains a REDO Queue of all changes that were made to a database for "roll-forwards." Before execution, a copy of the database is created and saved if a new version of an application will be run, a new database version will be deployed, or some other system change will be made. If problems occur after executing the system changes, such as database corruption, the saved database copy can be quickly restored and the database changes selectively re-applied using Shadowbase REDO. Shadowbase REDO selectively applies only the valid database changes from its REDO Queue to the saved copy in order to roll it forward to a known-working state. A major benefit of Shadowbase REDO is that if a large volume of database corruption has occurred, it can quickly recover it to a known-working state.

Shadowbase UNDO maintains an UNDO Queue of all changes that were made to a database for "rollbacks." After a corruption occurs, Shadowbase UNDO starts with the current online database, and undoes the corrupting database changes while filtering and retaining the correct changes. The database is thus selectively restored ("rolled back") to a known, consistent, and current state. A major benefit of Shadowbase UNDO is that rollback of corrupted data can be accomplished while the application continues its processing functions and the database remains online (i.e., no application service outage occurs).

With HPE Shadowbase Data Recovery Software, safely restore corrupted databases with minimal downtime and maximum confidence, eliminating corrupted data and its associated costs and risks.

Table of Contents

Table of Figures

HPE Shadowbase Data Recovery Software – UNDO and REDO

Data is the new currency. Statistics show that 93% of companies that lost their data result in filing for bankruptcy within a year. In today's online economy, a company's data is mostly stored in databases in datacenters. The information stored in these databases is fundamental to applications that drive business decisions that management makes and the services that the company provides to its stakeholders – its employees, its customers, its vendors, its regulators, and its shareholders.

Erroneous information in a company's databases can wreak havoc on the recipients of application services, and to the operations of the company (significant costs may be incurred, audit compliance may be compromised, corporate reputation can be damaged, etc.). Credit cards cancelled in error prevent cardholders from making purchases. Incorrect banking account balances can cause unintended overdrafts or credit denials. Incorrect stock trades may lose thousands or even millions of dollars. Erroneous medical records can cause improper treatment of patients and, in extreme cases, even kill patients. Degradation of services range from severe disruptions (such as these), to the mundane, such as providing theatergoers with the wrong movie times.

Resolving Database Corruption with HPE Shadowbase Data Recovery Software

Errors can be introduced into a database by a faulty application, user error, errant system administrator commands, employee malfeasance, or some other entity. It is imperative that such erroneous changes be quickly and efficiently removed from a corrupted database so that it can be restored to a known-working state, thereby enabling business operations to access accurate information. This is the role of the *HPE Shadowbase Data Recovery Software*.

As shown in Figure 1a, Shadowbase REDO selectively *rolls forward* correct changes, as it brings a previous database version up-to-date, which eliminates the effects of corrupting changes.

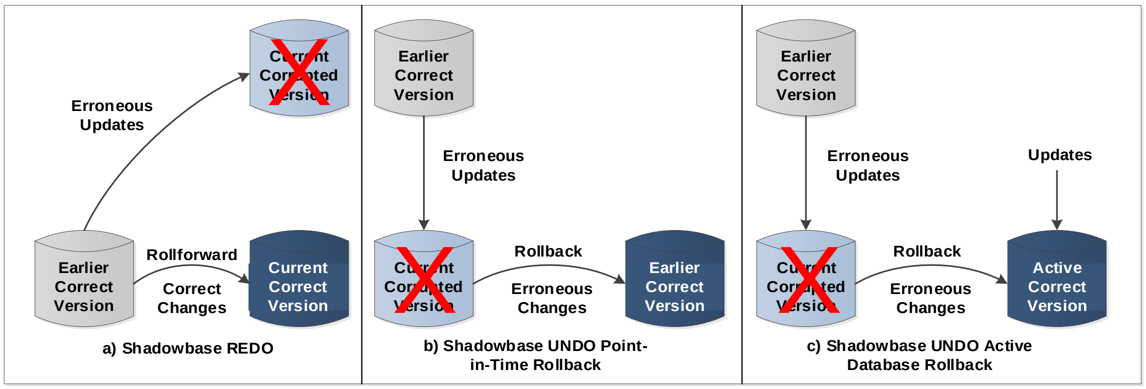

Figure 1 – Repairing Database Corruption with HPE Shadowbase UNDO and REDO

Shadowbase UNDO *rolls back* all incorrect changes in a corrupted database, in order to recreate a correct version of it at some previous point-in-time, known as a "Point-in-Time Rollback" (Figure 1b). Alternatively, Shadowbase UNDO can rollback all incorrect changes to restore the database to its current and correct state while the database remains online and continues to be updated by applications, known as "Active Database Rollback" (Figure 1c). A description follows of the architecture and use of Shadowbase UNDO and REDO.

The HPE Shadowbase Data Replication Engine

Both Shadowbase REDO and Shadowbase UNDO rely upon the Shadowbase data replication engine as their underlying foundation. The purpose of the HPE Shadowbase data replication engine is to keep a source and target database synchronized. Replication sends source database updates as they occur in real-time to the target system and applies them immediately to the target database. Shadowbase solutions are heterogeneous and support a broad range of source and target platforms, operating systems, and databases.

Basic HPE Shadowbase Data Replication (One-way/Uni-directional)

The basic Shadowbase data replication engine is shown in Figure 2. It comprises a Collector that runs on the source system and a Consumer that runs on the target system. The Collector follows changes being made to the source database via a type of change log that records every source database change as it is made. The change log is often the transaction log maintained by a transaction manager for purposes of reconstructing a database if it becomes corrupted or lost. Examples of transaction logs are the TMF Audit Trail in HPE NonStop systems and the Redo log in Oracle databases.

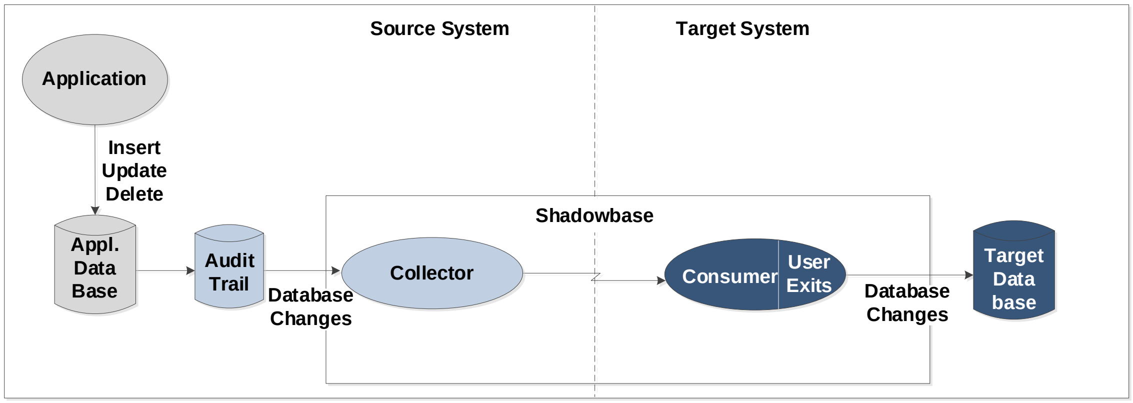

Figure 2 – Basic HPE Shadowbase Data Replication Engine

If a transaction log is not available, the Shadowbase Collector can be driven by an application-created change log or by database triggers. Regardless, as soon as the Collector reads a source database change, it sends the change through a communication channel (TCP/IP or Expand) to the Consumer on the target system. The Consumer then applies the change to the target database. The Consumer and Collector support SBMAP, related scripting, and embedded User Exits which allow database changes to be filtered, reformatted, and enhanced as necessary.

With Shadowbase basic replication, an update does not complete until the Consumer has applied it to the target database. Applying random updates to a database is a lengthy process. To achieve low latency and high data replication throughput, Shadowbase replication can be multithreaded by configuring one or more Consumers, which can be fed data by one or more Collectors. Further, an interposed Shadowbase *Queue Manager* (QMGR) can be used.

Using the HPE Shadowbase Queue Manager (QMGR) within the Data Replication Engine

The QMGR is an optional configuration of Shadowbase replication and can dramatically improve the delivery of data to the target environment. Rather than delivering database updates directly to the Consumer for writing into the target database, they are instead delivered to the QMGR resident on the target system, as shown in Figure 3.

By inserting the QMGR along the replication path, delivering data to the target system happens independently from the target database replaying that data. The intermediate queue accepts the data changes and holds them until the Consumer can apply them to the target database. It is much faster to write update events into a sequential queue file, than it is to do random writes into the target database; therefore, the use of the intermediate queue file dramatically improves data replication throughput and further reduces latency.

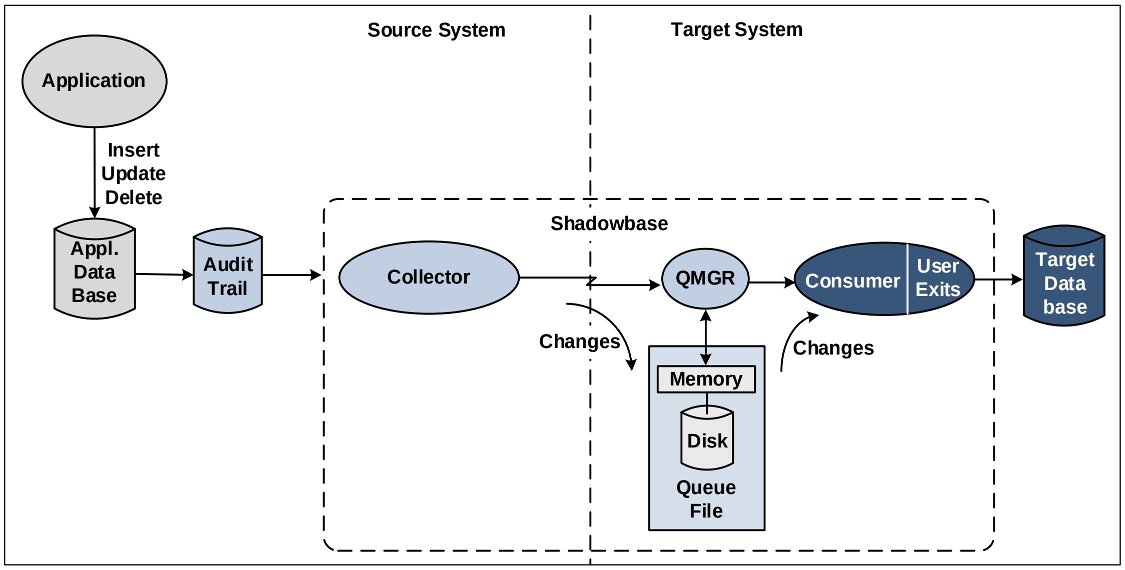

Figure 3 – HPE Shadowbase Data Replication Engine with Queuing

The Shadowbase queuing mechanism works as shown in Figure 3: the application makes changes to its database via INSERTs, UPDATEs, and DELETEs. These changes are also recorded in the application's change log (for instance, the HPE NonStop TMF Audit Trail).

The Shadowbase Collector on the source system reads the Audit Trail and sends changes to the QMGR on the target system. The QMGR writes the changes into a Queue File on the target system. The Queue File feeds a memory buffer into which replicated changes are stored while they await processing by the Consumer.

If the Consumer(s) falls behind the replication stream, rather than delaying replication and increasing Recovery Point Objective (RPO, or the window of lost data from a source system failure), the changes are held in the Queue File until the Consumers have time to read and apply them to the target database. Thus, replication is only delayed by the *replication latency*, or the time it takes to sequentially write changes to the Queue File, rather than the required time of randomly applying changes to the target database. The result is less data in the replication latency window, and a lower RPO, meaning more data is saved after a failure occurs because the amount of database updates that could be lost is greatly reduced. The QMGR has a number of significant benefits, including how it enables Shadowbase UNDO and REDO.

HPE Shadowbase REDO

System Upgrades Can Cause Database Corruption

It is often necessary to perform system upgrades, which may include application, system, platform, or even database changes. When these upgrades need to be made, there is always the concern that problems will be encountered after the upgrade occurs and that the system will have to be returned to its initial state (often referred to as "*failback"* or "*fallback"*).

One such potential problem is database corruption. A remedy is to failback to a known-working copy of the database, and then filter transactions so that only valid ones are reapplied, thereby bringing the database into a current and correct state before resuming the application. This remedy is called "*rolling forward"* and is the basis for the REDO approach.

Current Techniques for Roll-Forward

Backups

Several approaches exist to restore a corrupted database by rolling it forward. One is making a copy of the database on magnetic tape or disk prior to initiating the upgraded system to process. If problems occur, the database is loaded with the database copy to restore it to its prior state, and a change log is used to roll forward the changes that occurred after the upgraded system began processing.

This technique presents some significant challenges:

1. One challenge is the data volume that needs to be processed to perform the restore operation.
    a. Typically, the entire database (or perhaps just a set of files or tables) has to be restored to the prior backup point even if only a small amount of the database was subsequently corrupted.
    b. This restoration process can involve the recovery of massive amounts of data from the backup medium and can require access to hundreds or thousands of tapes or disk packs.

2. Another challenge is the time it takes to copy and restore a large database from magnetic tape or disk.
    a. Modern databases are massive, often measuring several terabytes (trillions of bytes) in size.
    b. Even backing up and restoring from high-speed disk, rather than from magnetic tape, can take hours if not days. During this recovery time, the application is typically down and normal business operations *cannot proceed* if the data required to perform those services is even temporarily corrupted.

3. Another problem is that the roll-forward capability provided by many transaction managers does not allow selective roll-forwards.
    a. All of the changes that are in the change log are applied to the database to bring it to a current state. This capability can result in the transaction manager reapplying the corrupting changes, thereby corrupting the recently restored database!
    b. It is also possible that the roll-forward will be unsuccessful, as some of the database changes made during the subsequent period may have depended upon a new data schema and hence cannot be applied to the database copy. If this is the case, the roll-forward operation is aborted by the transaction manager, and cannot be completed.

Split Mirrors

The time taken to restore the database copy that is needed for the recovery operation can be improved via another recovery method. In fault-tolerant systems such as HPE NonStop servers, disks are mirrored. Every logical disk actually is comprised of two physical disks, and all updates are simultaneously applied to both disks (the “mirrored” pair). In this instance, prior to starting the upgraded system, the mirrors are split.

One mirror is used by the upgraded application, and the other is saved as the pre-upgrade copy. Splitting a mirror and attaching the application to one of the mirrors is a relatively fast operation. If desired, a second mirror can be attached and revived to the new database so that the application is once again running against a mirrored database during the upgrade trial. This method helps protect the upgraded system from subsequent media failure that may cause a system outage.

If problems are encountered after running the upgraded system, failback to the original system prior to the upgrade involves reverting to the original system infrastructure and mounting the saved mirror of the database. Thus, the required time is very small to *back up* the initial version of the database and then *restore it* in the event of a problem. The database can be re-mirrored (revived) while the application is running so that the database copy and restoration activity do not significantly impact application availability.

Since the new database schema may be vastly different from the restored database schema, the transaction manager typically could not selectively roll forward the change queue updates. Thus, the application database cannot be rolled forward to its current, correct state using this method, resulting in a nasty and ugly outage. All of these issues are avoidable and resolved with Shadowbase REDO.

Use HPE Shadowbase REDO to Roll-Forward a Corrupted Database

The Shadowbase REDO engine is shown in Figure 4. It is a similar architecture to Figure 3; however, note that the Queue File is used as the change data source, and is shown as the "REDO Queue."

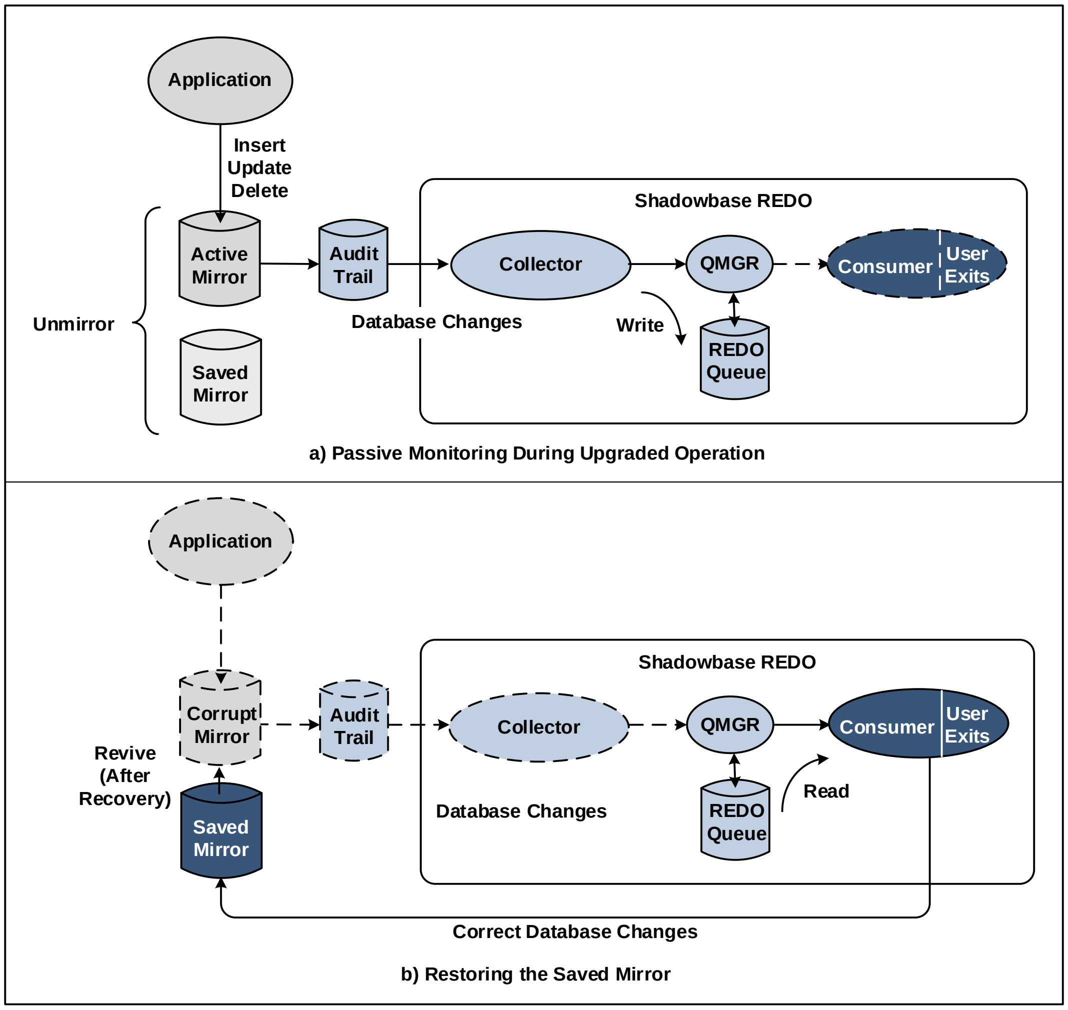

Figure 4 – HPE Shadowbase REDO Roll-Forward

1. When performing an upgrade, make a copy of the application database pre-upgrade, for example, by un-mirroring and saving the application database, as discussed above and shown in Figure 4a. Save the pre-upgrade mirror for database recovery if a problem is encountered during the upgrade.
    a. The other mirror (the "active" mirror) serves as the active database for the upgraded application.
    b. A new mirror can be added to re-mirror the active database to provide disk media fault tolerance.
2. Second, perform the upgrade. The active mirror now reflects the upgraded database. Start the newly upgraded environment. Shadowbase REDO is configured and running when the upgraded system starts processing.
    a. As the upgraded system runs, updates made to the application database (meaning the active mirror) are saved in the Audit Trail.
    b. The Shadowbase REDO Collector reads the Audit Trail and captures the REDO data by sending each change to the QMGR, which writes the change into its REDO Queue.
    c. If the user determines that the updated system is successfully and correctly running and no recovery operation will be needed, then the activities of Shadowbase REDO can be terminated and the contents of its REDO Queue deleted.
    d. However, if a problem should arise, simply roll back the database by mounting the saved mirror copy, as shown in Figure 4b. In this case, update activity for the application is stopped, though reads of the database may still be allowed.

The QMGR in Shadowbase REDO reads the changes from its REDO Queue in first-in, first-out (FIFO) order. Any invalid or undesirable changes, such as to nonexistent tables, can be filtered out by the Consumer, and valid changes are written to the saved mirror to bring it into a current (excluding the corrupted changes) and correct state.

For complex changes, Shadowbase replication supports adding additional business logic (called User Exits) into the Consumer processing logic path to direct the Shadowbase engine on the actions to take during recovery. If desired, Shadowbase REDO can set aside invalid changes for later processing, such as adding them to the active database via a SQL JOIN operation after the appropriate schema changes are made.

Upon emptying the REDO Queue and rolling back other necessary changes (for example, resetting a bad application), normal processing can resume from the point of the last-known valid transaction. At this time, the corrupted mirror can be revived by copying the now-current database from the saved mirror to the corrupted mirror. The application can typically remain online during the database restoration process.

Thus, Shadowbase REDO solves many of the problems of existing methods for correcting a corrupted database using roll-forward:

- A backup copy of the database can be rapidly created by un-mirroring a mirrored pair.
- The backup copy can be restored rapidly by using the mirrored copy.
- Most importantly, data updates are filtered to post only valid changes to the restored database copy.
- The roll-forward capabilities of Shadowbase REDO do not depend upon a copy of the transaction log, which may be lost on the abandoned, old active copy.
- Shadowbase REDO will remain online even if the database schema does not support an update. The update is simply ignored, can be configured to be saved to a log file, and can be processed later through other means.

HPE Shadowbase REDO Roll-Forward of a Corrupted Target Database

The use of Shadowbase REDO to restore a corrupted application database, as described above, can be extended to the restoration of a corrupted target database in a replication environment. In this instance, the application is typically running on a source system and is updating a source database. Changes made to the source database are replicated in real-time by Shadowbase replication to a target database running on a target system, as shown in Figure 4a and b. All changes are replicated, including corrupt changes. Therefore, if the source database becomes corrupted, so does the target.

In order to restore a corrupted target database with Shadowbase REDO, a target database copy must have been saved prior to the corruption, usually by splitting the mirrored database for source system REDO. To roll forward the target database, the saved target mirror is made active. Then, rolling forward changes on the source database will cause those changes to be replicated to the target database, thus restoring it to the same consistent state as the source database while the REDO process progresses.

Note that the REDO approach is a procedural approach, restoring a corrupted database to a known-working, initial state, then, applying selected transactions against it to “roll it forward” and preserve the newly created data. As discussed above, it requires an outage of the application/database environment while the REDO operation occurs, unlike Shadowbase UNDO.

HPE Shadowbase UNDO

Shadowbase REDO starts from the original, mirrored database copy. Then, it replays or rolls forward valid updates that occurred after the original copy was created and applies them in order. Shadowbase UNDO starts with the current online database, and undoes the corrupt database changes by rolling them back without affecting valid data; it restores the database to a known, consistent, and current state.

A unique feature of Shadowbase UNDO is that online business services can continue to actively perform their processing functions and update the database while the UNDO process is in progress. Corrupted data objects will be returned to their correct state prior to the corruption. However, if a data object was properly updated following its corruption, the new correct state can be retained.

Shadowbase UNDO runs the Shadowbase replication engine with queuing on the system whose database needs to be protected or restored, as shown in Figure 5. This configuration is essentially the same one as shown in Figure 4 except that the QMGR, now known as the *UNDO Queue*, is modified as described below to support the UNDO function.

Use HPE Shadowbase UNDO to Rollback a Corrupted Database

Shadowbase UNDO can be installed and permanently added to a system, or can be installed as needed to repair a corrupted database, and then uninstalled. In Figure 5, Shadowbase UNDO is installed and permanently configured.

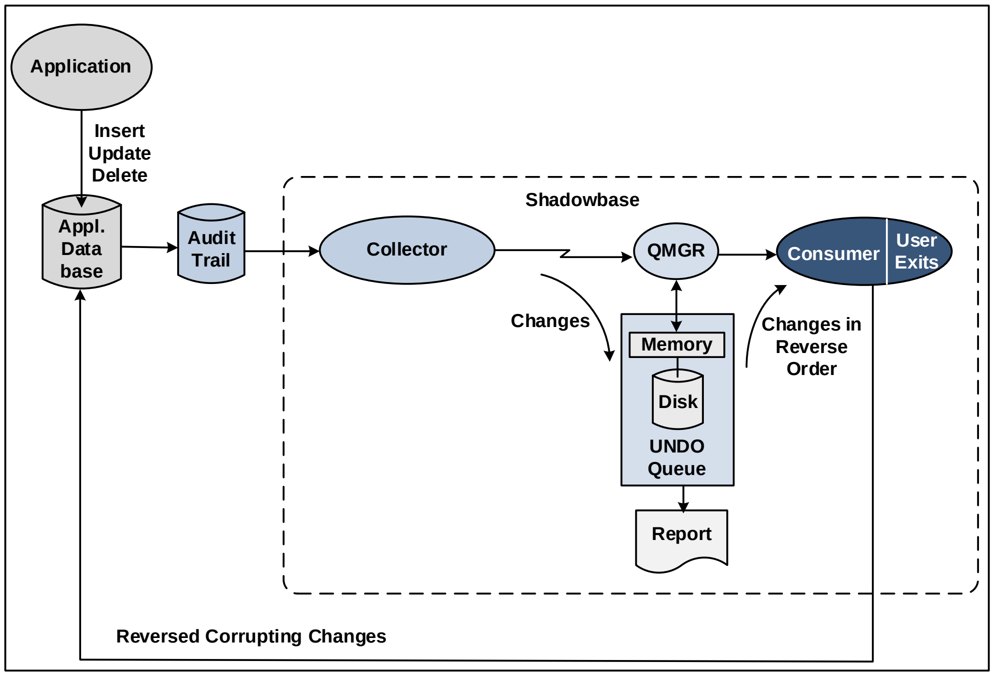

Figure 5 – HPE Shadowbase UNDO Rollback

The Shadowbase Collector process monitors Audit Trail events. It sends the changes to the QMGR, which writes them into its UNDO Queue. Changes are queued in the UNDO Queue, and Shadowbase UNDO sits until it is invoked to correct a corrupted database.

However, the corrupting event may have occurred before Shadowbase UNDO was installed and configured. For this situation, the Shadowbase Collector process will read the change data by referencing a specified time period prior to the corruption in the Audit Trail until some point after the corruption. It will then send these database changes to the QMGR, which will write the changes to the UNDO Queue.

Whether Shadowbase UNDO was installed before or after the corrupting events, it can prepare a report based on the data in its UNDO Queue. The report simplifies investigation into root cause analysis, when the corruption began and ended, and what parts of the database (such as tables, files, and record rows) are affected.

Armed with this information, the system administrator can run Shadowbase UNDO by specifying the begin and end times of the corruption, the beginning and ending transactions, or even change log information such as event time, file name, table name, and more information to determine the beginning and ending positions when the corruption occurred. System administrators can leverage Shadowbase UNDO to greatly simplify and expedite restoring corrupted databases.

Once given the required parameters (“restore specifications”), Shadowbase UNDO begins its recovery processing. The QMGR reads database change events from its UNDO Queue in reverse time order (that is, the most recent changes are read first). It filters out the valid events that are of no interest according to the restore specification. It also filters out change events that are within the scope of aborted transactions. As it reads each change to be rolled back, the QMGR converts the change into a reverse operation to undo its effect on the database. It sends these undo events to the Consumer, which applies them to the application database in order to return the database to its correct state before the corruption occurred.

HPE Shadowbase UNDO Reverse Operations

As the QMGR obtains the pertinent database changes that must be reversed from the UNDO Queue, it translates them into database operations that will reverse the original changes.

These undo operations are as follows:

Original Change (Original Order)	UNDO Change (Replay Order)	Net Effect
1. Begin transaction	5. Commit transaction	Commit an UNDO transaction
2. Insert Row A	4. Delete Row A	An erroneously inserted row is deleted
3. Update Row B from “x” to “y”	3. Update Row B from ‘y’ back to ‘x’	An erroneous update is reversed
4. Delete Row C	2. Insert Row C	An erroneously deleted row is inserted
5. Commit transaction	1. Begin transaction	Begin an UNDO transaction

Table 1 – Undo Changes to Corrupting Updates

The order of the UNDO changes is then:

UNDO Change (Replay Order)
1. Begin transaction
2. Insert Row C
3. Update Row B from ‘y’ back to ‘x’
4. Delete Row A
5. Commit transaction

Each event in the UNDO Queue contains the before and after images of the row that was changed. The Shadowbase UNDO operations are accomplished by using the before images:

- The before image of an INSERT is a null row (meaning the row does not exist).
- The before image of an UPDATE is the row’s value before the UPDATE.
- The before image of a DELETE is the row’s value before the DELETE.

The Shadowbase UNDO changes are composed into a transaction that is the reverse order of the corrupting transaction. Corrupted transaction’s COMMITs become BEGINs, and, likewise, BEGINs become COMMITs.

An HPE Shadowbase UNDO Example

Consider the sequence of events shown in the left half of Figure 6, which shows a corrupted sequence of operations involving Row A (the top-right of the figure) in the database. Row A was initially set to the value “w.” The first operation of the corrupted sequence deleted Row A. The next operation inserted Row A with a value of “x.” The last operation changed the contents of Row A from “x” to “y.”

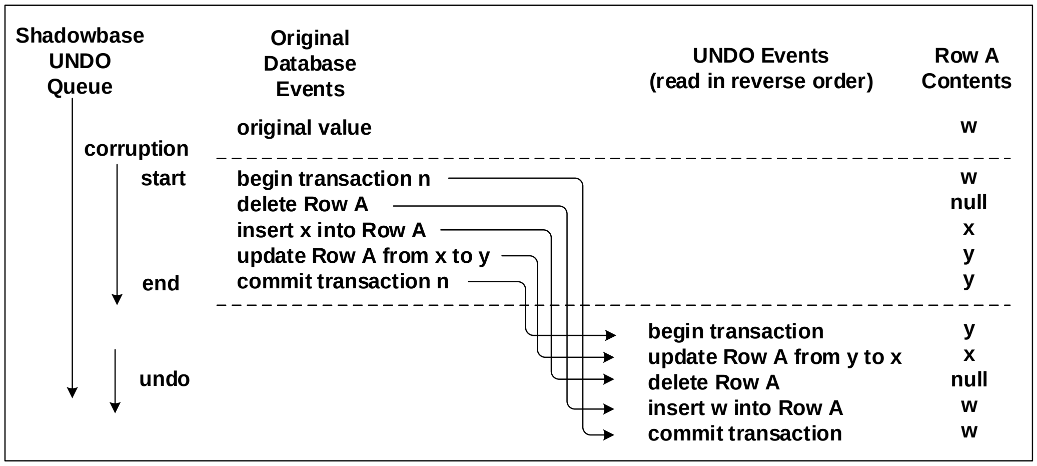

Figure 6 – A Database Restore Using HPE Shadowbase UNDO

This sequence of operations was identified as erroneous by the system administrator, who created a restore specification for Shadowbase UNDO to back out the sequence. Shadowbase UNDO reads the change operations in the corrupted sequence from the UNDO Queue in reverse order.

In this example:

- The first entry that Shadowbase UNDO reads is the last operation of the corrupted sequence, the COMMIT directive. Shadowbase UNDO converts it to a BEGIN transaction directive.
- The next entry is the “UPDATE Row A from x to y” operation. The before image of this entry is a value of “x” for Row A, so Shadowbase UNDO updates Row A to “x.”
- The next entry is the INSERT operation. Therefore, Shadowbase UNDO DELETEs Row A.
- The fourth entry is the “DELETE Row A” operation, which before image is “w,” so Shadowbase UNDO INSERTs Row A with a value of “w.”
- Finally, the BEGIN transaction event is read and is converted to a COMMIT transaction to complete the Shadowbase UNDO transaction that is applied to the application database.

The Shadowbase UNDO sequence has thus restored the database to its initial state, Row A is set to value “w” – undoing the corruption.

Shadowbase UNDO Can Restore Databases While the Application Remains Online and Operationally Active

This feature is particularly useful if the corruption is isolated to a subset of the database or, for example, a particular set of users, and it would be unacceptable to take the database offline in order to recover, because that would cause a service outage.

Preserving Subsequent Changes

Shadowbase UNDO can detect and queue change data that occurred to a corrupted data object. If a corrupted data object was subsequently modified, its value can optionally be considered as valid and left intact. This situation is useful, for example, when an UPDATE would remediate the original data corruption. Otherwise, its value is reversed to reflect its value prior to the corruption.

For instance, consider Figure 6 in an architecture where the application continues to run after the corruption. If Row A has been further modified following the corrupted sequence, the corrupted value may be overwritten by a valid value. The procedure described above will overwrite the valid value with the original value prior to the corruption. However, new change data that arrived during the corrupting time period needs to be properly accounted for. Shadowbase UNDO *additionally* provides the means to preserve the change data’s new values that have been entered into the database since the corruption occurred.

The left half of Figure 6 shows a corrupted sequence of operations involving Row A in the database. Row A was initially set to the value “w.” The first operation of the corrupted sequence deleted Row A. The next operation inserted Row A with a value of “x.” The last operation changed the contents of Row A from “x” to “y.”

Figure 6 illustrates both the problem and the solution. The example in Figure 7 is extended, and a second, valid, transaction is executed that updates Row A from its corrupted value “y” to a correct value “z.” The result of the Shadowbase UNDO operation, as described above, will result in a value for Row A of “w.” This value was correct before the corruption occurred, but the new, correct value of “z” has been lost.

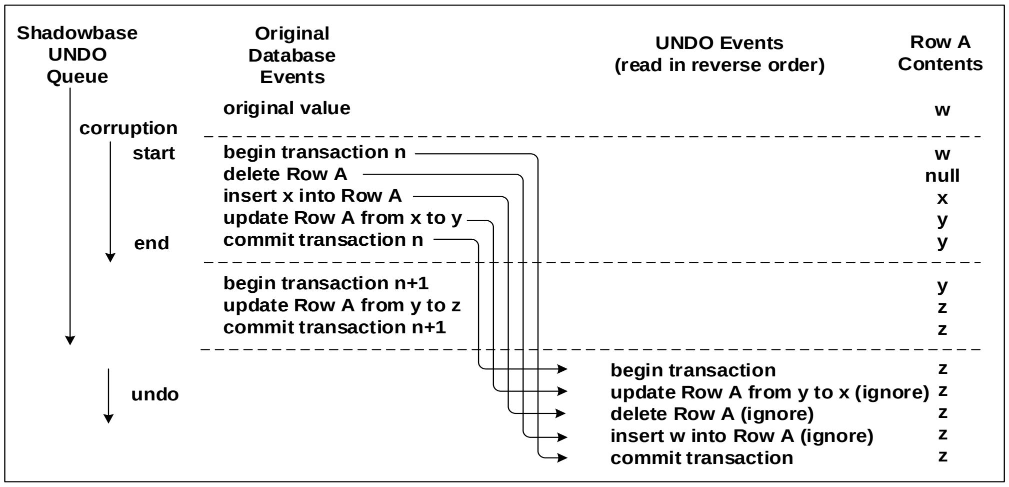

Figure 7 – HPE Shadowbase UNDO Protecting Subsequent Changes

A subsequent valid update can be detected during the Shadowbase UNDO operation by comparing the after image of the last corrupting database operation to the current value of that row in the database. If they are different, the row has been subsequently modified and should not be changed. With reference to Figure 7, the last operation in the corrupted transaction is "UPDATE Row A from x to y." Shadowbase UNDO reverses this operation with "UPDATE Row A from y to x." If there has been no subsequent operation on Row A, its database value will still be "y," which is the value of the after image of the corrupted operation. Therefore, the row has been corrupted and can be corrected. However, Row A has been subsequently modified to a valid value of "z" following the corruption. This value is different from the after image ("y") of the corrupted update. Therefore, a subsequent operation has been performed on Row A, and no Shadowbase UNDO operations should be performed if the intent is to protect subsequent operations. The result is that Row A is left with its ultimate correct value of "z" rather than its original pre-corrupted value of "w."

One issue that can occur with this algorithm is when a subsequent valid change sets the value of the data object to the same value that it had at the end of the corruption event as the now-correct value of the data object will be mistaken for the ending corrupted value. In this case, the data object will be reset to its value prior to the corruption, thus losing the valid value. This problem can be resolved by using version information in the rows. The version information can be, for instance, a timestamp or a row version number. When beginning the Shadowbase UNDO process on a data object, if the current version of the row is later than the version of the reversing operation as contained in the UNDO Queue, the row has been subsequently updated and should not be rolled back, referencing the newest version. (Note – Shadowbase UNDO also has parameters to help system administrators manage these various situations.)

Another issue with this approach can occur for *relative* changes (instead of absolute changes shown in Figure 7), for example, arithmetic operations applied to account balances. In this case, a Shadowbase User Exit can be added to the UNDO logic processing in the Consumer to track and determine the correct value to use to properly reverse the corruption during the UNDO sequence.

Undoing DDL Changes

Shadowbase UNDO can rollback all data manipulation language (DML) changes and some data description language (DDL) changes, but may not be able to rollback all DDL changes. In this instance, the operations can be reported and skipped, or a more complex User Exit can be developed to handle the processing in conjunction with Shadowbase DCR (DDL Command Replication). Alternatively, this limitation is resolved by Shadowbase REDO, as described earlier.

HPE Shadowbase UNDO to Rollback a Corrupted Target Database

Shadowbase UNDO can also restore a corrupted target database. In this case, the application is running on a source system and is updating a source database. Changes made to the source database are replicated in real-time by Shadowbase replication to a target database running on a target system, as shown in Figure 8. All changes are replicated, including corrupting changes. Therefore, if the source database becomes corrupted, that corruption is passed to the target database.

By the same token, changes applied to the source database by Shadowbase UNDO to correct the source database corruption will also be automatically replicated to the target database (by the Shadowbase Collector and Consumer processes), thus correcting the target database's corruption.

HPE Shadowbase UNDO to Simultaneously Correct both Source and Target Databases

Two instances of Shadowbase replication must be deployed to simultaneously correct both source and target databases as shown in Figure 8. One instance is the normal Shadowbase replication engine used for production to keep the target database synchronized with the source database. The other is Shadowbase UNDO running on the source system.

To correct the database corruption, the corrupted range of database updates are loaded from the Audit Trail (not pictured) into the UNDO Queue of Shadowbase UNDO by its Collector and the QMGR. Once this has been completed, the Consumer in Shadowbase UNDO reads the corrupting changes starting with the latest one, reverses their operations as shown in Figure 7, and applies them to the source database to reverse the corrupted change events.

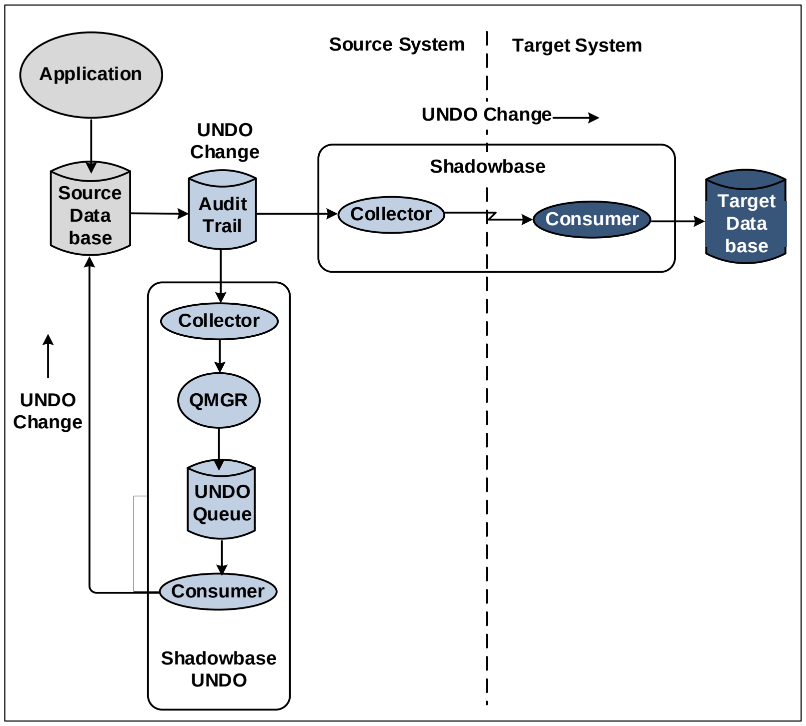

Figure 8 – Undoing Replicated Corruption

As each reverse event is posted to the source database, it is written to the Audit Trail and is automatically replicated by the Shadowbase replication engine to the target database. Consequently, the target database is subsequently corrected and synchronized with the source database. Ultimately, both databases are returned to an earlier, known-working state, (except, in this specific example, valid changes made to the source database following the corrupting events are not pictured, but are described above in "Preserving Subsequent Changes").

Summary

Database corruption can be caused by programming errors, user errors, system faults, employee malfeasance, and a number of other actions. Erroneous information in a company's databases can wreak havoc on application services, and to the operations of the company in the form of significant costs incurred, audit compliance being compromised, corporate reputation damaged, etc. Hence, if a database becomes corrupted, then it is important to quickly and completely restore it to a consistent state to minimize the corruption's impact and restore the business operation's valuable data. *HPE Shadowbase Data Recovery Software* provides *HPE Shadowbase REDO* and *HPE Shadowbase UNDO* to solve this problem.

If a new application version is run, a new database version deployed, or some other system change is made, a copy (or “mirror”) of the database may be created and saved before executing the system modification. Shadowbase REDO maintains a REDO Queue of new change data during this recovery process. If problems result in database corruption, Shadowbase REDO reads selected valid database changes from its REDO Queue and applies them to the saved copy in order to quickly roll the database forward to a known-working and correct state.

Conversely, by maintaining the UNDO Queue of changes that were made to a database, Shadowbase UNDO follows the UNDO Queue in reverse time order to the initial point of corruption and reverses any corrupting changes that were made, in order to roll the database back to a known-working and correct state. The rollback of corrupted data is accomplished by Shadowbase UNDO while the application remains online, providing business services to users. Correct updates made after the period of corruption are preserved.

Helpful reports are generated from either the UNDO Queue or REDO Queue to aid in determining the period of corruption from the contributing sources. Corrective activity is specified by a user or system administrator to include a time or transaction range, a list of affected files and tables, and a list of causes of corrupted transactions.

HPE Shadowbase Data Recovery Software can quickly and easily remediate the serious effects of corrupted data. It is flexible and able to meet specific needs, including allowing embedded logic and customized processing via powerful User Exits.

Depending on the nature of the corruption:

- *HPE Shadowbase REDO* may be used when significant data corruption has occurred, affecting a large part of the database. However, an application/database outage is required to complete the operation.
- *HPE Shadowbase UNDO* may be used if the amount of data corruption is small, localized, or service availability must be maintained while recovery is in process.

HPE Shadowbase Data Recovery Software enables companies to safely restore corrupted databases regardless of the cause, with a minimum of downtime and a maximum of confidence, eliminating the potential costs and risks associated with corrupted data.

International Partner Information

Global

Hewlett Packard Enterprise
6280 America Center Drive
San Jose, CA 95002
USA
Tel: +1.800.607.3567
www.hpe.com

Japan

High Availability Systems Co. Ltd
MS Shibaura Bldg.
4-13-23 Shibaura
Minato-ku, Tokyo 108-0023
Japan
Tel: +81 3 5730 8870
Fax: +81 3 5730 8629
www.ha-sys.co.jp

Gravic, Inc. Contact Information

17 General Warren Blvd.
Malvern, PA 19355-1245
USA
Tel: +1.610.647.6250
Fax: +1.610.647.7958
www.shadowbasesoftware.com
Email Sales: shadowbase@gravic.com
Email Support: sbsupport@gravic.com

Hewlett Packard Enterprise Business Partner Information

Hewlett Packard Enterprise directly sells and supports Shadowbase Solutions under the name ***HPE Shadowbase***. For more information, please contact your local HPE account team or visit our website.

Copyright and Trademark Information

This document is Copyright © 2014, 2017, 2024 by Gravic, Inc. Gravic, Shadowbase and Total Replication Solutions are registered trademarks of Gravic, Inc. All other brand and product names are the trademarks or registered trademarks of their respective owners. Specifications subject to change without notice.