Craig S. Mullins

Database Performance Management

Return to Home Page

February 2000

The Managers Publication of Data Solutions

DATABASE TRENDS

Minimizing Outages With Transaction Recovery

By Craig S. Mullins

Although data availability is high on the list of objectives for all DBAs, the demands of e-business exacerbate the need for high availability. If your data is not available, your applications can not run. If your applications can not run, your company is losing business. And lost business translates into lower profitability and perhaps a lower stock valuation for your company. When you business is tied to the Internet it dramatically changes the way you need to do business. User

expectations are for your businesses to be more connected, more flexible, and importantly, more available. It may be three o'clock in the morning in New York, but it is always prime time somewhere in the world. So an e-business must be available and operational 24 hours a day, 7 days a week, 365 days a year (366 for leap years, like 2000).

In past issues, I have talked about the need for DBAs to transform themselves into eDBAs to manage the data requirements of e-businesses. One of the biggest requirements of the eDBA is to enhance data availability by reducing or eliminating outages. But with the demands for higher and higher availability traditional forms of database recovery are becoming more and more inadequate.

To combat this reality the eDBA should become knowledgeable of transaction recovery techniques. Transaction recovery provides the speed and ease of a point-in-time recovery with the selective capability provided by custom programming.

Types of Recovery

There are many different types of recovery that can be performed. The first type of recovery that usually comes to mind is a recovery-to-current to handle some sort of disaster. This disaster could be anything from a simple media failure to a natural disaster destroying your data center. Applications are completely unavailable until the recovery is complete.

Another type of traditional recovery is a Point-in-Time (PIT) recovery. PIT recovery usually is performed to deal with an application level problem. Conventional techniques to perform a point in time recovery will remove the effects of *all* transactions since that specified point in time. This sometimes can cause problems if there were some valid transactions during that timeframe that still need to be applied.

Transaction Recovery is a third type of recovery that addresses the shortcomings of the traditional types of recovery: downtime and loss of good data. Thus, Transaction Recovery is an application recovery whereby the effects of specific transactions during a specified timeframe are removed from the database.

Most technicians think about recovery in order to resolve disasters such as hardware failures. But the majority of recoveries in this day-and-age are performed because of application software error and human errors. Hardware failures are just not as prevalent as they used to be. In fact, the Gartner Group estimates that as much as 80% of application errors are due to application software failures and human error.

So, if the reason you need to recover has changed, it makes sense to rethink your recovery strategies.

What is Transaction Recovery?

Transaction Recovery is the process of removing the undesired effects of specific transactions from the database. Sounds easy, but the actual details are somewhat complicated. Let’s examine transaction recovery in a bit more detail.

When performing a traditional recovery, you execute recover jobs moving object-by-object through the database. The recovery reads a backup copy of each object and applies it to the object. Then database log entries are read and applied to the object.

Contrast that approach with transaction recovery. When using transaction recovery the impact is at the transaction level, not at the database object level. The transactions are defined by the user's view of the process (for example the set of panels comprising a new hire operation or the set of jobs that post to the General Ledger).

Why is Transaction Recovery a much-needed tool in the arsenal of e-DBAs? Well, applications are prone to all types of problems, bugs, and errors. Using Transaction Recovery, the DBA can quickly react to application-level problems and maintain a higher degree of data availability. The database does not always need to be taken offline while Transaction Recovery occurs (it depends on the type of Transaction Recovery being performed).

So, how exactly is Transaction Recovery performed? Given that we have identified the transaction to recover, there are three types of Transaction Recovery that can be performed:

- Point-in-time – remove all the transactions since a given point in time and then manually rerun or reenter the work that was valid.
- UNDO – back out the bad transactions only.

- REDO – remove all the transactions after a given point in time. Then, redo the good transactions only.

Let’s examine each of these possibilities in a little more detail.

Point-in-Time (PIT) Recovery

Point-in-time recovery is the most basic strategy. It is also the only one actually supported by the native utilities of most DBMS products. With PIT recovery, you must be able to determine a common recovery point for a set of database objects.

After the PIT recovery you may need to rerun valid operations that occurred after the point-in-time to which you recovered.

UNDO Transaction Recovery

The second possibility is to deploy UNDO Transaction Recovery (refer to Figure 1). This is the simplest type of SQL-based Transaction Recovery. It involves generating UNDO SQL statements to reverse the effect of the transactions in error. To accomplish this

transformation you will need a solution that understands the database log format and can create the SQL needed to undo the data modifications. With transaction recovery, the database log is read for the transaction in question, and the effects of the transaction are reversed. In SQL parlance this means:

- INSERT statements are converted to DELETE statements
- DELETE statements are converted to INSERT statements
- UPDATE statements are converted to modify the data to its state prior to the original UPDATE

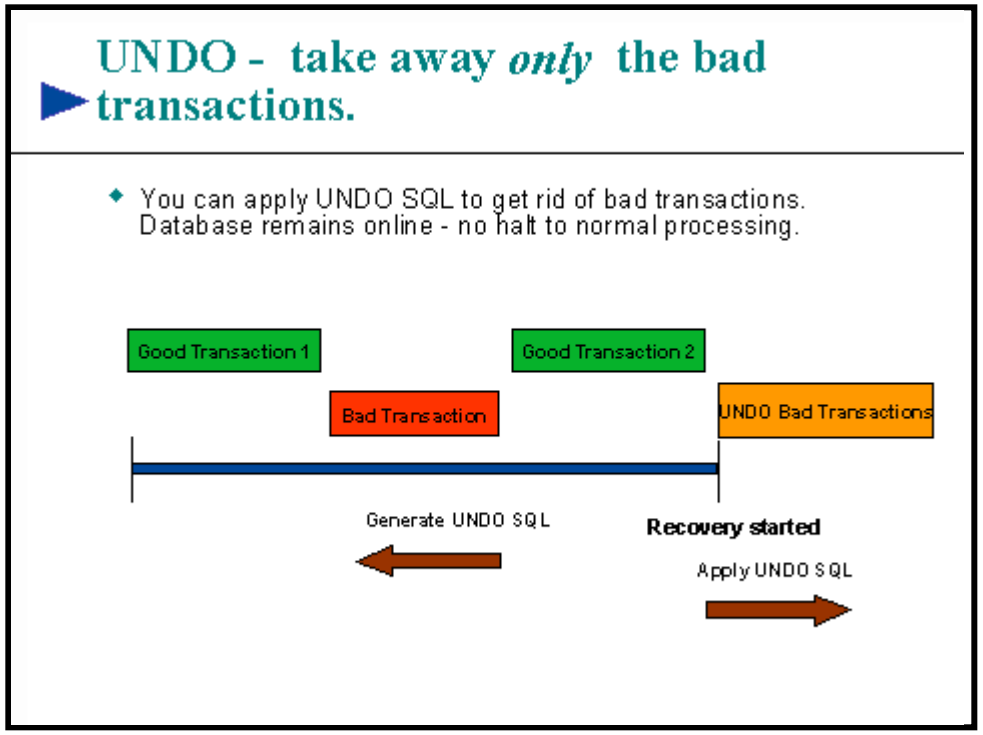

Figure 1. UNDO Transaction Recovery

In the case depicted in Figure 1, UNDO SQL statements are generated for the "bad transaction" and then applied. Note that in the case of UNDO Transaction Recovery, the portion of the database that does not need to be recovered remains undisturbed. When undoing erroneous transactions, recovery can be done online without suffering an outage of the application or the database. However, the potential for anomalies causing failures in the UNDO is certainly a consideration. You must

understand the data and the application processes that access and modify the data undergoing the transaction recovery. You will need to analyze the effects of subsequent modifications that were perhaps based on the state of the data introduced by the "bad transaction."

REDO Transaction Recovery

The REDO Transaction Recovery strategy is a combination of the first two recovery techniques we have discussed—but with a twist (refer to Figure 2).

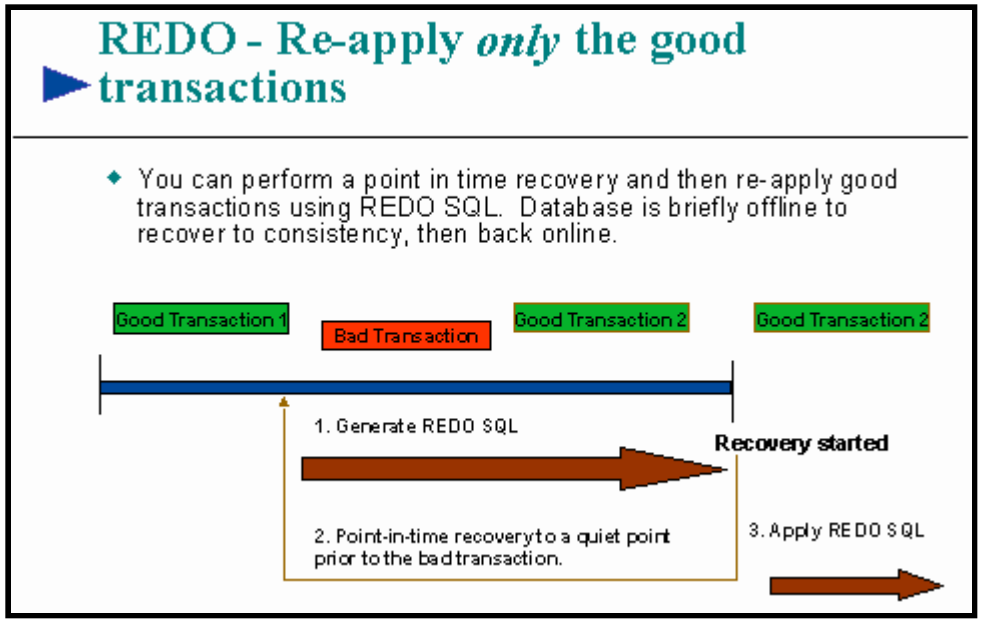

Figure 2. REDO Transaction Recovery

Instead of generating SQL for the bad transaction that we want to eliminate, we generate the SQL for the transactions we want to save. Then we do a standard point in time recovery eliminating all the transactions since the recovery point. Finally we re-apply the good transactions captured in the first step.

Unlike the UNDO process which creates SQL statements that are designed to back out all of the problem transactions, the REDO process creates SQL statements that are designed to reapply only the valid transactions from a consistent point of recovery to the current time. Since the REDO process does not generate SQL for the problem transactions, performing a recovery and then executing the REDO SQL can restore the object to a current state that does not include the problem transactions.

To generate the redo SQL statements, you will once again need a solution that understands the database log format and can create the SQL needed to redo the data modifications.

When redoing transactions in an environment where availability is crucial, a PIT recovery can be done and then the application and database

can brought online. The subsequent redoing of the valid transactions to complete the recovery then could be done with the data online, thereby reducing application downtime.

Choosing the Optimum Recovery Strategy

So, what is the best recovery strategy? Of course, as with most database-related questions, the answer is: it depends. While transaction recovery may seem like the answer to all your application recovery problems, there are a number of cases where it may neither be possible nor advisable. In determining when to choose transaction recovery, you must consider several questions:

1. **Transaction Identification.** Can all the problem transactions be identified? You must be able to actually identify the transactions that will be removed from the database. Can all the work that was originally done be located and redone?
2. **Data Integrity.** Has anyone else updated the rows since the problem occurred? If they

have, can you still proceed? Is all the data required still available? Recovering after a REORG, LOAD, or mass DELETE may require the use of image copies or the table space may need to be undamaged. Will any other data be lost? If so, can the data lost be identified in some fashion?

3. **Availability.** How fast can the application become available again? Can you afford to go offline?

These questions actually boil down to a matter of cost. The ultimate transaction recovery solution should analyze your environment and the transaction(s) needing to be recovered, and recommend which type of transaction recovery to perform.

Transaction Recovery Solutions

In order to avoid the high cost in manpower, time, and machine resources of a manual transaction recovery process, a robust transaction recovery solution is needed. The first requirement is the ability to read and analyze log data to:

- provide detailed diagnostic information about the selected transactions
- identify where recovery should begin
- find all bad transactions
- store information for analysis
- generate UNDO and/or REDO SQL for the different types of transaction recoveries
- provide a high speed capability for applying the SQL statements during the UNDO and/or REDO processing

The transaction recovery solution should address all of the problems associated with quickly analyzing problem transactions, determining the scope of the recovery, getting information to determine the best recovery method, and managing recovery resources that result in optimal recovery performance. Additionally, your solution should produce a work estimate analysis to provide the information needed to choose the optimal recovery solution. This analysis should be based on the relative cost of each option: PIT, UNDO, or REDO.

Although several products could be used to provide a transaction recovery solution, an integrated solution is key to providing a tool that

makes transaction recovery an option that can be chosen with confidence.

Summary

In summary, transaction recovery has become a critical need in any complete recovery toolbox. The transaction recovery solution chosen must provide:

- Powerful diagnostic features for problem identification
- Automated assistance in choosing optimal recovery method
- Features that provide speed, manageability, and accuracy

And with a transaction recovery solution in your arsenal, you just might be able to deliver the availability required to help transform your company into an e-business.

From Database Trends, February 2000.

Home.