High-Speed Transaction Recovery

By Craig S. Mullins

Availability is the Holy Grail of database administrators. If your data is not available, your applications can not run, and therefore your company is losing business. Lost business translates into lower profitability and, perhaps, a lower stock valuation for your company. These are all detrimental to the business, so the DBA must do everything in his power to ensure that databases are kept online and operational. This has been the duty of the DBA since the days of the first DBMS.

But the need for availability is increasing. The days of the long batch window where databases can be offline for extended periods to perform nightly processing are diminishing. Exacerbating this trend is the drive toward e-business. And when your business is on the World Wide Web, it never closes.

These demands for higher availability make traditional forms of database recovery inadequate. Today's DBA must understand Transaction Recovery techniques to be able to prepare an optimal approach for every recovery situation. This article will cover Transaction Recovery from a DB2 for OS/390 point-of-view.

When DBAs hear the word "recovery," the first thing that usually comes to mind is handling 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 traditional type of 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 PIT recovery will remove the effects of all transactions since that specified point in time. This can cause problems if there were any valid transactions during that timeframe that needed to be kept.

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

Historically, recovery was performed primarily to overcome disasters and hardware failures. However, this is simply not the case anymore. Application failures, not hardware failures, are the predominant drivers of recovery needs. Industry analysts at the GartnerGroup estimate that as much as 80 percect of application errors are due to software failures and human error. Although hardware and operating system failures were common several years ago, modern hardware and operating systems are more reliable, with a high mean time between failure.

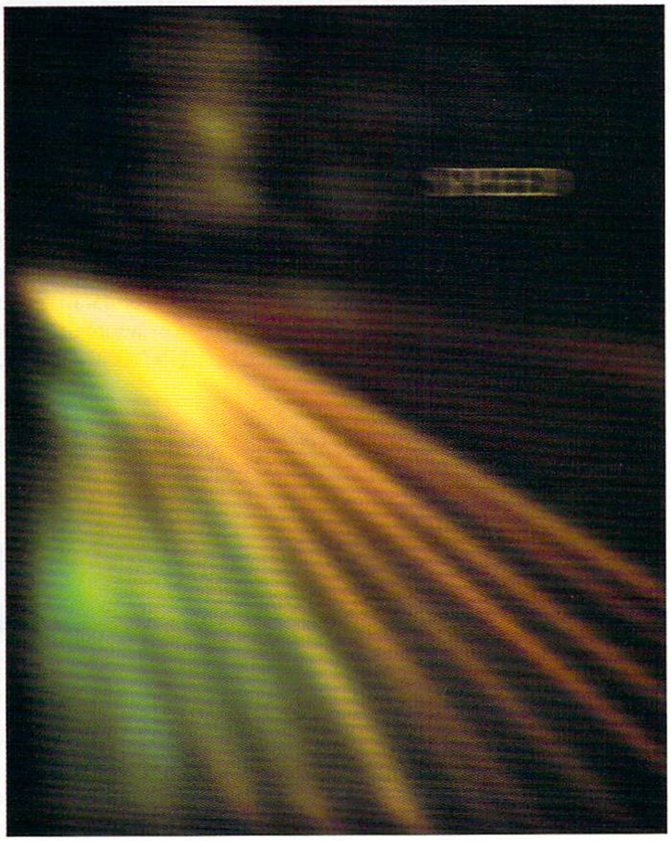
In reality, except for disaster recovery tests, very few DBAs need to perform true disaster recovery. While media does fail, it's actually quite rare. User errors and application failures are the most common causes of problems requiring recovery, and therefore, the primary cause for system unavailability. Any number of problems can occur at the application level.

As databases grow in size and complexity, so do the chances that bad transactions will corrupt the data on which your business depends.

Transaction Recovery Defined

Transaction Recovery is the process of removing the undesired effects of specific transactions from the database. This statement, while simple on the surface, hides a bevy of complicated details.

Traditional recovery is at the database object level: for example, at the tablespace or index level. When performing a



traditional recovery, a specific database object is chosen, a backup copy of that object is applied, and then log entries are applied for changes that occurred after the image copy was taken. This approach is used to recover the database object to a specific, desired point in time. If multiple objects must be recovered, this approach is repeated for each object impacted.

Transaction Recovery allows a user to recover a specific portion of the tablespace, based on user-defined criteria, so only a portion of the data is affected. Any associated indexes are automatically recovered as the transaction is recovered. The transaction may impact data in multiple tablespaces, too.

A transaction is a set of related operations that, when grouped together, define a logical unit of work within an application. Transactions are defined by the user's view of the process, for example, the set of panels comprising a new hire operation or perhaps the set of jobs that post to the general ledger.

Using Transaction Recovery, application problems can be addressed quicker, thereby maintaining a higher level 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).

There are three types of Transaction Recovery: PIT, UNDO and REDO.

Point-in-Time Recovery

Point-in-time recovery is the simplest strategy. It also is the only one supported by native DB2 utilities (see Figure 1). With PIT recovery, you remove all transactions since a given point in time and then manually reenter the valid work. The desired result is to maintain "Good Transaction 1" and "Good Transaction 2," while removing the "Bad Transactions" from the system.

You must be able to determine a common recovery point for a set of tablespaces. A DB2 QUIESCE works fine, but if that is not available, you will have to determine a point of consistency to be used for recovery.

After the point-in-time recovery, good transactions are missing from the database. If the information is still available, the user could rerun or reenter "Good Transaction 2."

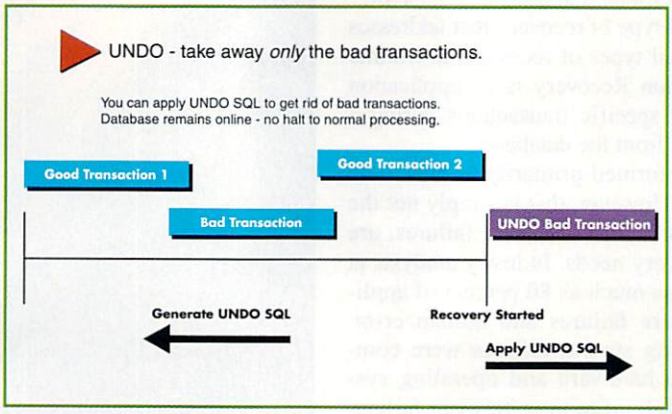

UNDO - take away *only* the bad transactions.
You can apply UNDO SQL to get rid of bad transactions. Database remains online - no halt to normal processing.
Good Transaction 1
Good Transaction 2
Bad Transaction
UNDO Bad Transaction
Generate UNDO SQL
Recovery Started
Apply UNDO SQL

Figure 2: UNDO Transaction Recovery

Regardless of the type of recovery to be performed, if the error that caused the recovery to be performed is caught too late, subsequent processing could have occurred using the "bad data." How to deal with these types of problems depends on the nature of the data and the type of updates applied, and needs to be handled on an application by application basis.

UNDO Transaction Recovery

The second possibility is to deploy UNDO Transaction Recovery (see Figure 2). 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 generate UNDO SQL, the DB2 log is read to find the data modifications that were applied during a given timeframe and:

- INSERTs are turned into DELETEs
- DELETEs are turned into INSERTs
- UPDATEs are reversed to UPDATE to the old value

To accomplish this transformation, a solution is required that understands the DB2 log format and can create the SQL needed to undo the data modifications.

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 application or database outage. But, the potential for anomalies causing failures in the UNDO is certainly a consideration.

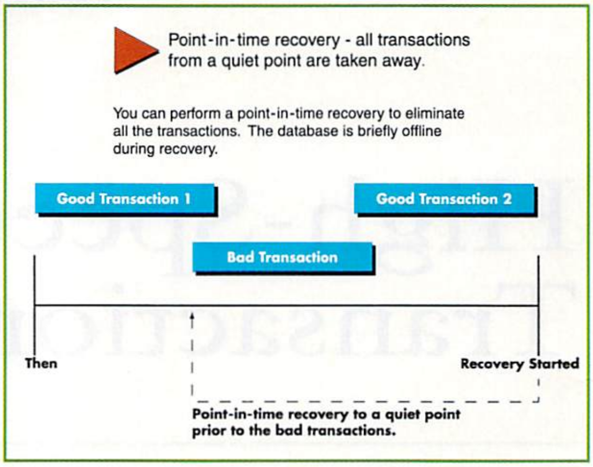

Point-in-time recovery - all transactions from a quiet point are taken away.
You can perform a point-in-time recovery to eliminate all the transactions. The database is briefly offline during recovery.
Good Transaction 1
Good Transaction 2
Bad Transaction
Then
Recovery Started
Point-in-time recovery to a quiet point prior to the bad transactions.

Figure 1: Point-in-Time Recovery

REDO Transaction Recovery

The REDO Transaction Recovery strategy is a combination of the first two recovery techniques we have discussed – but with a twist (see Figure 3 on page 68).

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 PIT recovery, eliminating all the transactions since the recovery point. Finally, we reapply 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 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 tablespace to a state that does not include the problem transactions.

To generate the REDO SQL statements, you will need a solution that can read the DB2 log and create the necessary SQL 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 can be accomplished with the data online, thereby reducing application downtime.

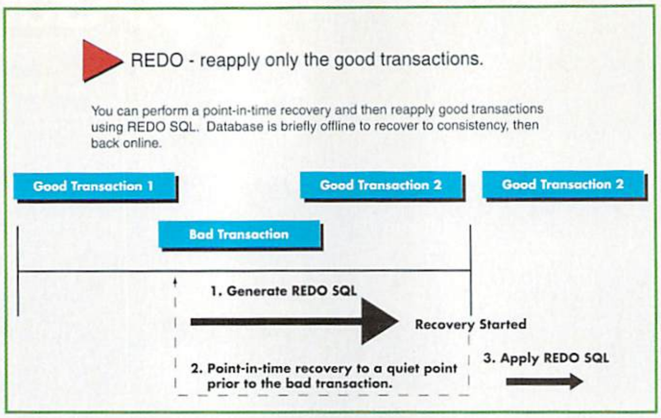

REDO - reapply only the good transactions.
You can perform a point-in-time recovery and then reapply good transactions using REDO SQL. Database is briefly offline to recover to consistency, then back online.
Good Transaction 1
Good Transaction 2
Good Transaction 2
Bad Transaction
1. Generate REDO SQL
Recovery Started
2. Point-in-time recovery to a quiet point prior to the bad transaction.
3. Apply REDO SQL

Figure 3: REDO Transaction Recovery

Choosing the Optimum Recovery Strategy

While Transaction Recovery may seem like the answer to all of your recovery problems, there are a number of cases where it may be neither possible, nor advisable. Consider the following questions when determining if Transaction Recovery is appropriate:

1. *Transaction Identification.* Can all 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 required data still available? Recovering after a REORG, LOAD or mass DELETE may require image copies. Will any other data be lost? If so, can the lost data be identified somehow?

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

These questions boil down to a matter of cost. What is the cost of rework and is it actually possible to determine what would need to be redone? This cost needs to be balanced against the cost of long scans of log data sets to isolate data to redo or undo, and the cost of applying that data using SQL.

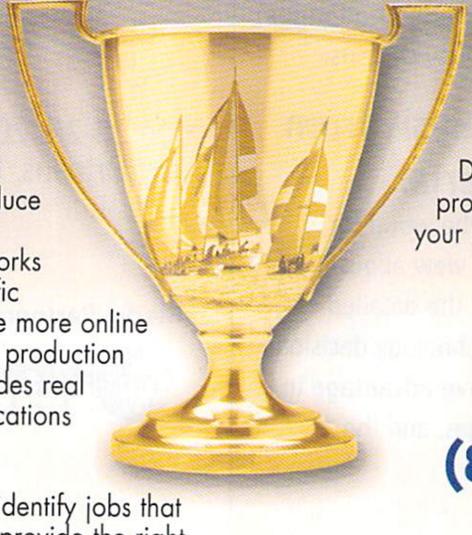
The ultimate Transaction Recovery solution should analyze your environment and the transaction(s) needing to be recovered, and recommend which type of Transaction Recovery to perform.

Transaction Recovery Planning

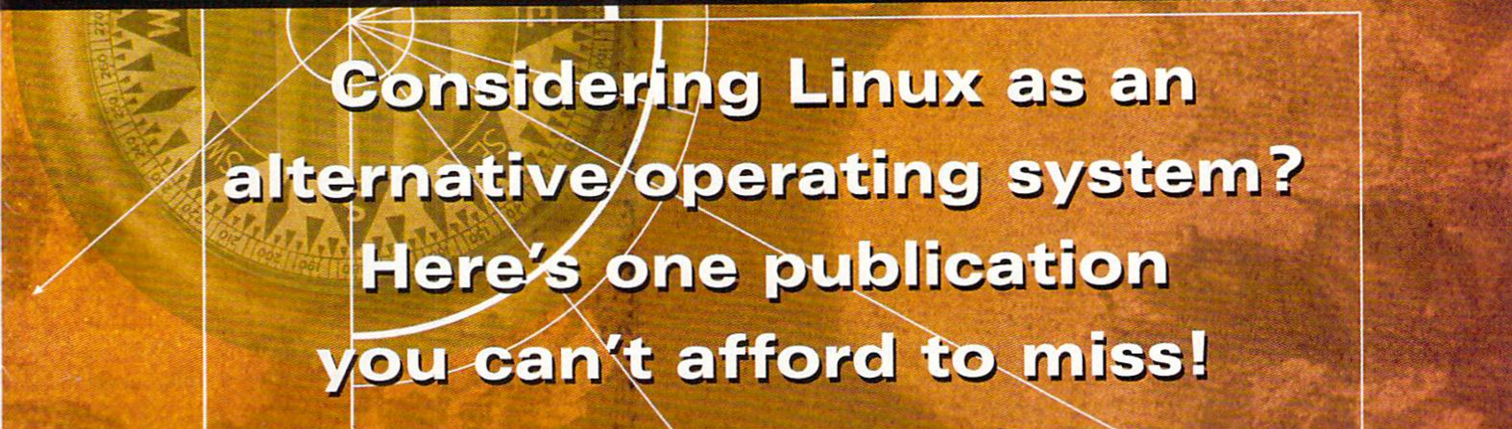
When planning for Transaction Recovery, you should review your image copy frequency. This frequency will help determine how far back you have to go to get a consistent state for a PIT recovery. Also, review your policy on Data Change Capture (DCC). DCC is an option of the DB2 CREATE TABLE statement that causes logged data changes to be written in an expanded format. To implement DCC, alter your tables to include the clause DATA CAPTURE CHANGES. With DCC enabled, generating UNDO and REDO SQL statements will be faster because the SQL generation process need not perform expensive row completion operations. However, your Transaction Recovery solution should be able to function with, or without, DCC enabled.

To effectively plan for and perform a Transaction Recovery, follow these steps:

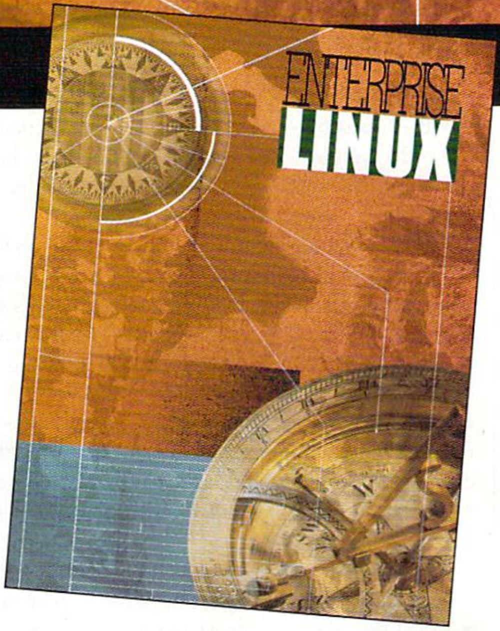
1. Identify the problem.
2. Identify the transactions causing the problem.



FREE SUBSCRIPTION OFFER!



Considering Linux as an alternative operating system? Here's one publication you can't afford to miss!



ENTERPRISE LINUX

Business Models For OpenSource Computing

For many IS Managers, Linux has made it into the "consideration" phase. No longer regarded as merely a fringe movement, Linux has emerged as a viable server operating system.

101 Communications, the innovators in platform-specific publishing, announces its newest publication, *ENTERPRISE LINUX*. Published quarterly, *ENTERPRISE LINUX*'s editorial mission is to approach Linux as IT has never been approached before – as a ready-for-prime-time operating system in the enterprise. *ENTERPRISE LINUX* looks at Linux applications in the real world and brings IS Managers information that will enable and justify their decisions to migrate to Linux or stay put. *ENTERPRISE LINUX* informs, educates and validates this emerging platform like no other publication in the market.

Free subscriptions. Subscribe online today!

www.enterpriselinuxmag.com

For advertising opportunities,
contact Eric Derby, (973) 786-0340.

101 communications

Anomaly Outcomes

As a rule of thumb, anomaly analysis will yield one of two outcomes: little activity and a feasible transaction recovery, or significant activity and a difficult or impossible transaction recovery. Let's take a look at several examples of anomalies. Consider the following scenario:

- 8:00 Monday – row is added for a new customer.
- 12:00 Monday – new customer enters a large order.
- 18:00 Monday – new customer provides a new shipping address.

Now, think about the havoc that would be caused if the same transaction that updates the shipping address at 18:00 also accidentally clears the total orders column for the customer. Simply backing out the transaction would remove the incorrect data (the total orders column), but it would also cause you to lose valid data (the new shipping address).

The key to this issue is an understanding of the error. Once it is understood that the only column in error is the total orders field, the Transaction Recovery can be made sensitive to that column only.

Let's look at a more complicated case. Consider the following scenario:

- 8:15 Tuesday – row is added for a new customer.
- 9:30 Tuesday – new customer fails a credit check and is deleted.
- 9:00 Wednesday – new customer reapplies with updated information and passes the credit check.

In this scenario, what would happen if the job that added the new customer at 8:15 on Tuesday runs out of sequence, causing numerous other problems? Knowledge of the error alone is insufficient to understand the implications of backing it out.

The desired result, actually, is to leave the new customer alone since the final record in the database is actually correct. If you just rollback the bad job, the insert of the new customer's first record will be turned into a delete that will remove the good record. The only alternative is to scan the log for any other activity on the new customer's record and report it. Once reported, the DBA can analyze the anomalies and hopefully make some reasonable choices on how to proceed. The bottom line is that the Transaction Recovery process needs to take this type of scenario into account, or good data may be lost.

Let's look at another case:

- 13:00 Monday – employee receives a pay increase.
- 23:00 Tuesday – employee's increase is updated in the PAYROLL table.
- 12:00 Thursday – employee gets the promotion associated with the pay increase and it is updated into the EMPLOYEE table.

What if the payroll clerk mis-entered some data on Tuesday night when the employee's increase was entered into the PAYROLL table, and the error is not discovered until Thursday afternoon?

The timeliness of problem recognition is crucial. The sooner you recognize a problem, the more feasible Transaction Recovery becomes. Had the problem with the clerk been recognized on Wednesday prior to the next payroll run, the problem would have been manageable. By waiting until Thursday, the possibility of accurately backing out the bad clerk's data has decreased significantly. Even if it turns out to be possible, a great deal of manual effort will be required to select the set of data to rollback.

If you simply rollback the clerk's work, both the employee's pay and promotion will be lost. You may have planned on reentering the transactions from Tuesday at 23:00, but did you plan on reentering all the other subsequent transactions, such as those from Thursday at 12:00?

– C.M.

3. Identify available recovery points.
4. Identify the relative cost of UNDO versus PIT + REDO (Strategy Selection).
5. Identify the cost of performing additional operations to resolve any integrity issues caused by UNDO.
6. Choose the optimal recovery strategy: PIT, PIT+REDO, or UNDO.
7. Generate the recovery job(s).
8. Execute the recovery job(s).

Transaction Recovery Solutions

The Transaction Recovery solution should provide a single point-of-control over each stage of the recovery process. A single interface for the entire Transaction Recovery solution minimizes the difficulty of analyzing, generating and executing the recovery. Switching from one interface to another to accomplish the recovery would introduce needless complexity to an already volatile situation.

It must offer sophisticated filters to identify the objects that are affected by the transaction. It should be possible to identify the transaction by many different criteria, for example, by correlation ID (batch job), correlation type (batch, CICS, IMS), plan name, authorization ID, tablespace name, table name, object identifier (OBID), column name, update type or other such qualifying criteria.

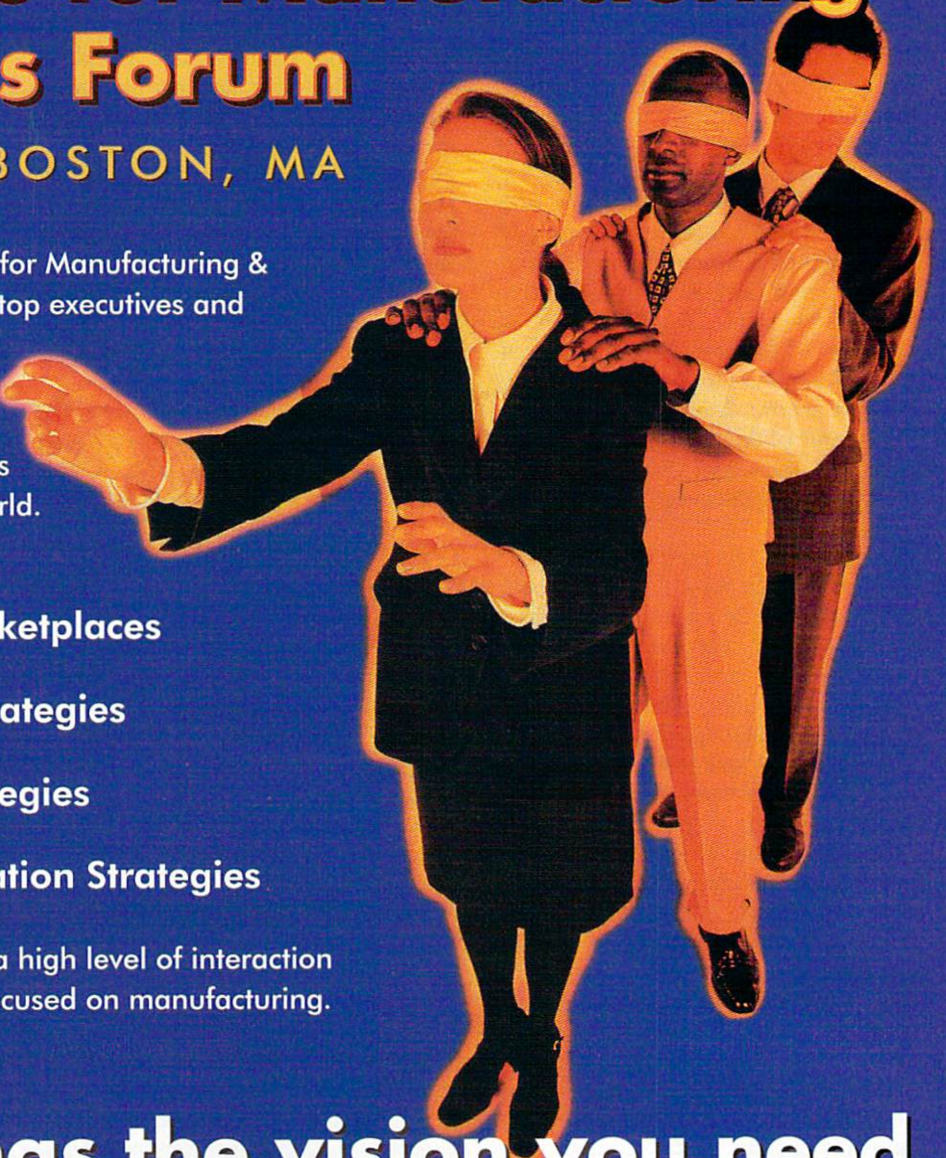
A Transaction Recovery solution should provide the ability to read and analyze log data to provide detailed diagnostic information about the selected transactions, diagnose the problem and identify where recovery should begin, find all bad transactions, and generate UNDO and/or REDO SQL.

It should provide assistance in choosing the optimal recovery method. The solution must understand the tradeoffs in terms of the scope of the transaction, the potential impact on the application, and the time required for each type of recovery to determine whether to perform PIT, UNDO or REDO Transaction Recovery.

Finally, it should deliver automated job generation to eliminate programming errors by providing the ability to restart the process should the recovery fail anywhere along the way; a high-performance "capture agent" to quickly gather the required elements of the Transaction Recovery; and a high-speed capability for applying the SQL statements during the UNDO and REDO processing.

Your solution should provide reports on image copy frequency and Data Change Capture impact to help determine your readiness for Transaction Recovery. During an actual Transaction Recovery, your solution must be able to analyze the log for a quiet point. Quiet points are often used as a beginning point of recovery. You still must determine if the quiet point found is the correct starting point for the recovery that you need to perform.

A very important feature of the Transaction Recovery solution is the ability to assist you in choosing between the various recovery methods. Once the appropriate transaction definition and recov-



High-Speed

Continued from page 70

ery information is collected, a relative work estimate should be prepared for each type of transaction recovery.

Backout Integrity

Maintaining backout integrity is critical. It must be possible to determine the feasibility and impact of performing an UNDO or REDO process. Data may have been changed after the offending application process was run. Data changes of this nature can have an impact on the integrity of the data after a proposed Transaction Recovery.

Analysis of the subsequent activity on the database is critical to determining the feasibility of the recovery. Generally, the activity will conform to known application patterns. It is the responsibility of the DBA and the application group together to determine the impact of the recovery on the subsequent work (and vice versa). It may be possible that through judicious selection of the rows to be processed and the columns to include that only the data in error can be processed.

As a rule of thumb, anomaly analysis will yield one of two outcomes: Either there is little subsequent activity and a transaction recovery is feasible, or there is a significant amount of activity and transaction recovery may be difficult or impossible. See the sidebar on page 70 for anomaly examples.

Applying the SQL for Transaction Recovery

Speed is critical for Transaction Recovery. Your Transaction Recovery solution should use a multitasked SQL apply process that efficiently distributes work among multiple streams for parallel processing. The apply process must be restartable should it fail. Workload balancing by table and partition with respect for referential integrity constraints and table group should be supported. The high-performance apply should give you the ability to convert from dynamic SQL to static SQL "on the fly" to enhance performance. Other desirable features include the ability to process very large volumes of transactions, and the option to apply changes online or in batch.

The apply process should have the capability to react to anomalies found during the processing of the SQL. A minimum set of capabilities would include:

- The ability to retry on deadlock situations
- The ability to log and defer SQL statements with problems (e.g., inserts where the key is already in the table, updates where the column values do not match, etc.)
- The ability to ignore any errors and just continue processing

Applications are prone to all types of problems, bugs and errors. Therefore, Transaction Recovery has become a critical need in any complete recovery toolbox, especially for e-businesses.

The Transaction Recovery solution chosen must provide powerful diagnostic features for problem identification; automated assistance in choosing an optimal recovery method; Integrated Transaction Recovery analysis; generation and apply features; and features that provide speed, manageability and accuracy. ⬙

About the Author:

Craig S. Mullins is Director of Technology Planning for BMC Software, and is the author of the book DB2 Developer's Guide. *He can be reached via his Web site at www.craigsmullins.com.*

High Returns

Continued from page 32

by pass-through ports that give ample room for complex wiring requirements. Vertical cable management channels, separate power and data cables. Wire management covers along slide panels detach to provide pass-through ports for cabling. Top and bottom cable ports every 24 inches along the unit framing simplify floor-to-unit and ceiling-to-unit cabling.

Since installing the command center, UMB's growth has required the addition of monitoring points, but the day-shift staff has held steady at eight employees. The other benefit is that operators can monitor all systems closely, detecting potential problems in time to prevent headaches for the end users.

The most important aspect of the new console, however, is practical. It helps the operations staff do their jobs better, something UMB customers around the country will appreciate. ⬙

About the Author:

Steve Collins is Assistant Vice President/Manager of the MIS Data Center Services for UMB Bank (Kansas City, Mo.).

Streamlining *Continued from page 32*

ticular workstation does and concentrate, instead, on what users need. The use of directories would virtually eliminate desktop maintenance and provide an extremely scalable environment whereby IT manages central administration servers while software is installed and run on users' desktops.

The implementation of directories provides IT with a formidable, yet flexible, defense against the risks that result from insufficient authentication, lax security, and non-existent or unenforceable use policies. At a time when most organizations are beginning to implement e-commerce and e-business, it is more critical than ever to have a system in place that identifies users and includes mechanisms that help IT define and enforce users' privileges and restrictions. In addition, as the number of users grows, the implementation of directories is an effective tool for controlling the cost of keeping corporate systems running, accessible and secure.

The use of directories breaks IT administration down into three clear phases: System Setup, Application Deployment, and Administration and Management. During the System Setup phase, the network administration group defines users and other entities in a directory. Usage policies are then set and appropriate software installed on a server from which it will later be deployed.

The Application Deployment phase occurs on demand when users need an approved application. Deployment might involve dispatching full applications to desktops or Java thin clients to PDAs, or simply providing a URL that allows the user to access more information through a Web-based portal. Again, every time software is deployed, it is first checked against the user privileges stored in the directory and only provided to approved users.

About the Author:

Yoram Baltinester is Director of Product Management at NetManage. He can be reached at balti@netmanage.com.