Payoff

When a DBMS crashes, all or a portion of the data can become unusable. Appropriate procedures must be followed to restore, validate, and return the system to normal. In a client/server environment with distributed data bases, additional procedures are needed. Users and IS professionals must do their part to ensure the security, integrity, and validity of information and DBMS transactions.

Problems Addressed

Many organizations, such as banks and airlines, have online computer systems that must function at all times. In most online applications, there are many application programs that access data bases concurrently. Therefore, the data bases must be correct and up-to-date at all times.

Yet technology is imperfect and computer systems are subject to many types of failure. When a system fails, recovery procedures must be in place to restore, validate, and return the system to normal.

Information is an essential tool used by all levels of management in planning and organizing, directing, and controlling an organization. Therefore, the security, availability, and integrity of information are of utmost importance. Technological advances have significantly influenced the way an organization’s information is collected, processed, and distributed. Data base management systems (DBMSs) have evolved from some of these technological advances and are of primary concern to auditors and IS managers who are responsible for securing an organization’s data while facilitating the efficient dissemination of information. Although DBMSs can organize, process, and generate information designed to meet user needs, the integrity and security of this information are also essential to protect users.

Importance of DBMS Recovery

Recovery—that is, the return to a fully operational environment after a hardware or software failure—is an important process. Moreover, the effects of a system failure on the organization must be curtailed to minimize any substantial financial loss. Actions must be taken to prevent DBMS failures or resolve them quickly if they occur.

It is not always cost-effective to implement all possible DBMS controls and use all known review techniques. The choice of whether or not to audit can have a direct impact on the financial consequences caused by these failures. A review of DBMS recovery ensures adherence to appropriate practices and procedures and minimizes business losses. A review further ensures that an organization can recover and return to full operational status following a disaster. For example, the January 1994 earthquake in the Los Angeles area caused sustained interruption of business in many organizations; those organizations that had established recovery procedures were able to more readily restore operations and minimize losses.

Developing, implementing, maintaining, and auditing the DBMS recover controls and processes involve a considerable amount of money and company resources. Costs and
benefits must be considered to ensure that company resources are expended efficiently. Systems managers who are either developing or maintaining a DBMS must understand data base structures and participate in the recovery process. This article explains the process and techniques for reviewing DBMS recovery.

The Recovery Process

The DBMS recovery process is designed to restore data base operations to their prefailure status. Users and IS professionals play a critical role in restoring the DBMS to operation; that is, after the system has been successfully restored, the entire staff must participate to ensure the security, integrity, and validity of the information and its transaction properties.

Transaction Properties

The transaction is the fundamental activity of a DBMS and an area of concern for the reviewer. transactions maintain consistency constraints or controls determined for an application. This consistency must be maintained at all times, even during a transaction failure. concurrent processing must also be protected against adverse effects during a transaction failure.

A transaction is a command, message stream, or input display that explicitly or implicitly calls for a processing action (e.g., updating a file). Transaction Processing is a sequential process that does not overlap or parallel a single application. It is started with a BEGIN TRANSACTION and ended with an END TRANSACTION identifier. The following typical transaction properties must be reviewed in assessing recovery controls:

- **Atomicity.** During a transaction, either all or none of its operations are performed on the data base; that is, atomicity ensures the preclusion of partially completed transactions.

- **Permanence.** If a transaction completes the END TRANSACTION function the results of its operation will never subsequently be lost.

- **Serialization of transactions.** If more than one transaction is executed concurrently, the transactions affect the data base as if they were executed in serial order; this ensures that concurrently executing jobs do not use inconsistent data from partially completed transactions.

- **Prevention of cascading aborts.** An incomplete transaction cannot reveal results to other transactions, thereby limiting the effect of a transaction error throughout the entire system.

- **Consistency.** A transaction that reaches its usual end commits its results to memory, thereby preserving the consistency of the data base contents.

Transactions are more effective when written in Sybase, Oracle, Access, or Structured Query Language than in Common Business Oriented Language, Formula Translation, or Beginner's All-Purpose Symbolic Instruction Code. They are well suited to structured programming and can help make systems development a routine process by modularizing the actions being performed in code and simplifying the treatment of failures and concurrency. These transaction properties have specific control functions (which, from a
review standpoint, should be organized and verified for DBMS operational validity and reliability).

**Causes of DBMS Failure**

There are many causes of DBMS failure. When a DBMS fails, it falls into an incorrect state and will likely contain erroneous data. Typical causes of DBMS failures include errors in the application program, an error by the terminal user, an operator error, loss of data validity and consistency, a hardware error, media failures, an error introduced by the environment, and errors caused by mischief or catastrophe.

Typically, the three major types of failure that result from a major hardware or software malfunction are transaction, system, and media. These failures may be caused by a natural disaster, computer crime, or user, designer, developer, or operator error. Each type of failure is described in the following paragraphs.

**Transaction Failure.**

Transaction failures occur when the transaction is not processed and the processing steps are rolled back to a specific point in the processing cycle. In a distributed data base environment, a single logical data base may be spread across several physical data bases. Transaction failure can occur when some, but not all, physical data bases are updated at the same time.

**System Failure.**

System failure can be caused by bugs in the data base, operating system, or hardware. In each case, the Transaction processing is terminated without control of the application. Data in the memory is lost; however, disk storage remains stable. The system must recover in the amount of time it takes to complete all interrupted transactions. At one transaction per second, the system should recover in a few seconds. System failures may occur as often as several times a week.

**Media Failure.**

Disk crashes or controller failures can occur because of disk-write bugs in the operating system release, hardware errors in the channel or controller, head crashes, or media degradation. These failures are rare but costly.

By identifying the type of DBMS failure, an organization can define the state of activity to return to after recovery. To design the data base recovery procedures, the potential failures must be identified and the reliability of the hardware and software must be determined. the following is a summary of four such recovery actions:

- **TRANSACTION UNDO.** a transaction that aborts itself or must be aborted by the system during routine execution.

- **GLOBAL REDO.** When recovering from a system failure, the effects of all incomplete transaction must be rolled back.

- **PARTIAL UNDO.** While a system is recovering from failure, the results of completed transactions may not yet be reflected in the data base because execution has been
terminated in an uncontrolled manner. Therefore, they must be repeated, if necessary, by the recovery component.

- **GLOBAL UNDO.** If the data base is totally destroyed, a copy of the entire data base must be reloaded from a backup source. A supplemental copy of the transaction is necessary to roll up the state of the data base to the present.

These definitions imply that the transaction is the sole unit of recovery in the data base, which reduces the programmer's required recovery processing inclusions.

**Disaster Recovery Tools**

There are numerous disaster recovery tools and techniques available that can assist an organization in DBMS recovery. For example, several software programs can scan the network and data bases and record the configuration of the systems. Novell's Netware Management System provides a logical view of the system and network. Other products such as LT Auditor, LAN automatic Inventory, Palindrome, and LAN Directory are software support packages that can assist in DBMS documentation and recovery processes and functions.

Several companies use video camera techniques to document equipment, hardware, and software configurations. Others are using CD-ROMS as a technique for saving critical DBMS structures and data. Such applications of media or computer-assisted disaster recovery tools and techniques is providing organizations detailed documentation to aid troubleshooting DBMS problems, identifying weak points in the network or distributed DBMS, and reconstructing destroyed DBMSs.

**Techniques for Reviewing DBMS Recovery**

The review of a DBMS recovery must ensure that employees with specific responsibilities perform their functions in accordance with operational policy and procedure. There are several useful DBMS recovery review techniques.

There are two ways to make the system operate again. First, all transactions that have occurred since the last backup can be reapplied, which would bring the data base up to date. Second, the current contents of the data base can be taken and all transactions can be backed out until the integrity and validity of the data are restored. Whichever method is selected, it should be documented and a checklist of specific tasks and responsibilities identified.

The DBMS typically provides exhaustive review trails so that the system can know its exact state at any time. These review trails should be complete enough to reconstruct transactions and aid in recovery procedures. A data base administrator should know how to use these review trails in recovery to fully understand the inner workings of the DBMS.

A data base that has been backed up regularly helps the system recover from a failure and begin operating again as soon as possible. Daily backups are sufficient in most organizations. Those organizations that must always have current data must sometimes perform hourly backups. Each backup should be well documented to provide further insight into the review process.

Review techniques should examine application design, security procedures, and personnel control to ensure that managers can meet emergencies and have effective contingencies in place. These three areas are extremely critical review points for the auditor, management, users, and IS personnel.
Application Design

It is important to build sound recovery procedures and processes into an application during the design phase. The design of an application should take into consideration the data base control issues that affect backup and recovery processes. Possible weaknesses in controls include:

- Inaccurate or incomplete data in the data base.
- An inadequate audit trail.
- An inadequate service level.
- Failure of the DBMS to function as specified.
- Inadequate documentation.
- Lack of processing continuity.
- Lack of management support.
- Fraud or embezzlement.

The data base administrator should be responsible for examining the backup and recovery controls being considered by the user and developer when reviewing application design. The user and the developer of the application must assess the risks of not having appropriate controls in place to aid in recovery. Some key controls that should be adopted are:

- **Review trails.** A method of chronologically recording system activities that allows the reconstruction, review, and examination of each event in a transaction from inception to the final results.
- **Recovery procedures.** Automated or manual tools and techniques for recovering the integrity of a data base.
- **Application system failure procedures.** Procedures for users to follow in the event that their applications cannot operate.
- **Checkpoint data bases.** Copies of the data base and transaction files that are made at specific point in time for recovery purposes.

At a minimum, these controls should be tested during the module and integration testing phases of development. In terms of a new system review before implementation, these controls are most effective if thoroughly validated and approved by the user and developer before the system is placed into operation. One important issue to be considered in application design is data integrity.
Maintaining Data Integrity.

Data integrity concerns the accuracy of the contents of the data base. The integrity of the data can be compromised because of failures (i.e., events at which the system fails to provide normal operation or correct data). Failures are caused primarily by errors, which may originate in programs, interactions between these programs, or the system.

A transaction is a sequence of actions. It should be designed and executed so that it either is successfully completed or has no effect on the data base. A transaction can fail to be completed for the following reasons:

- An action violates a security or integrity constraint.
- The user cancels the transaction.
- An unrecoverable I/O error occurs.
- The system backs out the transaction to resolve a deadlock.
- The application program fails.
- The system crashes.

Semantic Integrity.

This refers to the accuracy of the data base despite the fact that users or applications programs try to modify it incorrectly. Assuming that the data base security system prevents unauthorized access, and hence malicious attempts to corrupt data, most potential errors will be caused by incorrect input, incorrect programs, or lack of user understanding.

Traditionally, most integrity checking has been performed by the application programs and by periodic auditing of the data base. The following are some problems that occur when relying on application programs for integrity checking:

- Checking is likely to be incomplete because the applications programmer may not be aware of the semantics of the complete data base.
- Each application program relies on other programs that can modify the data base, and a problem in one program could corrupt the whole data base.
- Code that enforces the same integrity constraints occurs in several programs. This leads to unnecessary duplication of the programming effort and exposes the system to potential inconsistencies.
- The criteria for integrity are buried within procedures and are therefore difficult to understand and control.
- Maintenance operations performed by users of high-level query language cannot be controlled.

Most of these errors could be detected through auditing, although the time lag in detecting errors by auditing can cause problems, such as difficulty in tracing the source of
an error and hence correcting it as well as incorrect data used in various ways, causing errors to propagate through the data base and into the environment.

The semantics, or meaning, of a data base is partly drawn from a shared understanding among the users, partly implied by the data structures used, and partly expressed as integrity constraints. These constraints are explicitly stated by the individuals responsible for data control. Data bases can also be classified as:

- A single record or set.
- Static or transitional.
- General or selective.
- Immediate or deferred.
- Unconditional or conditional.

A system of concurrent transactions must be correctly synchronized—that is, the processing of these transactions must reach the same final state and produce the same output. Three forms of inconsistency result from concurrence: lost updates, an incorrect read, and an unrepeatable read. Lost updates can also result from backing up or undoing a transaction.

**Correcting Inconsistency Problems.**

The most commonly used approach to eliminate consistency problems is locking. The DBMS can use the locking facilities that the operating system provides so that multiple processes can synchronize their concurrent access of shared resources. A lock can be granted to multiple processes, but a given object cannot be locked in shared and exclusive mode at the same time. Shared and exclusive modes conflict because they are incompatible. The operating system usually provides lock and unlock commands for requesting and releasing locks. If a lock request cannot be granted, the process is suspended until the request can be granted. If transactions do not follow restrictive locking rules, Deadlock can occur. Deadlock can cause the loss of an entire file; therefore, it is critical to have a recovery system in place to alleviate this problem.

The Deadlock problem can be solved either by preventing Deadlock or by detecting them after they occur and taking steps to resolve them. Deadlock can be prevented by placing restrictions on the way locks are requested. They can be detected by examining the status of locks. After they are detected, the Deadlock can be resolved by aborting a transaction and rescheduling it. Methods for selecting the best transaction to abort have also been developed.

A synchronization problem can occur in a distributed data base environment, such as a client/server network. Data bases can become out of sync when data from one data base fails to be updated on other data bases. When updates fail to occur, users at some locations may use data that is not current with data at other locations. Distributed data bases provide different types of updating mechanisms. In a two-phase commit update process, network nodes must be online and receive data simultaneously before updates can occur. A newer update method called *data replication* enables updates to be stored until nodes are online and ready to receive. Update methods must ensure currency in all network data bases.
Security Procedures

A data base usually contains information that is vital to an organization's survival. A secure data base environment, with physical and logical security controls, is essential during recover procedures.

Physical Security.

In some distributed environments, many physical security controls, such as the use of security badges and cipher locks, are not feasible and the organization must rely more heavily on logical security measures. In these cases, many organizational members may have data processing needs that do not involve a data base but require the use of computer peripherals.

Logical Security.

Logical security prevents unauthorized users from invoking DBMS functions. The primary means of implementing this type of security is the use of passwords to prevent access to files, records, data elements, and DBMS utilities. Passwords should be checked to ensure that they are designated in an intelligent, logical manner.

Security Logs.

Each time an unauthorized user attempts to access the data base, it should be recorded in a security log. Entries in this log should consist of user ID, terminal or port number, time, date, and type of infraction. With this information, it is possible to investigate any serious breaches of security. From the data base administrator's standpoint, evidence that the DBMS is detecting security violations and that a consistent procedure is used to follow them up should be sufficient.

Personnel Control

Data base recovery involves ensuring that only authorized users are allowed access and that no subsequent misuse of information occurs. These controls are usually reestablished when a system becomes operational. When operations cease or problems occur, however, controls often become inoperative.

The three primary classes of data base users are data base administrator, applications and systems programmers, and end users--and each has a unique view of the data. The DBMS must be flexible enough to present data appropriately to each class of user and maintain the proper controls to inhibit abuse of the system, especially during recovery, when controls may not be fully operational.

Data Base Administrator.

The data base administrator is responsible for ensuring that the data base retains its integrity and is accountable if the data base becomes compromised, no matter what circumstances arise. This individual has ultimate power over the schema that the organization has implemented. Any modifications or additions to this schema must be approved by the data base administrator. Permission to use subschema (i.e., logical views)
is given to end users and programmers only after their intentions are fully known and are consistent with organizational goals.

Because the data base administrator has immediate and unrestricted access to almost every piece of valuable organizational information, an incompetent employee in this position can expose the organization to enormous risk, especially during DBMS recovery. Therefore, an organization should have controls in place to ensure the appointment of a qualified data base administrator.

The data base administrator must ensure that appropriate procedures are followed during DBMS recovery. The data base administrator should also validate and verify the system once it has been recovered before allowing user access so that if controls are not functioning or accessing problem continue, users will not be affected.

**Applications and Systems Programmers.**

After recovery, programmers must access the data base to manipulate and report on data according to some predetermined specification or to access whether data loss has occurred. Each application should have a unique subschemas with which to work. After recovery, the data base administrator validates the subschemas organization to ensure that it is operating properly and allowing the application to receive only the data necessary to perform its tasks.

Systems programmers must be controlled in a slightly different manner than applications programmers. They must have the freedom to perform their tasks but be constrained from altering production programs or system utility programs in a fraudulent manner.

**End Users.**

End users are defined as all organizational members not included in the previous categories who need to interact with the data base through DBMS utilities or application programs. Data elements of the data base generally originate from end users.

Each data element should be assigned to an end user. The end user is then responsible for defining the element's access and security rules. Every other user who wishes to use this data element must confer with the responsible end user. If access is granted, the data base administrator must implement any restrictions placed on the request through the DBMS.

Assigning ownership of specific data elements to end users discourages the corruption of data elements, thereby enhancing data base integrity. Reviewers should ensure that this process exists and is appropriately reinstated after the recovery process has been completed and operational approval has been provided by the data base administrator.

After recovery, the data base administrator should ensure that all forms of security practices and procedures are reinstated. These processes are a part of data base security.

**Recommended Course of Action**

Review of DBMS recovery is crucial to ensuring the integrity of a corporate information system. To adequately assess this complex issue, a review plan should be developed and procedures should be established for conducting the review. The procedures may involve the use of checklists or automated tools and techniques. The sample review plan and review procedures in Appendix A and the backup and recovery facilities checklist in
Appendix B are designed to give guidance in developing these key documents and tailoring them to the organization.

**Author Biographies**

Frederick Gallegos
Frederick Gallegos, CISA, CDE, is an adjunct professor and advisor for the computer information systems department at California State Polytechnic University, Pomona CA. He has been active in the EDP Auditors Association and Foundation for more than 20 years.

Daniel Manson
Daniel Manson, CISA, is an associate professor of computer information systems at California State Polytechnic University, Pomona CA. Previously, he worked in EDP auditing, software quality assurance, and systems development at Caldwell Banker, Avery International, and McDonnell Douglas.