EDP ANALYZER
© 1976 by Canning Publications, Inc.

NOVEMBER, 1976
VOL. 14, NO. 11

RECOVERY IN DATA BASE SYSTEMS

With batch-type application systems using files on magnetic tape, recovery from failures has not been conceptually difficult. True, some of the actual recoveries have been difficult, such as when, by mistake, the only backup copy of a file was destroyed. But in theory, recovery involved going back to the prior generation of the master file and re-running the work. However, with the arrival of data base systems—and particularly on-line data base systems—this rerun concept is far from adequate. The users of on-line systems are annoyed if the systems are down for any length of time. So new methodology has been developed to aid in recovering rapidly after failures occur. Based on user experiences, here is what you had best look for in the way of recovery facilities when selecting your next data base management system.

The Weyerhaeuser Company, with headquarters in Tacoma, Washington, is a leading timber grower and timber products manufacturer. Annual sales are over $2.5 billion, and the company employs about 49,000 people.

Weyerhaeuser installed an on-line order entry system for its wood products division in 1964, based on the use of a GE 235 computer. Data files were managed by the IDS data base management system. In 1966, in preparation for installing a GE 635 system, the company entered into a joint project with GE to develop their own operating system—WEYCOS, using the GECOS II operating system as a starting point. WEYCOS was designed to handle both transaction processing and batch processing.

While the system is the on-line type, it is not interactive. Turnaround time averages between one and two hours, from the arrival of an order until acknowledgement. Even so, fast system recovery after a failure or data error is important, so Weyerhaeuser has built data base recovery and system restart facilities in WEYCOS. The system processes about 3,000 jobs per day, with peak loads of some 500 jobs per hour.

There are two types of data base problems for which recovery is needed—hardware failures and software failures. In the case of hardware failures, such as a disk surface being partially or completely unreadable, the need exists to recreate the unreadable records. For software failures, such as caused by a partially processed transaction being aborted, the need is to undo the changes in the data base that the transaction has already caused.

The components of the recovery system are as follows. A *Before image log tape* is used to record a copy of the record immediately before it is changed along with a reference code and function code. A start-of-job indicator is also recorded on this tape, as the job starts. An *After image log tape* is used to record an image of the whole page containing the record, after the record has been changed and has been written successfully to disk. A *data base dump tape* is created by dumping disk contents to tape every night. Finally, *end-of-day procedures* are used to capture the status of important system files, used for restarting as well as for recovery.

Weyerhaeuser people described for us the three main types of recovery.

Recovery from hardware failure. Under this condition, the computer cannot read one or more pages of data from the disk storage. The system makes several attempts to read the data. When such attempts prove unsuccessful, the recovery system is invoked. The old After image file is closed and a new one is started by mounting a new tape. The recovery system then searches the old After image tape for the latest image of the unreadable page(s). Upon locating it, it is read into core and then the system attempts to write it to disk. If it cannot be successfully written, then the system stores it "permanently" in memory until the disk can be fixed. Fixed head disks are used, so the amount of spare disk space is limited.

Once a page has been retrieved from storage, WEYCOS keeps it in core as long as possible, in case it is wanted by other programs. When an After image has been retrieved, as just discussed, access to it by other programs is prevented until the disposition of the page has been settled.

Should a large amount of data be unreadable, it is necessary to restore the data base by going back to the file dump of last night and update it by means of the After images.

Recovery from user program aborts. When a partially processed transaction is aborted, the recovery routines take control and lock the data base, plus restricting job selection. The Before image file is closed, and is then read backward toward the start-of-job indicator for that job. Using the Before images, the data base records are restored to their condition prior to running the job that aborted. The Before image tape is then repositioned to the point where it was just closed, and the data base lock and job selection controls are released.

When a page is accessed for updating, access to that page by other programs is prevented by locking. This locking is used to prevent both concurrent updating and deadlock. Concurrent updating means that two transactions are updating the same record at the same time; one of those updates is destroyed when the other one is stored. The deadlock situation arises when program 1 has record A and needs record B, while at the same time program 2 has record B and needs record A, so that neither can progress.

WEYCOS prevents concurrent updating by automatically delaying the processing of the second transaction until the first has been completed. And when WEYCOS detects a deadlock situation, it aborts one job, restores that program's Before images to the data base, unlocks the records, automatically reschedules the job, and proceeds with processing the second job. Weyerhaeuser encounters about ten deadlock situations a day. The protection technique that is used allows several programs to be updating the data base simultaneously as long as they do not conflict with one another.

Recovery from system aborts. When the whole system goes down, for any of several reasons (power failure, hardware failure, etc.), as soon as it is operating again, the recovery routines automatically close the Before image log after the last record that had been successfully written. Then, by processing the system files, it determines which jobs were not completed but had changed the data base, at the time of failure. (WEYCOS extends the job completion point beyond the normal program termination in order to safeguard both the data base and the output produced by the job.) A "rollback" list is built, for the jobs requiring recovery. The data base is locked for all user programs until recovery is completed. Then a rollback recovery, as described above for user program aborts, is performed for all jobs on the list. The Before image tape is then repositioned to the point where it was just closed and the data base lockout is released.

Time to recover from hardware failures, user program aborts, and deadlock is about 30 seconds, we were told. Of course, if a large area of the data base has to be rebuilt, that would take much longer.

System restart. Space does not allow us to go into the subject of restarting the on-line system in this report. Weyerhaeuser's restart procedures have been designed to insure that no input messages are lost or duplicated, that no output messages are lost or duplicated, and that any data base errors are corrected. For correcting errors in the data base, the recovery procedure for system aborts is used. This restart procedure takes about 15 minutes and must be done about 100 times per year.

Weyerhaeuser Company is well satisfied with their data base recovery capabilities. The company has not yet found another hardware/operating system combination that offers the capabilites they already have, so they continue to support and enhance WEYCOS.

Blue Cross of Southern California

Blue Cross of Southern California, with headquarters in Los Angeles, is a large health care prepayment organization. It has some two million members in Southern California and employs about 2700 people.

Blue Cross of Southern California uses an IBM 370/168 Model 3 with over 400 on-line terminals connected to it. In the spring of 1974, the company installed IMS/VS for managing the on-line data base, which consists of some 5.2 billion characters of data. The on-line system operates 11 hours per day for both update and query, and four hours per evening for query only. Batch processing is done during the same four hours in the evening.

The IMS/VS data base recovery utilities which are used by Blue Cross are as follows. *Create image copy* is a utility to dump all or part of a data base to tape, for backup and recovery purposes. *Restore image copy* is a program to restore a copy of the data base to the condition as of time of backup. *Create system log* is a routine for logging all changes made to the data base, including Before and After images of changed records, as well as checkpoint information. Logging occurs in both the on-line and batch environments. *Change accumulation* is a program for processing the system log file and selecting the latest changes or final image of each modified record in the data base. *Data base recovery* is a program for rebuilding a data base using the previously created image copy, the net accumulated changes for that data base, and the log tapes created since the last execution of the change accumulation utility.

In addition, IMS/VS has some other recovery utilities not yet being used by Blue Cross. One such is a log directory facility, for keeping track of all log tapes and what information is recorded on them. Blue Cross is looking forward to using this utility in the near future. The problem of keeping track of log tapes can be appreciated from the fact that, with 1600 bits per inch recording density, Blue Cross used to require 9 tapes a day for the on-line system log purposes. Now, using 6250 bpi, three tapes a day will suffice for this system. In addition, between 12 and 22 log tapes are used each day for the batch applications.

Blue Cross made one addition to the IMS recovery utilities. Should an on-line system abort occur and normal recovery facilities fail, a routine was needed to scan the system log tape to find out just which programs, if any, were active at the time of failure, so as to perform a backward recovery only for those programs. Otherwise, all data bases might have to be rebuilt from the latest image copies, using forward recovery, to insure integrity.

The most common type of failure, the people at Blue Cross told us, is when on-line software aborts. This type of failure accounts for about 85% of all failures, and occurs an average of about once per day. (Actually, the frequency is quite variable, ranging from several times a day to perhaps once per week.) This failure is the so-called "normal ABEND." Upon detecting an ABEND, the system takes control, performs an orderly termination of the job, and closes the log tape. The backout recovery utility reads the system log tape backwards, selects the Before images for this particular job back to start-of-job or to a more recent checkpoint, and restores the associated data base records. Recovery from such an ABEND takes about 12 to 15 minutes.

Sometimes recovery is not so straight forward. A complete system abort, perhaps due to a temporary power failure or such, takes longer. In such instances, an orderly termination may not have occurred, so operator intervention is needed to terminate the log tape and initiate recovery. IMS provides a log termination utility to complete and close an unterminated log tape. These "somewhat more difficult" failures account for about 13% of the overall failures, and can take up to 45 minutes for recovery.

The remaining 1% or 2% of the failures are those cases where a whole data base has to be rebuilt. In such cases, it is necessary to go back to the latest image copy and apply all After images that have occurred in the interim. Such failures might occur a few times a year. It can take 4 to 6 hours to effect recovery of a single data base.

One critical component of the recovery system is the system log tape—or, more properly, the multiple log tapes. If the log tape cannot be read, recovery becomes much more difficult. We asked about this possibility. Only once since the recovery system was installed had the log tape been partially unreadable, we were told. However, the operators were able to print out a good part of the partially unreadable blocks, and could determine what information was in those blocks. Since the log tapes have a variety of types of information recorded on them—start of program indicators, end of program indicators, Before images, After images, etc.—there is a good chance that an unreadable block does not contain information bearing on the recovery that is in progress. Luckily, that is just what happened in this particular instance, so recovery was not affected by the defect in the log tape.

The data base recovery facilities offered by IMS/VS have been successful in keeping their on-line system running, the people at Blue Cross of Southern California told us. Further, they expect to use additional recent enhancements to the recovery system in the not-distant future.

The new environment

D. C. Lundberg of IBM discussed with us some of the recovery concepts found in IBM's IMS. Also, R. D. Pratt and P. G. Powell of UNIVAC described recovery concepts being implemented in UNIVAC's IMS-90 and DMS-90 systems. We are indebted to these men for the following information on the new data base recovery environment.

Two types of work units occur in a data base system—transaction or message oriented work units and batch (run unit) work units. In a transaction oriented system, each transaction is treated as a different job or work unit. In a batch oriented system, a group of transactions is treated as a job, a run unit, or work unit. These two types of work units may be present concurrently in the workload. Transaction oriented work units generally involve update-in-place for on-line files. Batch run units may involve update-in-place files or the familiar regeneration of sequential files, as in magnetic tape processing.

The two types of work units require somewhat different—but related—recovery facilities. If the computer must process both types of work units concurrently, then both types of recovery facilities are needed.

Isolation of the error. With batch processing of sequential files, application system designers have had two options for the recovery from errors. One option has been to go back to the beginning of the job and start over. The other option has been to insert checkpoints at one or more intermediate points within the job. At a checkpoint, the status of the system is recorded on a log file or checkpoint file. To recover from an error, it has generally been necessary to go back only to the most recent checkpoint. One approach or the other was chosen based on minimum cost considerations.

With on-line systems closely coupled to the organization's operations, there are other factors to consider beyond the financial one. It is not desirable to stop all processing for an extended period while a complete rerun of a lengthy job occurs. Also, with data bases in which numerous applications share the same data, confusion can result from a rerun. If the last 15 minutes of work of job A must be rerun, then it may be necessary to rerun the last 15 minutes of work of *all* jobs, if they all could have been accessing the same records.

So in data base/data communications systems, it is desirable to isolate each work unit so as to inhibit the impact of an error as much as possible. This means that checkpoints or synchronization points should occur quite frequently—certainly at the beginning and end of each work unit, as well as at intermediate points for run units. All processing may have to stop while a recovery operation is underway (although proper run unit isolation would not require other run units to be interrupted). But the recovery should be accomplished as quickly as possible—get the errors out of the data base and either fix the offending program or get it out of the active job mix.

Two general recovery situations. We have mentioned these earlier in this report and will not repeat the discussion in detail. One situation is where physical damage has occurred to all or part of the storage media and the data recorded thereon cannot be read. The other situation is where a partially completed work unit is aborted, and the changes made to the data base by that work unit have to be backed out.

Two general types of recovery. Here, too, we have discussed the types and will not repeat in detail. Forward, or long, recovery is used where physical damage has occurred to the storage media. It uses an old version of the data base (a data base dump) plus a file of After images. The dump copy is brought up to date by means of the After images.

The other type of recovery is backward recovery, which in turn can be sub-divided into off-line backward recovery and quick (or dynamic) backward recovery. In either of these cases, the storage media are not damaged; what is desired is to back out the changes made by partially completed work unit(s). Backward recovery uses a current version of the data base plus a file of Before images. By replacing current records with their "before" images, the effects of updating are removed.

For transaction work units, it is feasible to back out just the one transaction. For quick recovery in such systems, an on-line utility and a special, fast access Before image file are used.

For batch run units, where there are multiple transactions in the batch, the need is to back out the effects of only the failing transaction, while retaining the effects of the other transactions. This recovery is usually done by an off-line utility using the system log tape.

As we will see, there are a number of complications associated with these approaches to recovery. Not all of the problems have yet been solved. Even where solutions have been developed, not all have been implemented in some of today's operating systems and data base management systems.

This report is concerned with recovery in a data base environment. There are a number of related topics that fall outside the scope of this discussion. For instance, we will not discuss the restarting of an on-line system, or how to avoid the possible loss or duplication of input or output messages, or the question of backup of hardware and communications services. We expect to treat such questions in future issues.

Let us now consider the types of system faults that lead to the need for recovery facilities.

Types of failures

Gibbons (Reference 1) differentiates *faults* from *failures*. A fault, he says, is a malfunction in a hardware, software, or human component of a system that leads to the introduction of errors. In due course, the error(s) may lead to a failure, which is the cessation of normal, timely operation of the system or the delivery of incorrect output data. Sometimes the fault and the failure occur simultaneously. In other situations, the fault may introduce errors for an extended period before the failure is detected.

Following are the major types of faults identified by Gibbons:

TYPES OF FAULTS

1. Hardware malfunctions, covering all hardware elements in the system.
2. Application program faults, where coding errors tend to increase with the size and complexity of the program.
3. System software faults, including such aspects as a poorly designed priority system, the checkerboarding of memory, and the allowance of deadlock situations.
4. Data errors, due to inadequate validation of input data.
5. Design miscalculations, leading to improper operation during peak load periods (overloads).
6. Operator mistakes, such as mounting the wrong generation of files or selecting the wrong version of a program.

All of these can cause failures, either immediate or delayed, from which recovery must be made. Gibbons sees the following types of consequent failures.

TYPES OF FAILURES

1. Machine failures—some hardware components are essential for continued operation, such as the CPU; there is no option other than to fix them in order to recover or provide dual CPUs. In other cases, such as tape units, it may be possible to substitute another unit and continue operating, perhaps in a degraded mode of service.
2. Application program failures—these are usually caused by an application program attempting to perform an invalid operation, in which case control is transferred to the operating system.
3. Operating system failures—these can be caused by errors in the operating system code itself (most new releases have new errors in them) or because an application program has given wrong parameters to the operating system for, say, opening a file.
4. Data validity failures—data in the data base is invalid so that an application program fails or notifies the operating system.
5. Data base access failures—a record is unavailable, due to parity error, invalid key, etc.; or a record is missing or is invalid; or the request is invalid, such as attempting to write an unchanged record; or deadlock may occur.
6. Overload failures—when an overload occurs, the system may stop or performance may begin to deteriorate badly; memory space may be unavailable due to checkerboarding; control tables, temporary working areas, message queues, etc. may overflow their limits.

There is another way in which failures can be classified, as follows.

One or more work units abnormally terminate. Such failures occur when a user job is cancelled, when an I/O operation fails, in deadlock situations, and so on. An orderly termination of processing is possible.

Data base is contaminated. A program error may cause a spreading data error in the data base—such as an inventory program that adds issues into stock rather than subtracting them. Such program errors may be deliberate and malicious. Once the erroneous situation is detected, an orderly termination of processing is possible.

All processing stops suddenly. Examples of such failures are power failures, CPU failures, and operating system failures. Generally, no orderly termination of processing is possible.

Overload condition occurs. As indicated above, examples include message queues exceeding their limits, memory space is unavailable, etc. The failure can lead to loss of data and even to complete system failure. It may not be possible to have an orderly termination of processing.

Recovery operations proceed more smoothly if an orderly termination can be made when the failure is detected. Gibbons says that the operating system (1) should finish any data base requests and note those that are still incomplete, (2) should force each application program to finish and then shut down, (3) should send notifying messages to remote terminals, (4) should create a checkpoint, (5) should dump information for a post mortem analysis, and finally (6) should terminate the operation.

So these, then, are the types of situations in which recovery is needed.

Elements of a recovery system

Lundberg of IBM and Pratt and Powell of UNIVAC described for us the elements that make up a recovery system. These are:

- File dump facility
- System log facility
- Log scan facility
- Checkpoint facility
- Control of concurrent update and deadlock
- Forward recovery facility
- Rollback facility
- Recovery directory
- Broadcast facility

We will discuss each of these elements briefly. We will also draw on the works of several authors, particularly Gibbons.

File dump facility

With sequential files on magnetic tape—and in some cases, sequential files on disk packs—the file dump facility has been automatic. All that need be done has been to save two or three prior generations of a file.

With update-in-place files, explicit action is needed to create copies of the files at specified points in time. The usual process is to dump the files, or selected portions of files, from disk storage to magnetic tape periodically.

There are evident problems with file dumping. For one thing, it consumes resources, such as channel capacity. It may interfere with or completely stop all other processing, depending upon the design of the computer system.

For another thing, it can take appreciable amounts of time to dump large data bases. Gibbons (Reference 1) points out that even with a 1.25 million bytes per second (mbps) transfer rate for magnetic tape (the fastest of today's tape units, as far as we know) and an 800 kbps transfer rate for disk storage, the effective dump rate may be only about 330 kbps. At this rate, it would take about 50 minutes to dump a one billion character data base, and about eight hours to dump a 10 billion character one.

With very large data bases, two main approaches to dumping are used. One is to dump some small fraction of the data base each night—a small amount so that not too much time is consumed in the dumping. Another approach is to provide parallel paths for dumping; this requires less time but is much more expensive in hardware.

If small amounts of the data base are dumped each night, it is evident that good control of these dump copies will be needed. Since several generations of dumps may be needed, a good number of tapes may be involved.

System log facility

Lundberg of IBM says that the system log function should contain records of all input and output messages, the origin and destination of all transactions, identification of units of work along with their start and completion, all data base changes, checkpoint information, and records of all processing activities.

Gibbons (Reference 1) qualifies this list somewhat. He feels that the system log should contain records of all *valid* incoming transactions. Also, in addition to the Before and After images, the log should show the file to which each belongs, the program that updated it, the updating transaction, and the time.

Eastin (Reference 3) adds the following to the list. The log should contain messages of hardware failures, communication startup and shutdown details, message header inconsistencies, and all unusual occurrences.

Palmer (Reference 2) sees the possible need for a second log file. During a recovery operation, he says, the first log file is in use for recovery. If the remainder of the system is to be kept in operation, a second log file will be needed.

Lundberg points out that duplicate log tapes may be desired, just to make sure that the log tapes are readable—but these dual tapes are logically one file. A recent version of IMS provides a dual logging option, along with a utility for creating a new log tape from the combination of the dual logs, should both of those have errors in them.

It should be apparent from the above discussion that the system log file will have lots of data recorded on it, including a wide variety of records types. One of the problems will be the number of log tapes that are produced. If the log tapes are recorded at 1600 bpi density, it will not be unusual for installations to accumulate log tapes at the rate of from 10 to 30 tapes per day.

Another aspect of the system log file should be singled out—the aspect pertaining to quick recovery. Dynamic or quick recovery requires a file of Before images, start of job, and end of job information stored on a fast access device. Logically, each job could have its own file, achieved by tying together all records for that job by means of chains or indexes. Alternatively, each job could have its own distinct physical file. Powell of UNIVAC pointed out to us that all updates between two successive checkpoints for a job are logically related. Once a checkpoint for that job has been performed, the quick Before images can be erased. Since the exclusive use locks are released at the time a checkpoint is recorded, any Before images recorded before the checkpoint can no longer be used for quick recovery.

Log scan facility

From the above discussion, it can be seen that the regular system log file contains a wide variety, and probably a large number, of records. The records are intermixed. Further, many of the records will be "obsolete" in the sense that if three successive updates occur to the same record, only the third After image may need to be retained.

So a facility is needed for processing the system log file, in an off-line manner, in order to select the appropriate information and to make it conveniently available for a recovery operation. In IBM's IMS/VS terminology, this is called the "change accumulation" utility.

What about the old Before and After images on the system log, after a recovery has been accomplished? IMS/VS obviates concern for these images by logging a backward recovery and by creating new record image copies in the case of forward recovery. These records are then used, if needed, in any future recovery.

Checkpoint facility

When to checkpoint. As indicated earlier, today's concepts of recovery require that each work unit be isolated from other work units, so that the changes made by an aborted work unit can be removed from the data base. This requirement applies to both transaction work units and batch run units.

Lundberg of IBM says that checkpointing should be done (1) at the work unit start time, (2) at the work unit termination time, and (3) at any user-requested intermediate points. These intermediate checkpoints are needed in lengthy batch run units.

In a number of today's installations, both on-line and batch jobs are being run concurrently. Also, most of today's recovery systems, as we understand them, do in fact stop all processing while a recovery is underway. So intermediate checkpoints are needed for lengthy batch jobs, not just to reduce the recovery time for those jobs but also to reduce the time that the on-line system is down awaiting the completion of the recovery operation.

Lundberg points out another complication, along with the solution used in IMS/VS. If it is desired to run a batch job against a data base used by an on-line system, two monitors could be involved—the on-line monitor and the batch monitor. Since these two monitors might work at odds with each other, undesired actions could occur. IMS/VS provides "batch message processing" to solve this problem, wherein the batch program goes to the on-line monitor for all data base accesses.

Gibbons (Reference 1) lists some of the conventional criteria for determining when to checkpoint. These include: (1) at specified time intervals, such as every 10 minutes, (2) after X transactions have been processed, (3) when the processing load is light, to minimize the interruption caused by checkpointing, (4) before a critical event such as file dumping, (5) at the request of an application program, and (6) at the request of a system console operator. Note that these criteria do not directly meet the need of isolating each work unit.

Browne and Lasseter (Reference 3) have developed a quantitative model of recovery that relates the resources used in checkpointing versus the costs of complete reprocessing. This model helps determine the frequency of checkpointing from a cost standpoint.

What to checkpoint. Checkpointing (sometimes called synchronization) means recording the essential details of the status of the system at a point in time. It indicates what programs are active at that point in time. It records the status of any variables that carry across checkpoints, such as the current status of control totals, hash totals, and accounting information. If a work unit uses one or more sequential files, it records the current position of each one of those files.

Gibbons (Reference 1) lists a number of other types of information that perhaps should be included in a checkpoint. These include the current system and network configuration, pointers to message queues (by message type, by terminal, and by priority), the current message sequence number for each terminal, current data base information such as bad areas that are locked out, and the status of temporary working files.

The determining factor of what information to record for a checkpoint is: what is the necessary and sufficient information for achieving a recovery, for the types of recovery that are planned? The above brief discussion gives some idea of what must be considered in selecting the checkpointing information. True, the user may not have much influence on what checkpointing information is saved by a packaged recovery system—but an analysis of this type will indicate just how effective the recovery system is likely to be.

Control of concurrent update and deadlock

Lundberg of IBM pointed out to us that if the operating system does not have controls to prevent concurrent update and deadlock, and facilities to cure those conditions when they are encountered, then the recovery system is incomplete.

Here is the way that IBM's IMS/VS handles these situations. When a user program makes a data base call for a record, the *request* is put into a control queue. When the record has been retrieved, the request is assigned control level 1, which means "read only" use. Access to the record by other read-only user programs is not prevented and the record need not be written back to the data base when the user program no longer needs it.

Access that may result in an update is so defined at retrieval time. Two potential update users may not retrieve the same record. The second requestor is automatically held in abeyance by the system.

As soon as a user program changes a record, the request in the control queue is assigned control level 2. This means that the record has been updated and belongs exclusively to the updating program. Access to it by other user programs is prevented, and it will have to be written back to the data base. Exclusive control is maintained until a synchronization point is reached and dynamic recovery is no longer a possibility. When that user program begins *another* transaction, it is assumed that the program is finished with all records held in exclusive control by the previous transaction. Those records are then written back to the data base and access to them is unlocked.

Powell of UNIVAC described the approach used in DMS-90. When a change is made to a page of data, that page is implicitly locked until (1) the end of the run unit, or (2) a user-specified intermediate release point. A page that is retrieved but not updated can be explicitly locked by a run unit to preserve its current state, in order to insure the consistency of a future update affecting multiple pages. A page lock prevents other run units from either retrieving or updating the page. Once an inter-related series of updates has been performed (either a transaction work unit or a segment of a batch work unit), the run unit can execute a command to release all outstanding locks. This command also establishes a data base recovery checkpoint and, optionally, a batch application program checkpoint. The locking and recovery features of the system are closely inter-related.

Similarly, deadlock situations can be detected by processing the request queue. If program 1 has record A and needs record B, and program 2 has record B and needs record A, this situation can and is being detected by such controls. The operating system then aborts one of the jobs, performs a quick recovery on it, and reschedules it.

As we indicated earlier, the Weyerhaeuser Company encounters about ten deadlock situations a day—so it is a non trivial problem.

Forward recovery facility

This recovery procedure is used when physical damage has occurred to the storage media, such as disk surface damage from head scraping. The concept here is somewhat similar to the rerun concept in magnetic tape processing.

The damaged area is first identified. Then the latest copy of the file dump for that area is retrieved. The system log tapes applying to the period from that file dump to the present are retrieved. All but the current one of these may have been processed by the log file scan facility, so that all of the After images are organized for easy retrieval. The After images applying to the damaged area, starting immediately after the file dump time, are then applied to the file dump. The data base area is now up to date, and would be stored in an alternate storage area.

Several potential problems are evident. If the dump copy of the file is not readable, it may be necessary to go back to the next previous dump point—which means there may be a need for multiple generations of file dumps. Or the system log tape may not be readable. If the system log tape has not been duplicated and if some of the essential information is unreadable, it will be necessary to manually determine just what the missing information is.

IMS/VS provides a facility for individual (disk file) track recovery. When track damage is detected, the system creates search criteria for finding all records pertaining to the track in error. So only the track is recovered, not some larger portion of the data base.

Rollback facility

This recovery procedure is used when partially completed work units are aborted. The changes made to the data base by those work units have to be backed out.

In theory, this rollback or backout type of recovery does not appear complex. We have described it several times in this report. The appropriate Before images are used to step the data base back to the condition it was in at the time of the last checkpoint for that work unit. And then the system is restarted.

Pratt of UNIVAC has pointed out to us, though, that rollback has its fair share of controversy. The basic question is: how far back should the rollback process be allowed to go? This is still an unresolved question, to his mind. Some people argue that rollback should go back only to the start of the current work unit. The reason here is that other user programs may subsequently update the same records—and rollback erases the effects of those updates. If rollback is allowed only to the most recent checkpoint, the system's lock mechanism can assure that the records have not been used by other user programs. Other people argue that the decision should be left up to the user. Just by the nature of the system operation, the likelihood of updates by other user programs may be very small. In such a case, the user may want to perform rollback "to 8 PM last night"—and should be allowed to do so, in the view of these pragmatists.

Lundberg of IBM observed that the data base is the property of all users and hence rollback decisions should not be left to the user. There is one exception to this observation, he said. In the case of an on-line transaction, the user should be able to roll back a single transaction, thus eliminating the transaction and all of its effects. However, if an ABEND occurs, the user should have no options; the system should define the rollback process.

Rollback does have the potential for erasing updates that should not be erased. The threat here should be recognized and user policies adopted to minimize the threat.

Recovery directory

Powell of UNIVAC pointed out to us that the many types of information used in recovery operations—file dumps, system log tapes, quick recovery files, and checkpoint data—make recovery a complex process. In our discussion, we have mentioned the need for multiple generations of file dumps, as well as many (and perhaps duplicated) reels of system log tapes. We mentioned that during a forward recovery, a new After image log tape may be mounted while the old one is used in the recovery process.

So the question arises: how can the computer system operators perform the recovery function without error, in such a complex situation?

Powell argues for the use of a recovery directory, to control the operation and avoid errors. The recovery directory is essentially an inventory of all recovery data—what it is and where it is located. It identifies all recovery records by type, by time period, and by location. It contains a history of all file dumps by area of data base, with the points in time at which they were taken, and where they are located. It has a history of all log tapes, the time periods they cover, and where they are located. It holds a listing of all run units active at any point in time against any data base in the system. And it provides a cross reference between file dumps and log tapes, for each area of the data base.

With a directory of this type, says Powell, the selection of the data to support the recovery process can become more automatic, with less of the operation subject to operator errors.

Broadcast facility

When an on-line system goes down, data processing management certainly does not want remote users to begin calling in on the telephone to find out what is wrong. Such calls would swamp the data center.

What *is* desired is an optional means for notifying users that a trouble has occurred and how long it is expected that repair will take.

Depending on what part of the system has gone down, the system might send a message or signal to remote terminals indicating that a recovery operation is underway. Another solution, for use in lengthy downtimes, is to provide a special telephone number to call which is answered with a recorded message.

When the system is operating again, it may be desirable to notify each remote user that was active at the time of failure just what the status of his work is.

Some complexities of recovery

Palmer (Reference 2) lists some of the ways in which recovery operations can become more complex than they normally are. These ways include (1) the failure of a second work unit while the recovery of a first work unit is underway, (2) the failure of the recovery utility itself while recovery is underway, and (3) a failure while a data base reorganization is underway (resetting pointers and indexes).

Gibbons (Reference 1) points out that recovery is complicated when two or more failures occur in rapid succession. Moreover, this is not just a theoretical possibility, he says; it is not unusual for a series of failures to occur in a short period of time.

Lundberg pointed out to us that the application programmer can make the recovery operation more complicated if he allows processing activities to begin before a checkpoint and end after a checkpoint. This situation is more likely to occur for intermediate checkpoints made during a long run unit.

Pratt mentioned to us that, during the recovery operation for an on-line system, it is important that the system let the terminal operator know just what has happened and how far back in time recovery has gone. The terminal operator must know what he has to do next and what he has to do over. The situation can get particularly messy, says Pratt, if an input or an output message has been lost, when a system restart is performed along with a recovery. Communication between the terminal operator and the system can become badly snarled by such an event.

There are a number of complicating situations that have been discussed earlier in this report. We will not repeat those discussions but will simply list the situations to indicate the scope of possible complexities.

If a system abort occurs, an automatic orderly termination is not possible. Not only must all loose ends be cleaned up as a first step in recovery but also all jobs active at the time of the failure will require backward recovery. If updates to the same record can be made by multiple user programs since the last checkpoint of one user program, backward recovery runs the risk of erasing legitimate updates. A recovery system that does not prevent and cure concurrent updates and deadlocks is not a complete recovery system. And if any of the recovery information media are unreadable—file dump tapes, system log tapes, and so on—then recovery can become much more difficult.

It is clear that recovery of data base systems is a complex operation.

What of the future?

The trend for the future of data base recovery systems is to make them even more automatic and dynamic than they are today. The goal is to take over essentially all operator functions, leaving only manual functions such as mounting tapes. As Palmer says (Reference 2), the recovery system should assess the damage, perform the recovery, and advise users of the possible effects of the recovery. None of today's systems are able to do this much, although some leave little for the human operators to do.

It may be possible for recovery to affect only the failing work unit—plus, of course, the records that are being restored and hence cannot be accessed by other work units. Other work units should be able to continue operation uninterrupted.

Pratt commented to us on work that some people in the field are doing. Each user of an online system triggers a hierarchy of actions, he says—programs are called, data records are accessed and modified, the data base organization may be changed (pointers and indexes amended, etc.). The work unit concept has a different granularity at each level. On any single level, a work unit consists of the sequence of actions required to implement a single action at a higher level. The operating system should have knowledge of this hierarchy and be able to determine just what has been done so far. If a failure occurs, the recovery system would then be able to determine just how far to roll back. Failure at a lower level results in a shorter rollback operation.

Lundberg observed that IMS/VS knows the progress of each transaction during its entire life in the system; a transaction is logged six to eight times as it passes through the system. So the system has extensive information upon which to base action. In addition, he sees future progress in several other areas of recovery system design. One is dynamic checkpoint and file dumping, so that these can proceed automatically and without interrupting normal processing. Another is the partitioning of data bases. And another area is dynamic forward recovery, making use of the partitions of the data base.

So, as good as some of today's recovery systems may be, there is still substantial room for enhancements.

REFERENCES AND ADDITIONAL READING

1. Gibbins, T., *Integrity and recovery in computer systems*, NCC Publications (National Computing Centre, Oxford Road, Manchester M1 7ED, U.K.) or order from Hayden Book Company, P.O. Box 978, Edison, N.J. 08817, $9.95; published 1976.
2. Palmer, I., *Data base systems: A practical reference*, Q.E.D. Information Sciences Inc. (141 Linden Street, Wellesley, Mass. 02181), 1975, price $29.50 prepaid.
3. Eastin, C. P., "System and software controls for on-line systems," *Management Controls* (Peat, Marwick, Mitchell & Co., 345 Park Avenue, New York, N.Y. 10022), June 1972, p. 141-145.
4. Oppenheimer, G. and K. P. Clancy, "Considerations for software protection and recovery from hardware failures in a multiaccess, multiprogramming, single processor system," *Proceedings of 1968 FJCC* (AFIPS Press, 210 Summit Avenue, Montvale, N.J. 07645) p. 29-37; price (microfiche) $20.
5. Browne, J. C. and G. L. Lasseter, "An optimizable model for application of rollback/restart/recovery procedures for large data bases," *Proceedings of International Conference on Very Large Data Bases* (order from ACM, 1133 Avenue of the Americas, New York, N.Y. 10036), 1975, price $15 prepaid. This is a summary of the paper. For further information on the complete paper, write Prof. J. C. Browne, Department of Computer Sciences, University of Texas, Austin, Texas 78712.
6. Tonik, A. B., "Bibliography on checkpoint, restart, and recovery papers," *ACM Performance Evaluation Review* (ACM, 1133 Avenue of the Americas, New York, N.Y. 10036), April 1976, p. 100-104; price $2.50.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

PRICE SCHEDULE

The annual subscription price for EDP ANALYZER is $48. The two year price is $88 and the three year price is $120; postpaid surface delivery to the U.S., Canada, and Mexico. (Optional air mail delivery to Canada and Mexico available at extra cost.)

Subscriptions to other countries are: One year $60, two years, $112, and three years $156. These prices include AIR MAIL postage. All prices in U.S. dollars.

Attractive binders for holding 12 issues of EDP ANALYZER are available at $4.75. Californians please add 29¢ sales tax.

Because of the continuing demand for back issues, all previous reports are available. Price: $6 each (for U.S., Canada, and Mexico), and $7 elsewhere; includes air mail postage.

Reduced rates are in effect for multiple subscriptions and for multiple copies of back issues. Please write for rates.

Subscription agency orders limited to single copy, one-, two-, and three-year subscriptions only.

Send your order and check to:
EDP ANALYZER
Subscription Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-3233

Send editorial correspondence to:
EDP ANALYZER
Editorial Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-5900

Name______________________________

Company______________________________

Address______________________________

City, State, ZIP Code______________________________