To: MTB Distribution

From: M. Weaver and A. Bensoussan

Date: December 16, 1980

Subject: Multics Data Management: Problem statement.

Purpose

The Multics strength in data management is the existence of its relational data base manager, MRDS. The demand for relational data base managers is increasing every day and it is likely that in the years ahead they will become an absolute requirement for large systems.

Multics has a relational data base manager that is attractive and operational. However, the data management facility, as a whole, needs to be improved in order to meet the user's expectation. Improvements are needed not only in MRDS but also in those parts of the system that provide services to MRDS such as vfile, large files, concurrency control and recovery.

This memo describes the weaknesses of the Multics data management facility. Its purpose is to identify the most serious problems in order to address them soon by the Multics developers. The topics of discussion include:

- Recovery
- Concurrency control
- Support for large files
- Vfile
- MRDS

Recovery

The present system does not provide support for data base recovery in any of the following situations:

a. Disk damage due to a head crash or a system crash.

b. Main memory damage due to a power failure or ESD failure, even though there was no disk damage.

c. Incomplete data base operation due to a system crash or a process abort, even though there was no disk damage and no main memory damage.

d. Incomplete data base operation due to interference between concurrent processes, even though there was no disk damage, no main memory damage, no system crash and no process abort.

The Multics backup system is clearly not appropriate for data base recovery. What seems to be required is a journalization method, similar to that used in GCOS. A before image journal would keep track of the data before modification and would be used to roll back; an after image journal would keep track of the data after modification and would be used to roll forward after a disk damage.

Concurrency Control

The system does not provide adequate support to preserve the integrity of the data base in concurrent access. What is missing is the ability to:

a. Define atomic operations, also called "transactions". A transaction is an operation that must be done completely or not at all. If it has started and, for any reason, it cannot be completed, the system must undo it.

b. Execute concurrent transactions on the same data base with the guarantee that the system will detect any undesirable interference due to the concurrency and undo the started transactions that cannot be completed.

c. Integrate concurrency control with recovery so that, after any system failure, the data base can be restored to a consistent state that has no unfinished transaction.

What seems to be required is a standard protocol to tell the system when a transaction begins and when it has to be committed or aborted. The system would be responsible for keeping all the necessary information to undo a transaction that cannot be finished; it would also be responsible for detecting deadlocks and for keeping concurrent transactions from compromising the integrity of the data base. Using the standard protocol would ensure that the effect of concurrent transactions would be the same as if their execution was serialized.

File Implementation

Files are implemented by the juxtaposition of 256k segments created in the same directory. Although this implementation has unquestionable virtues in terms of making use of hierarchical levels of abstraction, it has introduced unacceptable limitations and overhead in creating and accessing large files due to the poor hardware support for large segments.

a. Limitation in file capacity. The size of a file is limited by the number of segments that can be created in a single directory. This number is around 1000, which limits the size of a file to 1000 times 256K words.

b. High overhead in accessing large files. This overhead comes mainly from the activation and deactivation mechanism associated with accessing segments in the Multics virtual memory. This overhead is growing with the size of the file and with the degree of random access to the file. It also comes from the existence and the maintenance of a directory entry, a VTOC entry, a KST entry and a Descriptor Segment entry for each segment.

c. Placement control. The data base administrator wants to have the ability to advise the system as to where a record or a set of records should go on the disk. This capability is non existent in the current system.

What seems to be required is an implementation of files out of pages in order to eliminate the limitations and overhead associated with segments. There is no intrinsic reason why segments should be visible in the implementation of a file. Pages of a file would be made accessible to the various programs through a buffer manager instead of segment control and page control. The file would be allocated on disk as several extents, allowing placement control by the selection of the appropriate extent.

vfile

The problems perceived with vfile are of various nature. From the user's point of view, the major complaint is the inability to recover when an indexed file has been damaged by a system failure. The file becomes unusable if one page of the index is damaged, even when all the rest of the file is undamaged.

From the MRDS system programmer's point of view, it is not clear whether or not vfile presents the right interface for the implementation of a relational data base manager.

From the vfile system programmer's point of view, the major complaint is that vfile is too complex and that it is very difficult to maintain, modify or extend.

If one addresses the problems in the areas of recovery, concurrency and large file implementation, it is clear that vfile will require a substantial amount of changes. The question is: Is it worth changing vfile or should it be rewitten or replaced by something else?

MRDS

The efficiency of MRDS is not acceptable to our users; it takes too many IO's and too much CPU time to perform the simplest data base operation.

Limitations built in MRDS are also not acceptable. The maximum number of relations is 128, the maximum number of tuples in a relation is 10,000,the maximum number of attributes allowed in a submodel is 200, the maximum number of data base openings allowed is 64.

Limitations built in the system will be hit very soon when users start having larger data bases. The size of a file is limited by the maximum number of components in an MSF, the number of components directly accessible is limited by the maximum number of known segments in a process, the number of active segments is limited by the size of the SST segment an cannot grow enough to avoid frequent activations and deactivations of MSF components.

The integrity of a data base can easily be compromised even when no data has been lost. A quit followed by a release, for example, may leave the data base inconsistent if the process was interrupted in the middle of an update.

Recovery procedure from process abort, system abort, ESD failure, and disk failure is nonexistent.