Rdb/VMS Support for Multi-media Databases

T.K. Rengarajan
Digital Equipment Company

Abstract

Rdb/VMS is Digital's premier database management system for production systems. Since version 1.0, Rdb/VMS has provided support for blobs, which can be used to store multi-media objects. However, storing large multi-media objects as blobs on magnetic disk is expensive due to the relatively high cost of storage for magnetic disks as compared to Write Once Read Many (WORM) devices.

With Rdb/VMS V4.1, it is now possible to manage large multi-media databases in a cost-effective manner with the new functionality to support WORM devices. The goal of WORM area support is to provide transaction semantics transparently for infrequently updated large objects stored on WORM devices. We had to solve a number of interesting problems associated with the "write once" nature of these devices before we could support multi-media databases. This talk provides a comprehensive overview of the factors involved in the design of Rdb/VMS WORM support.

The file system that manages the WORM device is assumed to provide the standard file operations like file creation, allocation and extension. Such operations involve updates of on-disk structures and hence involve waste of disk blocks. However, knowing that a certain database area is on a WORM device, we employ different algorithms to minimize space waste due to repeated updates of on-disk structures in the database.

Database records are classified into first-class and second-class records. Only second-class records are permitted in a WORM area. First-class records point to second-class records. Blobs are second class records and the rows of a relation constitute first-class records. Transactions are undone by undoing the first-class records on read-write media, pointing to blob segments in the WORM area. No undo records are written to the undo log for second-class records.

No data structures on database pages are updated in WORM areas. When a new segment is stored as part of a blob, the previous segment must be updated to point to the new segment. However, the previous segment may have been flushed to disk by the time such an update is required. Therefore, a new format for blobs has been designed to cope with this problem.

The storage algorithm for blobs in WORM areas is essentially sequential. Buffers from a WORM storage area are managed specially by the buffer manager to permit packing a number of segments on the same WORM page. The end of written area for each WORM area is controlled by a lock in the VAXcluster data-sharing environment. Pages of a WORM area are allocated to different users of the database for storing new blobs. When a page is so allocated to a user and then that user process aborts, that page remains unupdated. WORM holes are such pages in a WORM area that were never written due to transaction failure. Utilities like backup and verify have been updated to recognize WORM holes.

A set of WORM areas can be designated as the target for storing blobs in any column of a table. Among members of a set, either a random or a sequential algorithm can be selected as the store criterion. In addition, store failover is provided if the WORM area selected by the store criterion cannot be extended any more.