StorageTek®

APPLICATION NOTE

Using Lifecycle Director™ software for IBM DB2

Optimizing with database archival

DECEMBER 2004

Lifecycle Director™ software provides robust, row-level archival and transparent data retrieval requiring no application changes. Lifecycle Director software overcomes many of the problems organizations have had maintaining old and aging inactive DB2 data and helps improve DB2 performance.

In this article, we will explore the problems associated with DB2 data archiving. We will also look at past and current methods IT organizations have used to deal with these problems, and how these approaches differ from Lifecycle Director software. In particular, we will explain how this approach moves the concept of data storage management beyond HSM data archival into a more practical management scenario.

1. The problem

Managing exponential data growth with long retention

According to research from META Group, databases are growing at greater than 125 percent per year. In many organizations, the amount of database information that must be managed can double in as little as two to three years. As the volume of data grows, the problems associated with managing, protecting and using the information grow exponentially. This becomes most evident when performing routine management tasks such as reorganizations and backups.

Additionally, many organizations retain data for seven years or longer. Yet, the probability the data is used drops below 50 percent after the first month and to less than 5 percent by the third month. Still, all data is stored, managed and protected the same, using the same storage platform at the same cost.

Predetermining future storage demands

The growth problem is compounded by the fact that storage planners are required to allocate predefined set amounts of storage for their DB2 environment. This predetermination must consider anticipated future demands, and allocate ample space to accommodate that need. In many organizations, the amount of overcapacity that is pre-allocated — disks that contain no data — may be twice as much as current tables. Thus, a typical organization has increasing numbers of drives that are rapidly declining in actual value.

DB2 housekeeping

With the data continuing to mushroom in size, the task of reorganizing the data can become one that is deferred simply because the task requires too many system resources to complete in the scheduled maintenance window. All too often these common maintenance practices are not completed for the largest tables, which need it the most. The net result is that table structure does not get compacted and reorganized, and therefore provides less than optimal performance.

The backup window may also be especially time-sensitive. Although an organization may do frequent incremental backups, a full backup of these unprotected systems increases in importance over time. However, many organizations may forego backup because there simply is NOT enough time to take the system down for a full backup.

In a restore, the last full backup will be resurrected first, then each incremental backup will be applied, in the order in which they were made. A large data table could take hours — or even days — to rebuild. More significantly, the most recent data will most likely be the last data restored.

Regulatory compliance requirements

Regulatory compliance adds a few extra challenges to the fray. Regulations may dictate retention rules, automatic deletion or data to be store in unchangeable write once, read many (WORM) format. These requirements are often difficult to implement and costly to maintain.

Although the cost of disk drives has come down, the cost of managing the extra storage continues to climb. According to the Gartner Group, the cost of managing storage may be four to five times the cost of the drives.

Throw in the fact that many organizations have implemented Web interfaces, which allow nearly 24 x 7 access to the tables, and the window for backup, reorganization and general table maintenance becomes extremely limited. Additionally, with expectations of immediate access and rapid response to queries, a belief that no amount of new data would be too much for an existing table, and declining IT budgets, it becomes clear that a new approach to these challenges is needed.

2. Current and historical approaches

The most common approach has been to throw more hardware at the problem. As noted above, disk prices have been dropping. However, the increase in the amount of storage needed is outpacing the decline in disk prices. The net effect is that, strictly on a hardware cost basis, the drives required to handle the increased table sizes continue to cost an organization an increasing amount of money over time.

Further, with more data being stored on disk, the cost to maintain, tune, backup and restore the data on these disks continues to increase. The declining cost of disk drives may seem to ease the pain somewhat, but other issues related to the rapidly rising amount of data to be managed continue to be a major problem.

Hierarchical storage management (HSM) has been frequently touted as an approach to the problem. Although most managers are extremely familiar with the concepts behind HSM, it may be useful to point out that HSM is effective on a table by table basis. Thus, a table that has not been accessed since it was first created three years ago would correctly be archived to tape (or managed using whatever rules were created for the HSM system). However, a table that may have eight years worth of data, has been updated continuously and contains perhaps 10 percent active data, would not be subject to HSM management — although 90 percent of the records will be inactive, they will remain inside this active table. Thus, HSM was a useful concept for its time, and is still useful for managing fully inactive tables, but for tables that have any active data at all, it provides no benefit.

Third-party tools, available from a handful of vendors, solve the data problem by purging the old data, or moving it onto other data tables. Although purging will relieve the database of inactive data and reduce the size of the database, it can introduce larger problems caused from broken database relationships. This is especially true when the referential integrity of the database is maintained by the application and not the database.

When the data is purged using these tools, it is moved to a secondary database (which requires database licenses) and the application no longer has awareness of its existence. To resolve this issue, the application has to be modified to determine if the data has been archived and where to go get it, if requested. Alternatively, an archive browsing tool can be used to view the information. This tool is completely separate from the application and would need to be deployed to all users.

Custom applications can be developed in-house to resolve some of the problems noted earlier. In general, these measures only address a few applications, and are usually accomplished by purging data (or archiving it to other tables) or creating read-only copies. For example, in some organizations, each year, data more than one year old is moved to a new data table. Retrieving transactions for the past three years can pose a serious problem, because three or four tables must be opened and searched to complete a query. To accomplish this, additional coding is required. Application coding to enable searches of multiple databases can be fairly extensive — and may require annual modification. Referential integrity may be difficult to test and hard to fix if problems are detected. No wonder many organizations choose to live with overly large tables.

Most in-house or third-party approaches can create substantial problems. Perhaps the most significant problem is loss of referential integrity. If data is not purged with extreme care, this loss of referential integrity can cause orphaned data. Queries on such damaged data tables can result in applications aborting, because references within the table are no longer valid. The only way to safeguard against this peril is to carefully analyze the database relationships maintained in both the database and the application. This is a resource-intensive effort and can be a near-impossible task for aging applications where the people who developed the application are no longer in-house.

In order to implement either in-house or third-party approaches, extensive coding of applications is required. The cost of coding and testing the new applications is significant. Further, additional effort may be required to maintain these data tables, to evaluate and repair errors that may result from coding problems, and to resolve other issues that may be directly related to the additional coding necessary to make these changes.

With the exception of table-archived HSMs, nearly all database archiving solutions move the data from disk to disk. This may resolve some performance problems, but it does little to provide storage cost savings.

The current approaches, and the problems associated with them, should be clear. In the next section, we describe an approach that helps resolve most of these issues.

3. Lifecycle Director software solution

StorageTek® has a new approach to database archiving. This new approach, integrated into a product offering called Lifecycle Director software, uses two main components for resolving many of the issues noted earlier.

The first component, called Archive Manager, implements the logic necessary for managing access to, and maintenance of, data. It supports archive and retrieval functions and extensive data management functions, and manages the movement of rows, based on various criteria, to one or more storage devices (tape, WORM tape or disk). The second component, a Database Manager, provides the interface between SQL commands and the Archive Manager. Database Manager makes access to data in DB2 tables transparent to the user and to the applications requesting data.

When a table is archived-enabled using Lifecycle Director software, a new table and partition that are virtually identical to the original table being prepared for archive management are created. This new table includes one additional column that is used by Archive Manager to manage the location of each row's data. Once this new table has been created, the original table is dropped, and a view of the newly created table is created with the same name as the original table.

In many cases, the new table can be prepared with minimal production downtime. Creating this table uses standard DB2 commands, and no recoding is necessary.

The new approach does not impact indexes or referential integrity. Once the new table has been created, all rows remain in the new table. Referential integrity will be unchanged, and properly created queries will continue to be processed, without any code changes necessary. To the calling applications, the new table will look identical to the original table. All columns, offsets, indices and relationships will remain intact. All references will remain intact. No new coding is required to support the new table.

Although the table modification will be transparent to applications calling for data, the impact on table size, and on the overall organization, can be dramatic. The Archive Manager will apply rules to each row of the table. Policies for migrating and restoring of rows of data in an archive-enabled table are implemented via SQL commands in this new approach. Standard SQL selection criteria are used to select rows for processing.

To see how Archive Manager works, let's assume that Archive Manager is configured to archive any row that contains data more than six months old. The data to be archived for each row will be moved to tape (or, optionally, to disk). While the data in the row is being archived, an 18-byte stub is written into the main table. This stub provides a pointer to the location of the row's data.

During a query in which data in a row that has been archived is requested, Archive Manager reads the stub, locates the data on tape (or other media), reads the data, and passes it transparently back to the calling application. The data retrieval is transparent to the calling applications (although there may be delays related to retrieving archived data from archival media).

In most cases, the data is read and passed back to the active table, but it is possible to move the data back into the original table. When this is done, the data is put back into the original row, and the stub for that row is removed. At some time in the future, it remains possible that this old data will again be archived, with new stub data that identifies the location of the row's data.

When rows are archived, the archive stub in the active table and the row where the data once resided are deleted. The archived data will subsequently be removed from the Archive Manager database via a Database Manager housekeeping procedure. The data that once existed in the table will be gone and, unless the data was written to WORM media, the actual data will also be deleted. Complete removal of this data is easily implemented.

Because the active table is only a fraction of the size of the original table, reorganization of this active component can be done more rapidly than it can be with today's large tables, and using considerably fewer resources than would have been possible on the original table. Additionally, since so much space has been recovered as a result of the archiving of a major portion of the original disk storage, considerable free disk space will substantially reduce the need to throw new disks at the overcapacity problem.

Optionally, mirrored tape can be implemented. Tape mirroring offers a number of benefits. If the mirror is at a remote location the primary benefit is redundancy. Should a disaster occur, archived data can be restored from the mirrored tape.

For applications where regulations require write-once capabilities, Archive Manager can also support WORM (write once, read many) tape devices.

Clearly, reading data from tape takes longer than retrieving data from disk. Depending on the tape technologies in use, retrieval of archived rows may be done in as little as a few seconds. If the rows are located in adjacent areas of a mounted tape, access to the data can be nearly transparent to a calling application. Optionally, rows for which retrieval is anticipated can be pre-fetched from tape to disk by Archive Manager. This data, although still archived, can be retrieved by an application at disk speeds.

4. Faster disaster recovery

After a disaster that damages or destroys a data table, the traditional recovery method restores the data from tape in a first-in (first created)/first-out manner. The oldest, least frequently used, data is restored first. The most important data, recorded last, is restored at the end of the process. For large data tables, it could take days until this most valuable data again becomes accessible.

Using Archive Manager, it is possible to restore data in a LIFO (last-in/first-out) order. This restores access to the newest data, followed by rows that contain older data. If the tapes are intact, only data from most recently created rows (for example, the rows that were created in the last six months) is restored to the table, and the stubs for older data are rewritten to the restored table. An archive-managed table can not only provide rapid access to the most recent data, because less data is actually being written into the restored table, complete restoration can be extremely rapid when compared to the current approach. As long as indexes and tapes are intact, an organization can be up and running, usually within hours after a disaster.

Further, because the archived data is already stored on tape, backups need only be made of the much smaller, active tables. Additionally, when data is restored, it is the most recent data, plus the stub data, that is written. The archived data, which is not written to the active table, remains on the archive media. Backups thus take a much smaller time window than is currently possible, and restores are also much more rapid. The reduced overhead — and pain — related to backing up active tables should increase the frequency of backups being performed.

5. This is not HSM

Hierarchical storage management applies rules to entire tables. An active table that has been in constant use for the past 10 years will typically contain 10 years of data — most of which is of little value to the organization. Under HSM rules, this bloated table is not a candidate for any management, since it is still being used and contains new data.

Archive Manager works with data in the table on a row-by-row basis. Archive Manager would leave this decade-old table relatively intact. However, using basic rules, all data that is older than seven years may be deleted from the table (possibly reducing the active table size by 30 percent). Data that is older than six months will be archived, with the data in each archived row replaced by an 18-byte stub. The amount of disk space that is freed by archiving could be as much as another 65 percent. The overall size reduction for a still active, 10-year-old table — untouched by HSM — can be as high as 90 percent to 95 percent using Archive Manager. All the data for the last seven years, in this example, will still be accessible to applications, and the benefits of the reduced disk space will become obvious quickly.

In this example, it is assumed that data older than seven years can be purged, and that data over six months old can be archived. Archive Manager can be configured to apply rules needed by the organization.

6. Comparison to other approaches

The current options, available from third parties, or developed in-house, typically require re-coding of applications in order to access the data. Typically, these approaches require considerable coding of applications to read data from the many tables that are created. Further, significant testing may be required to assure that the applications are properly retrieving the data. Further, referential integrity is at risk.

The new approach does not require new coding. If an organization had an application that queried data for the last three years, the data can be retrieved from the archive-managed table. The data that has been archived will be recovered from tape (or archive disk).

Additionally, this new method also provides automatic storage allocation. This eliminates the necessity of maintaining significant overcapacity. No other approach provides this benefit.

Because most of the data in the archive-enabled tables is already stored on tape, backup windows are considerably smaller than with any other approach. Support for tape mirroring, WORM tape, and LIFO-restore capabilities also distinguish the new approach from others currently available.

7. Putting it all together: the implementation — install and configuration

The Lifecycle Director solution will begin with an assessment of the existing database and application set. During this assessment, database profiling scripts will analyze the DB2 catalog to examine data types, row lengths and basic aging patterns to determine which tables are the best candidates for archive. This assessment is generally coupled with an interview of subject matter experts, to solidify and validate findings. The results of this assessment are entered into a ROI tool, to develop a business case and benchmark against other implementations.

The trained professional services team will assist in the installation and configuration of the Lifecycle Director Archive Manager and DB2 Manager software products and will also assist in enabling DB2 data structures for archiving to an automated tape environment

8. Conclusion

The new approach to DB2 data management does not require coding modifications, thus reducing the expenses related to creating new (or modifying old) applications necessary to work with changes made by third-party or internal archiving modifications.

Further, this new approach evaluates each row of data, archiving to tape where appropriate, and deleting where rules allow deletion. Referential integrity is maintained, and no coding changes are required.

This approach can provide many benefits to organizations adopting the methodology. Active disk storage can be reduced by 80 percent or more. Old data can be archived, yet it can be easily called by applications. The cost of managing old data on hard drives can be severely reduced.

Automatic storage allocation will reduce the need for maintaining costly overcapacity. The frequent addition of new disk storage devices will be reduced or eliminated. This will enable longer use of existing storage devices, while also reducing the capital costs related to purchase of new drives, and the considerable costs related to managing the extra storage (and even the cost of housing and powering the drives).

Compliance with regulations relating to how data is stored and how long it is stored can be easily implemented using this approach. Backup and restore of active data will be much less time consuming (since less data must be backed up or restored), and, as a result less costly. Data can be recovered more rapidly, and, in many cases, the most recently used data can be restored first, significantly reducing an organization's costs from lost access to its data. Further, the smaller backup window may make it possible to perform more frequent backups, further protecting the data.

The vendor plans to charge on a per-mainframe basis. The pricing structure is clear and predictable. The overall TCO of data table management will be significantly improved as a result of reduced disk-related costs. Access to the most valuable data should be improved. The cost of managing data tables can be significantly reduced using this new approach and appropriate information lifecycle management strategies.

ABOUT STORAGETEK®

Storage Technology Corporation (NYSE: STK) is a $2 billion global company that enables businesses, through its information lifecycle management strategy, to align the cost of storage with the value of information. The company's innovative storage solutions manage the complexity and growth of information, lower costs, improve efficiency and protect investments. For more information, visit www.storagetek.com, or call 1.800.275.4785 or 01.303.673.2800.

WORLD HEADQUARTERS

Storage Technology Corporation
One StorageTek Drive
Louisville, Colorado 80028 USA
1.800.877.9220 or 01.303.673.5151

FT 0006 A 12/04