Relational Database Active Tablespace Archives Using HSM Technology

David I. Boomer
IBM
boomerd@us.ibm.com

Abstract

This paper describes a prototype active tablespace archive that was created using standard relational database data management capabilities of IBM's DB2 Universal Database (UDB) and the hierarchical storage capabilities of IBM's High Performance Storage System (HPSS). In the prototype, DB2 UDB and HPSS are linked using the capability of UDB to use a standard UNIX/Posix file for table space and the ability of HPSS to present a standard UNIX/Posix file system interface for both disk-resident and tape-resident data through its Linux Virtual File System (VFS) feature. This integration allows the current data and the active tablespace archive to be accessed with one set of semantics and, through federation, would allow them to be seen as a single data source.

1. Introduction

Here we introduce the topic of relational database active tablespace archives and explain its relevance to relational database archiving and hierarchical storage management.

A relational database active tablespace archive is defined here to be an archive that can be addressed directly using relational database semantics, even if the archive is on tape, or partly on tape. First we introduce the concept of an active tablespace archive in terms of integrating standard relational database capabilities with a hierarchical storage management system. We describe the Virtual File System (VFS) interface that makes this linkage possible without modifying either the relational database management system code or the hierarchical storage management system code. We then describe the prototype implementation and the functional tests that were run. Future work is described, including the need for more performance measurements. We conclude with an assessment that the active tablespace archive can significantly enhance access to archived data for data mining and similar near-real-time applications.

1.1. RDBMS Archiving

Typical RDBMS archive solutions are based on a two-tiered approach, the first tier being history databases and the second tier being flat or non-relational data files. The history databases use disk technology and house the archive data for a predetermined amount of time, after which, it is moved to the second tier file archives. This architecture allows the second tier file archives to take advantage of less expensive storage media, such as tape, either directly or via hierarchical storage management (HSM) technology. However, placing the data in non-relational file archives adds complexity to data retrieval process when the archived data is needed to satisfy business queries.

In the active tablespace archive architecture, the first tier history databases can interact directly with the HSM technology, reducing and/or eliminating the need for a "second tier" based on non-relational file archives. The second tier and beyond (3rd, 4th, etc.) in the active tablespace archive scenario now become relational database tables defined using different storage technologies managed by the HSM.

This architecture allows users to access both the current operational databases and the archive databases with the same semantics and table structures. It is even possible to federate the archive and production databases to appear as a single data source to the end user. This interaction between RDBMSs and HSMs reduces the need to design and manage complex retrieval processes for second tier non-relational file archives as well as maximizes the availability of the archived data. This may also simplify development of new applications and make it possible to interface legacy software with the Active tablespace archives.

1.2. Impact of Large Volumes of Data

Improving production database performance and minimizing the storage investment are common goals. Retaining seldom-accessed data in the production database adds cost and overhead. We quote Smith and Vogel on this topic:

> With databases growing at an exponential rate, ever-increasing volumes of data are a pressing concern for data centers. An accumulation of historical transactions, the advent of data warehousing, and other factors contribute to the accumulation of inactive data, which is less likely to be accessed. — But businesses can't necessarily (or may not want to) get rid of inactive data altogether for several reasons. Some

might need the data to comply with government requirements. Other businesses might want the data because they anticipate building trend analyses. And many keep inactive data to maintain a complete history of their customers [1].

It is clear that managing the size of mission critical databases is important. The benefits include [4, 5]:

- Reduced storage costs
- Improved application performance
- Faster response times
- Reduced overhead of maintenance activities
- Deferred and/or reduced upgrade and maintenance fees

1.3. The Basics

Archiving is the process of moving unused or seldom accessed data from the production database to an archive data store. The data stores can be [1, 4, 5]:

- Table archives when fast or SQL access is necessary.
- File archives when SQL access isn't needed and access is less time sensitive or to take advantage of less expensive storage technology. Access to the file archives is more complex and may require multiple steps such as determining the requested data location and the loading or staging the data back to a format that cam be queried.
- Combination of table archives and file archives. Used when the data has a defined access pattern that requires it to remain on disk for a specified amount of time then moved to file archives.

Figure 1 illustrates the typical archive architecture for relational data. The first step is to move the less-used data to the table archives (history tables). Then, after a predefined period of time, move the data from the table archive to the file archives.

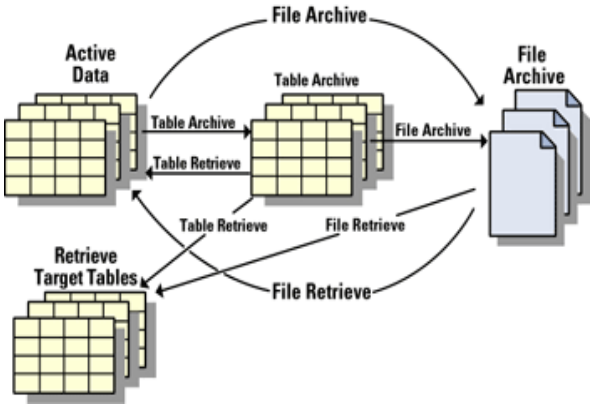

Figure 1. Typical Archive Architecture [1]

There are several archiving applications that use this architecture. These applications provide the tools for managing the data relationships and ensuring that all of the archived data maintains its business context. They also provide rules defining when and what gets archived (and then purged), so specific related sets of records get archived versus archiving an entire table [4, 5].

1.4. The Challenges

Organizing Archive Activities. Most of the archiving tools are very good at identifying data relationships and organizing the archive activities. Much thought needs to go into:

- Determining the access patterns of the data including access frequency, expected service times, and typical result set size (# of records), classify as 10's, 1000's, 1,000,000's, etc.
- Determining when data is available for archival.
- Determining purge characteristics, what data should be purged and when.
- Cost of storage technology alternatives.

Access to file archives. When access to data residing in file archives is needed, the following must be managed:

- Determining which file archives contain the requested data.
- Retrieving the data to the target tables.
- Querying/using the data.

Some of the generally available archiving tools provide interfaces to manage and simplify access to data stored in file archives [5]. However, access is still controlled and managed by the custom interface. History databases, on the other hand, provide a standard SQL based interface as well as other tools such as indexes and views that can be managed by the end user. One of the database tools available is a "union all view". This could be constructed over the production and history data to provide a single interface for the end user queries.

Schema Changes. Mission critical data schema can change. As the production application evolves, so do the data structures supporting it. Over time the structural context of the archived data will vary from the production version. Most archive tools will manage the metadata describing the data in the file archives. This will be used when retrieving the data in order to maintain its original business context. Deploying schema changes against historical data is typically avoided because of complexity, legal reasons and/or corporate policy.

More emphasis has been placed on how to restore it to its original structures and then relating it to the current production schema structures.

However, with the ability to store and access all of the historical data through a relational database, the option to deploy schema changes may be more feasible. Keeping 2 copies of the data may be an option now: an original to satisfy legal/corporate policies and a copy that incorporates the production schema changes.

2. Basics of Hierarchical Storage

HSM technology manages files across multiple tiers of storage technology. These tiers are classified by access requirements and cost. The top of the storage hierarchy is used for faster access and higher cost storage technology such as high speed disk, and the bottom is reserved for data requiring low frequency of access and lower cost storage devices such as tape. The HSM provides intelligent management of how and where the data resides in the storage hierarchy. In most cases, the data starts at the top of the hierarchy and moves down based on migration policies managed by the HSM [2].

When a file that has migrated down the hierarchy is again accessed, the file is staged back to the top of the hierarchy. Upon the I/O request, the file is staged back to disk from the storage hierarchy, where the I/O request is satisfied. However, when incorporating database object files it can be cumbersome, especially where the table files are very large (100s of GB and even terabytes) and the request is for only a few records.

3. The Prototype active tablespace archive

Our prototype used IBM's High Performance Storage System (HPSS) as the HSM component and DB2 as the relational database management system.

HPSS is software that manages multiple petabytes of data on disk and robotic tape libraries. A new version 6.2 of HPSS provides a standard Linux/POSIX read( ) write( ) interface that is implemented as a Linux Virtual File Server VFS). HPSS provides highly flexible and scalable hierarchical storage management that keeps recently used data on disk and less recently used data on tape [2].

DB2 Universal Database (UDB) has the capability to use a standard Linux/POSIX file for its tablespaces, and this enables it to place its table space on an HPSS file. This capability allows transparent storage of DB2 data object files directly in the HPSS storage hierarchy. This transparency allows the data to exist anywhere in the storage hierarchy regardless of the technology (disk, tape, etc.). HPSS has the capability to manage 5 tiers in the storage hierarchy. This capability would allow a data archive repository to be constructed with the following tiers:

- High speed disk
- Slower, less expensive disk technology
- High speed, low capacity tape libraries
- Low speed, high capacity tape libraries
- Shelf tape

Initial movement of the data from production could be directly to the first tier in the hierarchy (High speed disk), and then moved down the hierarchy based on policies managed by the HSM.

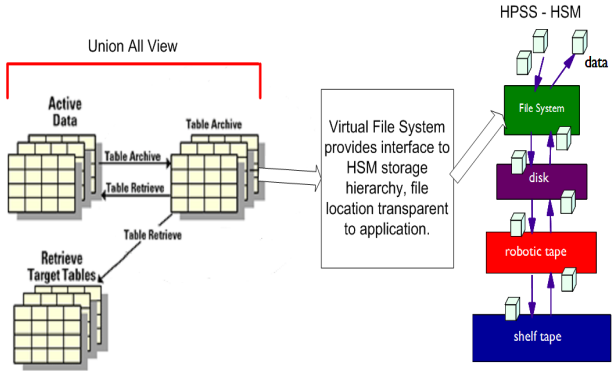

Figure 2. DB2 table archives built inside of HPSS

Figure 2 shows an archiving architecture that does not require file archives in order to take advantage of less expensive storage technology. When DB2 requests data via an SQL query, HPSS will satisfy the request by reading the data directly from the file object wherever it exists in the storage hierarchy, no staging back to disk is necessary.

4. The Prototype

The goal of the prototype was to demonstrate that a relational database could be constructed using an HSM storage hierarchy for the table data.

The prototype database was developed using multiple copies of data across a variety of tablespace configurations. The HPSS configuration used two storage hierarchies: "disk only" (single tier) and "disk to tape" (two tiered). Some of the tablespace configurations were distributed such that some used disk-only HPSS class of service and others used the "disk to tape" hierarchy [2].

Figure 3 illustrates table configurations using DB2 tablespace container files defined to use the mounted HPSS virtual file system and high speed disk arrays.

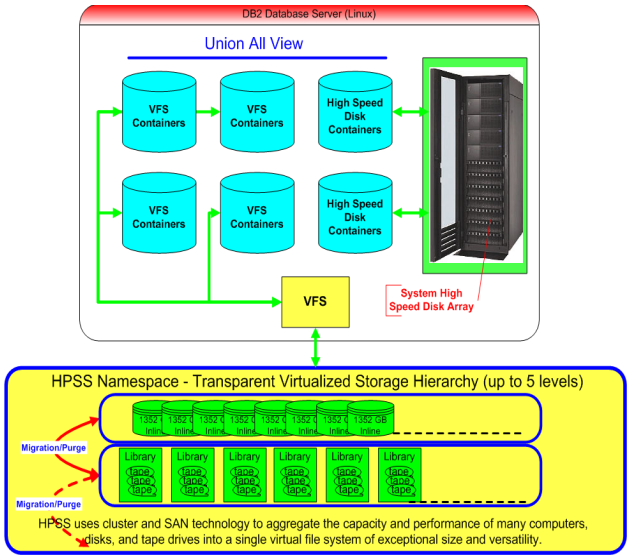

Figure 3. Prototype Environment

The prototype used a "union all" view defined over all of the data sources. This view was used as a query source to demonstrate the ability to submit a single query that searches data in production tables defined using high speed disk arrays and data in history tables defined using an HPSS storage hierarchy.

5. Future work

The current prototype environment demonstrates functional capability. Plans include:

- Incorporating new partitioning capabilities available in the next release of DB2 and analyzing its impact on the architecture of the archive databases and the HSM configuration.
- Environments that test scalability.
- Incorporating a Data Warehouse Monitor to analyze what is getting accessed in the archive. This information could be used to determine when and what to move up and down the HSM storage hierarchy [3].
- Federation of HSM-managed archived files into historical databases. New capabilities with federation make flat file data available to a relational database as though it were another table. These federated files will not have the same capabilities as regular relational tables, but there may be enough function to include the file data in normal business queries and satisfy the requirements of a data archive.
- Impact of an HSM managing the database storage objects on normal database maintenance activities such as backups and code updates.

6. Conclusions

The active tablespace archive architecture can significantly simplify the interface to the archived data. The remarkable capabilities of this architecture include:

- This capability allows users to access both the current operational databases and the archive databases with the same semantics and table structures.
- The ability to transparently store relational database object files directly in an HSM. This means that no custom code need be developed in order to access the files. It is all handled via the virtual file system interface into HPSS.
- The ability to read the requested data from its location in the storage hierarchy without staging it back to the HSM disk cache. This capability is very desirable for scenarios that have large relational database object files with queries that are typically requesting small numbers of records. If this capability did not exist, the HSM disk cache may have to be much larger and therefore more costly, to handle the expected query workload.
- This active tablespace archive architecture significantly reduces the need to remove data from a relational database format.. The ability for the database to transparently take advantage of less expensive storage alternatives, such as tape, makes the file archive tier unnecessary.
- Schema changes against the archived history data are more of a possibility. This might be accomplished with the methodology provided to modify the production data store's schemas. The fact that the archives are in a relational database as opposed to archived flat files makes this more feasible.

References

[1] Bryan F. Smith (IBM) and Thomas A. Vogel (IBM), "*Take a Load Off: Archive Inactive Data*", published in DB2 Magazine Quarter 4, 2003 · Vol. 8, Issue 4.

[2] Harry Hulen (IBM), "*The Basics of High Performance Storage System*" http://www.hpss-collaboration.org/hpss/about/HPSS-Basics.pdf

[3] W.H. Inmom (Sun Microsystems), "Managing The LifeCycle of Data"

[4] "Store Smarter: Enterprise Active Archive Solutions," Princeton Softech, March 2005

[5] "Archiving Complex Relational Databases," Princeton Softech, March 2005