StorHouse/Relational Manager (RM) –

Active Storage Hierarchy Database System and Applications

Felipe Cariño Jr.
FileTek. Inc.
360 N. Sepulveda Blvd. Suite 1080
El Segundo CA 90245
FCARINO@filetek.com

John Burgess
FileTek, Inc.
9400 Key West Avenue
Rockville MD 20850
JGB@filetek.com

Abstract

This paper describes how database systems can use and exploit a cost-effective ***active storage hierarchy*.** By active storage hierarchy we mean a database system that uses *all* storage media (i.e. optical, tape, and disk) to store and retrieve data and not just disk. We describe and emphasize the *active* part, whereby all storage types are used to store raw data that is converted to strategic business information. We describe an evolution to the Data Warehouse concept, called ***Atomic Data Store***, whereby atomic data is stored in the database system. Atomic data is defined as storing all the historic data values and executing queries against the historic queries. We also describe a Data Warehouse information collection, flow and central data store ***Hub-and-Spoke architecture,*** used to feed data into Data Marts. We also describe a commercial product; ***StorHouse/Relational Manager (RM)***. RM is a commercial relational database system that executes SQL queries directly against data stored on the storage hierarchy (i.e. tape, optical, disk). We conclude with a brief overview of a real world AT&T Call Detail Warehouse (CDW) case study.

1.0 Introduction

Commercial Database Management Systems (DBMS) have evolved and been developed to diverse and ubiquitous range of applications. DBMS have been based on hierarchical, network, relational, object-oriented and the new emerging object/relational database model. With few exceptions these database systems and applications primarily use disk media as their storage. Hierarchical Storage Management (HSM) is used by some of these applications to exploit some of the benefits of cost-effective optical storage systems.

We propose and analyze that database systems use and exploit a complete active storage hierarchy (i.e. tape, optical, and disk). The key proposal is that active data be *also* stored, queried and analyzed on tape farm libraries and optical jukeboxes. Figure 1 shows the cost, performance, size and reliability considerations. In this paper, we use the term active storage hierarchy when (say SQL) queries execute against data stored on diverse media.

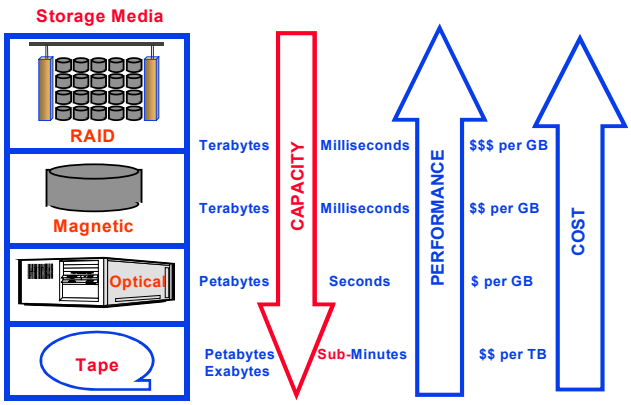

Figure 1: Cost, Performance Size Tradeoffs

We provide an overview analysis of different commercially available storage systems. We provide an update to tape and optical technology, performance and systems described and analyzed in [HS 96] [JM 98]. During VLDB 1998 10-year best paper award [GB 88] talk Jim Gray described emerging disk technology trends. Key disk technology trends are more storage and CPU-like processing capabilities [RGF 98]. We claim and discussed at a [VLDB 99] panel why the complete storage hierarchy media is needed to address certain application needs (i.e. atomic data, objects, etc.)

The *current* economics of the storage media are that storing data on tape is approximately 7% the cost of storing the data on disk. The cost for storing data on optical is about 42% of storing the data on disk. Some interesting byproducts of cost-effective use of active storage hierarchy described in the paper are:

- *Atomic Data:* where all the historical data can be stored and directly queried. Other users must make decisions what data to keep "on-line" on disk for querying. We could almost summarize this paper by the following: "Users should keep all data on-line for querying all the time". Data can be moved to the most cost-effective storage hierarchy media. Instead, what customers do today is "age out" data to archive and rarely restore it for querying.
- *Atomic Data Store (ADS):* a Data Warehouse (DW) concept evolution where historical atomic data is stored and used for information mining or decision support. Bill Inmon spawned an information revolution with what is now know as a Data Warehouse [Imn 92]. There are several potential DW and Data Mart architectures [Gar 98] and philosophies [BZ 98]. One of the DW architectures is an enterprise-wide DW where detail data is stored and used for strategic reasons [Arm 97]. Implicit in [Arm 97] value of detail data argument is that data is aged or migrated out of the repository. The case for ADS DW concept is the same as DW vendors that sell the notion of storing detail data, except that in ADS the detail data is actually stored and used (i.e. not migrated out of the DW).
- *Hub-and-Spoke Architecture (Figure 2):* where operational data stores load (all) their atomic data values into a Data Warehouse with active storage, and then feed Data Marts. This provides a cost-effective way to manage and load data in Data Marts.

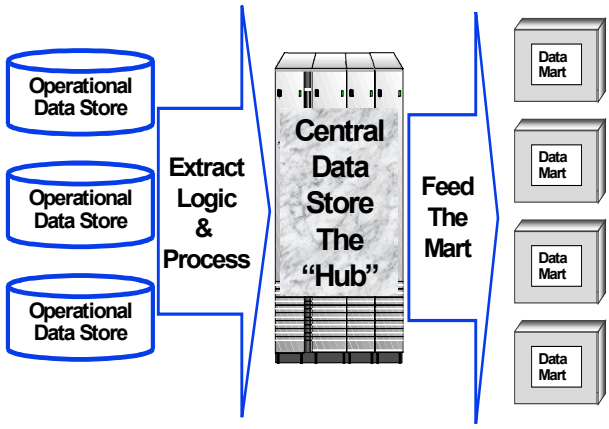

Figure 2 Hub-and-Spoke Architecture

StorHouse/RM is a commercial relational database system that support SQL-92 queries for data stored in the storage hierarchy. StorHouse/RM handles the data placement issues described in [CTZ 97]. We describe and analyze StorHouse/SM (Storage Manager) which is the storage manager used by StorHouse/RM. This paper concentrates on the relational database uses of storage. Other non-database centric uses of StorHouse/SM can be found at www.filetek.com. [CB 99] provides an analysis of (a) storage trends, (b) database trends, (c) commercial products and (d) AT&T's use of active storage hierarchy for their Call Detail Warehouse application.

This paper is organized as follows: Section 2.0 analyzes StorHouse/Relational Manager (RM), a database system that supports the diverse storage systems. Section 3.0 describes StorHouse/Storage Manager (SM) which manages diverse storage devices and storage media. We conclude describing future work.

2.0 StorHouse/Relational Manager (RM)

There are three major ways database systems use diverse storage hierarchies:

(1) Hierarchical Storage Management (HSM) is used to migrate data to optical storage. An HSM implementation strategy to place a marker in the database and move the value to optical. The query engine must understand where and how to access data (on disk or optical).

(2) Data Backup and Archival copies or moves data to tape or optical storage. Data backup is used to make a complete or partial copy of the database in order to restore databases in case of disasters or data corruption problems. Data archival writes the data to optical or tape (tape is most likely) and then removes the data from the database. A customer may decide to keep "N periods" of data and archive (i.e. migrate) data at the "N+1 period". In order to use the archive information, users must restore the data and then later delete it. This is a cumbersome process that for many practical reasons is rarely done.

(3) Atomic Data Store, as described earlier, is where all the historical data is stored on some or all of the storage hierarchy media.

StorHouse/RM

StorHouse/RM is a database system that supports SQL-92 queries against the storage hierarchy. This database system was designed and optimized to store (atomic) data on diverse media. StorHouse/RM

(RM for short) works in conjunction with StorHouse/SM (SM for short) to specifically administer the storage, access, and movement of relational data. SQL access is available from different platforms through a variety of industry-standard protocols. RM runs on Sun Microsystems Ultra Enterprise platforms.

Figure 3 shows that RM architecture and components are similar to all major commercial relational database systems. An RM database consists of the following:

- User table data that you store and access.
- Optional indexes—value, hash, and range—that locate the table data.
- Metadata that describes database components.

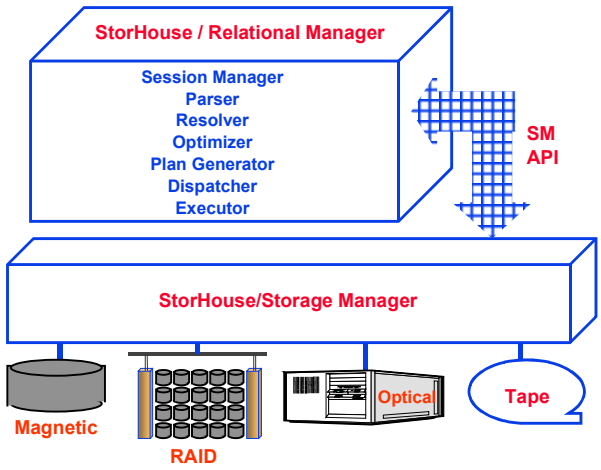

Figure 3: Relational Manager Architecture

RM databases (Figure 4) have both a logical and a physical structure. Logically, user tables and associated indexes reside in user tablespaces, and metadata resides in a system tablespace. Physically, user tables, indexes, and metadata are stored in SM files. RM user tables consist of one or more table segments. Each table segment is a separate SM file. Whether a user table is composed of one or multiple segments depends on the size of the user table and whether you later load more data into the table.

Extents

Table and index segments are RM files that can reside on any storage device in the RM storage hierarchy. These files are composed of different (Data, Definition and Map) extents. (1) Data Extent – holds user data and/or control data. (2) Definition (DF) extent - contains information necessary to retrieve the data. (3) Map Extent - is the high-level index that RM always reads first when doing index lookups. You can retain some or all extents in the magnetic disk performance buffer to enhance performance. For instance, you can hold the value index and hash index DF and Map extents on the performance buffer longer than the table Data extent to speed access to the data. To add more data into a table, RM would create a new table segment and corresponding value and hash index segments. Range indexes do not consist of index segments; they are stored in the Meta Data.

Value and hash indexes also consist of index segments. Each index segment is a separate RM file. For each value index on a table, there is always one value index segment associated with each table segment. For each hash index on a table, there is always one hash index segment associated with each table segment.

UNIX Database Files

RM database metadata and range indexes reside on and are managed by RM on the UNIX file system. Each system table, system table index, system table log, system table index log, and range index is a separate UNIX file.

User Tables

A user table is the basic unit of data storage in a RM database. User tables hold user-accessible data. Logically, RM user tables are like most RDBMS user tables; they consist of columns and rows of data. Each row contains data values conforming to the constraints of the columns that make up the row.

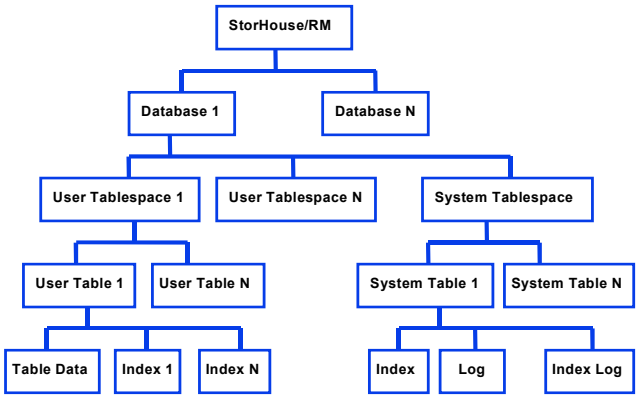

Figure 4: StorHouse/RM Database Layout

Physically, an RM user table is one or more files on the RM storage hierarchy. User tables can reside on any storage device in the hierarchy. The user tablespace defines the target storage device and the migration path through the hierarchy. For instance, you can store time critical data on RAID and then migrate that data to tape or optical as the data ages. The RM software automatically manages storage and migration based on your user tablespace parameters.

Indexes

Indexes provide efficient access to table data. You can create an index on a column or combination of columns in a user table. An index based on one column is a simple index. An index based on more than one column is a compound index. RM supports three index types—value, hash, and range.

Value index

Value indexes work best with queries that return multiple rows based on a range of values. A value index contains an ascending list of all the values in a column (or group of columns for a compound value index). For each column value, the index contains an index map to the table row containing that value. By searching the index rather than the table, then matching column values to row IDs, RM can more efficiently find requested table rows.

Hash index

Hash indexes work best with queries that return a specific record based on a specific value. A hash index is a two-part index based on an index map extent and a hit list that uses a proprietary RM algorithm to effectively locate individual table rows based on individual index values.

Range index

Range indexes are useful for user tables with multiple segments. A range index contains the lowest and highest column data values for each table segment for a user table. Instead of searching through

multiple table segments, RM first looks at the range index to find the specific table segment with the requested data values. Then, RM might use any hash or value indexes to find a specific data value or range of values in the table segment.

User tablespaces

A user tablespace defines where table segments, hash index segments, and value index segments are stored on the RM storage hierarchy. It also sets attributes that influence storage management like backup and migration. When you create a user table, you assign it to a user tablespace. The table's segments and corresponding index segments then are stored according to the specifications of the user tablespace.

Allocating Storage

You allocate storage by identifying the volume sets and file sets for all table, hash index, and value index segments stored in the user tablespace. Whether you use multiple volume sets and file sets in a user tablespace depends on your data and your access and performance requirements.

Volume

A volume is a unit of media on which data can be recorded and read. RAID, magnetic disk, erasable and WORM optical disk and DLT cartridges are all examples of RM volumes, or media.

A **volume set (VSET)** is one or more physical volumes that are treated as a logical unit of storage. You use volume sets to control the physical grouping of files. A user tablespace identifies the volume sets that will contain the table, hash index, and value index segments for all tables and indexes in that tablespace. You can store table segments and associated hash and value index segments on the same volume set or on different volume sets.

A **file set (FSET)** is an area of storage within a volume set. Files are stored in file sets. Table segments and index segments are files. You can store table and associated hash and value index segments in the same file set or in different file sets.

For example, if you want to minimize volume mounts and don't need to manage the storage of table data and indexes in different ways, then you could allocate one VSET with one FSET for all components.

Or if you want to manage the storage of table data and indexes in different ways but remove them from RM at the same time, you could allocate separate FSETs in one VSET. Or if you want to manage the storage of table data and indexes in different ways and don't need to remove them from RM at the same time, you could allocate separate VSETs.

Performance and Backup Copies

A user tablespace contains a **Vulnerability Time Factor (VTF)** attribute that determines whether and when you'll create performance copies and/or backup copies of user table and index data. Performance copies reside on the RM magnetic disk performance buffer. Backup copies reside in primary file sets on the designated RM media.

A user tablespace contains an **Access Time Factor (ATF)** that works with RM migrate function to keep data that is most likely to be accessed in the RM performance buffer while maintaining a supply of free space. In a user tablespace, you can assign the same or different ATF values for table, hash index, and value index segments. This means you can retain index segments on the performance buffer longer than the data for faster query processing.

Metadata

Metadata are system components that RM creates and uses to manage a database. These components include system tables, system table indexes, system table logs, and system table index logs. Metadata is stored on the RM UNIX file system within a system tablespace.

System Tablespaces

For each database, RM creates a separate database directory on the RM server. This database directory, also called the system tablespace, is a structure that contains all of the system components for a specific database. Each system component is a separate UNIX file. All system tables have corresponding table logs. Some, not all, system tables have indexes.

System Tables

RM automatically creates a set of system tables for each database and stores them in the database's system tablespace. System tables contain information about a database. RM uses the system tables to record, verify, and conduct work. For example, RM updates various system tables when you create database components, and it reads system tables to verify that database components exist and that accounts are authorized to access them. Each system table is a separate UNIX file. Just like user tables, authorized users can query system tables by submitting a SELECT statement.

System Table Indexes

RM automatically creates system table indexes for specific system tables when a database is created. System table indexes are stored as UNIX files in the same directory as the system table files. Their operation is transparent.

System Table Logs

Each system table has a corresponding system table log that is used to recover changes to system tables. Before RM updates a system table, it first copies a "before image" of any record being updated to the system table log and then makes the change in the system table. If the transaction fails or is rolled back, RM copies the before image back to the system table, removing the change. If the transaction completes or is committed, RM empties the system table log.

System Table Index Logs

Each system table index has a corresponding log that is used to recover changes to system table indexes. The operation of the index log is the same as the table log.

3.0 StorHouse/Storage Manager (SM)

StorHouse/SM, FileTek's comprehensive HSM software, controls a hierarchy of storage devices comprised of cache, redundant array of independent disk (RAID), erasable and write-once-read-many (WORM) optical disk jukeboxes, and automated tape libraries. StorHouse/SM is also responsible for critical system management tasks, like data migration, backup, and recovery. StorHouse/SM provides system-managed storage that optimizes media usage, response time, and storage costs for each application. StorHouse/SM runs on Sun® Microsystems Ultra Enterprise Servers and comes standard with all StorHouse systems.

Some of the important functions of this storage management software include:

- Provide common view of all storage including magnetic disk
- Automatically deal with all storage hardware failures to avoid system down-time
- Provide record/RDBMS-page access methods
- Maintain meta data on magnetic disk
- Maintain indexes on magnetic disk or optical
- Order accessing so all accesses for a volume are performed with one volume mount
- Order tape accessing so it is serial

- Duplex data on separate libraries so there is no single point of failure
- Create copies of data for off-site storage
- Maintain volume directories on volumes for recovery and transport to other systems
- Where data is on both optical and tape use optical for direct and tape for serial accessing
- Cache most active data on magnetic disk
- Migrate data from optical to tape.

Future Work

This paper is based on commercially available products and real customers. Future on-going work includes federated databases and development of StorHouse/ORM (Object/Relational Manager) to support (multimedia) SQL-3 types and functions.

Conclusion

In this paper we described the need for active storage hierarchy. We described the tradeoffs, uses and potential uses that using tape, optical and disk storage can provide. We defined a new Data Warehouse concept, called Atomic Data Store, whereby applications exploit atomic (historic) data using a central data store Hub-and-Spoke architecture, used to feed data into Data Marts. We analyzed StorHouse/RM, which is an SQL RDBMS that stores and retrieves data from the storage hierarchy (using StorHouse/Storage Manager).

Acknowledgements

We thank FileTek's technical publications Sandy Cornfeld from whose award-winning documentation we freely borrowed the StorHouse/RM description.

References

[Arm 97] Armstrong, B., "Data Warehousing: Dealing with Growing Pains", Data Engineering, March 1997. Pp. 199 – 205.

[BZ 98] Charles Bontempo and George Zagelow, "The IBM Data Warehouse Architecture", CACM September 1998, Volume 41, No. 9, pp. 38 –48.

[CB 99] Cariño, F. and Burgess, J., "Lost in Active Storage Space - New Frontier for IT Managers", To appear in Intelligent Enterprise Magazine 1999

[CTZ 97] Christodoulakis, S., Triantafillou, P. & Zioga, F. "Principles of Optimally Placing Data in Tertiary Storage Libraries", VLDB '97, pp. 236 – 245.

[Gar 98] Gardner, S. R., "BUILDING the Data Warehouse", CACM September 1998, Volume 41, No. 9, pp. 52 – 60.

[GB 88] Gray, J. and Bitton, D., "Disk Shadowing", VLDB 88, pages 331 – 338.

[HS 96] Hillyer, B. and Silberschatz, A., "Random I/O Scheduling in Online Tertiary Storage Systems", SIGMOD 96, pp. 195 – 204.

[Imn 92] Inmon, W., "Building the Data Warehouse", QED Technical Publishing Group, Wellesley, Massachusetts, 1992.

[JM 98] Johnson, T. and Miller, E., "Performance Measurements of Tertiary Storage Devices", VLDB 98, pp. 50 – 61.

[RGF 98] Riedel, E., Gibson, G. and Faloutsos, C., "Active Storage For Large Scale Data Mining and Multimedia", VLDB '98, pages 62 – 73.

[VLDB99] Panel @ VLDB 99,"Active Storage Hierarchy, Database Systems and Applications – Socratic Exegesis, http://www.filetek.com/vldb.htm