CONF-950985--6

PNL-SA-26099

SCIENTIFIC DATA MANAGEMENT IN THE ENVIRONMENTAL MOLECULAR SCIENCES LABORATORY

P.R. Berard
T.L. Keller

September 1995

Presented at the
IEEE Symposium on Mass Storage Systems
September 11-14, 1995
Monterey, California

Work supported by
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, WA 99352

Scientific Data Management in the Environmental Molecular Sciences Laboratory

Peter R. Berard and Thomas L. Keller
Pacific Northwest Laboratory, Richland, Washington

Abstract

The Environmental Molecular Sciences Laboratory (EMSL) is currently under construction at Pacific Northwest Laboratory (PNL) for the U.S. Department of Energy (DOE). This laboratory will be used for molecular and environmental sciences research to identify comprehensive solutions to DOE's environmental problems. Major facilities within the EMSL include the Molecular Sciences Computing Facility (MSCF), a laser-surface dynamics laboratory, a high-field nuclear magnetic resonance (NMR) laboratory, and a mass spectrometry laboratory. The EMSL is scheduled to open early in 1997 and will house about 260 resident and visiting scientists.

It is anticipated that at least six (6) terabytes of data will be archived in the first year of operation. Both the size of individual datasets and the total amount of data each researcher will manage is expected to become unwieldy and overwhelming for researchers and archive administrators. An object-oriented database management system (OODBMS) and a mass storage system will be integrated to provide an intelligent, automated mechanism to manage data. The resulting system, called the DataBase Computer System (DBCS), will provide total scientific data management capabilities to EMSL users.

The initial steps in implementing a production DBCS are complete. A prototype mass storage system based on the National Storage Laboratory's (NSL) UniTree has been procured and is in limited use. This system consists of two independent hierarchies of storage devices. One hierarchy of lower capacity, slower speed devices provides support for smaller files transferred over the Fiber Distributed Data Interface (FDDI) network. Also part of the system is a second hierarchy of higher capacity, higher speed devices that will be used to support high performance clients (e.g., a large scale parallel processor). The ObjectStore OODBMS will be used to manage metadata for archived datasets, maintain relationships between archived datasets, and hold small, duplicate subsets of archived datasets (i.e., derivative data). Metadata and derived data managed by the OODBMS will enable sophisticated command line, graphical, and programming language interfaces for organizing and efficiently accessing archived datasets. The resulting interim system is called DBCS, Phase 0 (DBCS-0).

The production system for the EMSL, DBCS Phase 1 (DBCS-1), will be procured and installed in the summer of 1996. The procurement of the DBCS-1 system is currently in progress. The scientific data management software developed on DBCS-0 will be ported to this production system in the fall of 1996.

This paper describes all efforts associated with DBCS-0 and DBCS-1, including software development, key lessons learned, and long term goals.

INTRODUCTION

Environmental Molecular Sciences Laboratory (EMSL)

The Environmental Molecular Sciences Laboratory (EMSL) is currently being constructed at Pacific Northwest Laboratory (PNL) under the aegis of the U. S. Department of Energy (DOE). The EMSL will house both permanent and visiting scientists in a 200,000-square-foot facility equipped with state-of-the-art instrumentation and computational resources. It will be a collaborative research facility, serving both its own staff and the scientific community at universities, industrial sites, and other government laboratories. This new facility will be a key element in PNL's response to DOE's environmental initiatives.

The facility will house the equipment and tools needed to perform advanced research (i.e., state-of-the-art laboratories, experimental equipment, and computers) in a single building. Facilities within the EMSL include the Molecular Science Computing Facility (MSCF), a laser/surface dynamics laboratory, a high-field nuclear magnetic resonance (NMR) laboratory, an environmental surface science laboratory, a mass spectrometry laboratory, and a host of additional instruments to support research activities. In this unique setting, scientists and engineers from a wide variety of disciplines (physical, environmental, chemical, materials, biological, and computational sciences) will collaborate in experimental and theoretical research in support of environmental restoration and waste management.

Scientific data management is a primary enabling technology for advanced research in many areas of science and engineering that are part of the EMSL. EMSL

research programs include:

- molecular level studies in the Theory, Modeling, and Simulation program
- soil and groundwater transport modeling in the Environmental Dynamics and Simulation program
- experimental data reduction and analysis in the Macromolecular Structure and Dynamics and Chemical Structure and Dynamics programs.

Some computational experiments in the Theory, Modeling, and Simulation and Environmental Dynamics and Simulation programs produce large data volumes on a large scale parallel processor. In addition, many instruments in the Macromolecular Structure and Dynamics Program produce large data volumes, including the Fourier transform ion cyclotron resonance (FTICR) and the NMR instruments. Data archival and retrieval supporting post-processing for these codes and instruments will be the primary driver of high performance database computer system procurements. The solutions to the data management problems in EMSL require acquisition of state-of-the-art computer networks, database management systems, and mass storage systems. Data management solutions also require the development of software that enables the integration of acquired technologies to be applied to individual applications. Accordingly, interim computing facilities were acquired with system characteristics similar to the envisioned production database computer system. (DBCS-1). The development database computer system (DBCS-0) [1] is a core piece of the interim computing facilities.

In the following sections we briefly discuss the current system, DBCS-0, and follow with a detailed discussion of the production system, DBCS-1, that is currently being procured. Next, an overview of the EMSL users' data projections for future years is presented. Finally, an in-depth description of the software development effort that is currently underway is provided.

Mass Storage Systems

Prototype and Software Development System

In order to support the scientific data management needs of scientists in EMSL, Object Design's ObjectStore object-oriented database management system (OODBMS) will be integrated with a state-of-the-art mass storage system. This integration effort has been underway on a prototype and software development system named the DataBase Computer System, Phase 0 (DBCS-0) [1]. The DBCS-0 system is composed of an IBM RS/6000 980 server running the National Storage Laboratory's (NSL) UniTree and two independent hierarchies of storage devices. Initially, this system was configured to support low-to-medium performance clients over the FDDI network with a hierarchy of storage devices composed of SCSI-attached disks and a Comtec 8-mm tape robot. A second hierarchy was provided to support high-performance clients over the High Performance Parallel Interface (HIPPI) network and consisted of an IBM 9570 HIPPI-attached RAID disk array and a Metrum VHS tape robot.

Due to a lack of HIPPI-connected clients and more demanding user requirements, the DBCS-0 storage hierarchies were reconfigured to more efficiently support the low-to-medium performance clients (refer to Figure 1). The SCSI-2 disk space was replaced with a 16-gigabyte Cambex RAID disk array and the Metrum tape robot was dedicated to supporting the low-to-medium performance hierarchy. The high-performance hierarchy still consists of the HIPPI-attached RAID disk array, but is lacking a tape robot to support near-line storage. For prototyping purposes, the Comtec 8-mm robot may be used for this hierarchy. In the event a HIPPI-connected client does become a reality in the future, a higher performance tape robot will be acquired for this hierarchy.

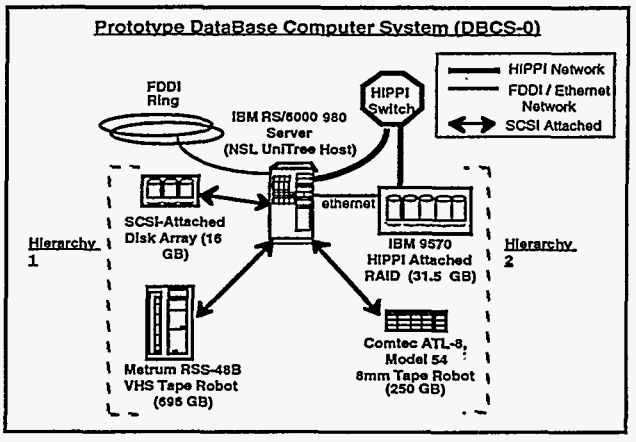

Figure 1. Prototype DataBase Computer System(DBCS-0)

Production System: Functional Requirements and Characteristics

The DBCS-0 system will eventually be replaced by the production database computer system, DBCS-1. The procurement of DBCS-1 is currently underway and the system is planned for delivery in June 1996. Until the contract is awarded in February 1996, the architecture and specific characteristics of DBCS-1 are unknown. The remainder of this section describes architectural concepts we believe are important in an enterprise-wide mass storage system (DBCS-X).

DBCS-X will provide the capability for an enterprise-wide mass storage system accessible from the variety of the EMSL's client computing platforms. We believe an enterprise-wide mass storage system should be

architecturally designed to efficiently .support the respective throughputs, capacities, and file sizes of the different classes of client computing platforms.

A DBCS-X system should be capable of efficiently supporting the data storage, search, and retrieval needs of a variety of high-performance and medium-performance client computing platforms connected to different classes of networks within the EMSL facility. High-performance clients include a soon-to-be-acquired large scale parallel processor named the High-Performance Computer System, Phase 1 (HPCS-1), as well as large experiment computers, analysis computers, and high-performance graphics servers. Medium-performance clients include workstation class nodes and servers, as well as compute clusters. The experimental instruments in EMSL may be considered high- or medium-performance clients, depending on the network connection. Lower performance clients are defined to be desktop workstations (i.e., DOS/Windows and Macintosh platforms). All client computing platforms in EMSL are candidates for two types of DBCS-X services based on their network connectivity and the size of files manipulated. The two *classes of clients* are a medium-performance class and a high-performance class. Those computing platforms that fall under the high-performance class may also qualify for the medium-performance class' services if the client has the proper network connection and is manipulating small- to medium-sized files.

The networking structure within the EMSL can be logically divided into two levels of performance. A high-speed network (e.g., HIPPI, ATM, FCS) called the Computer to Computer Network (C2N) will be used to support high-performance clients, and multiple medium-speed networks (e.g., FDDI, ATM) called the EMSL Backbone Information Network (BIN) will be used to support medium-performance clients. A DBCS-X system must provide sufficient throughput and storage capacity to support all classes of client computing platforms that are connected on these two networks.

The HPCS-1 system that is initially deployed in EMSL will have multiple (4 to 8) high-performance I/O channels. This system will likely be upgraded with additional high-performance I/O channels (possibly up to 16) in future years. Most high-performance graphics servers will only have a single high-performance I/O channel, although it is possible that some graphics servers may have two or more. It is expected that most medium-performance clients will only have one I/O channel connected to the medium-performance network (note that some medium-performance clients may also be connected to the high-performance network). A DBCS-X system must provide sufficient throughput and capacity to support all I/O channels of the HPCS-1 system that is deployed in 1996-1997, as well as the throughput and capacities required of the medium- and low-performance clients. A DBCS-X system must be capable of scaling in both throughput and storage capacity in future years in order to support the increased throughput and storage requirements of an upgraded HPCS-1 system, as well as the other EMSL computing clients.

Efficiently supporting two classes of clients within the EMSL requires a mechanism to clear and populate the client's disk space in a timely manner to ensure optimal use of the client system. The HPCS-1 system, a very high-powered large scale parallel processor with considerable memory and disk space (several hundred gigabytes), will be one of the most demanding DBCS-X client computing platforms. Other high-performance clients will also require fast access to storage and retrieval from a DBCS-X system. High-performance storage peripherals are capable of meeting the needs of these high-performance clients. These devices are typically optimized for handling large files efficiently. Consequently, high-performance devices should not be responsible for storing smaller files typically associated with lower performance clients. A DBCS-X system must also be capable of scaling to meet the future needs of these high-performance clients. This can be realized by the scaling of storage capacity in any of three-dimensions:

1. adding new levels of storage devices within a *class* of storage (medium- or high-performance storage classes)
2. adding new classes of storage with appropriate storage devices
3. adding more storage capacity to existing levels within a class of storage.

At a minimum, a DBCS-X system should have no less than 400 gigabytes of disk cache and 20 terabytes of robotically-controlled near-line (tape) storage. All storage peripherals will be dedicated to one of two classes of storage. That is, one class of storage will be dedicated to supporting the archiving needs of low- to medium-performance client computing platforms connected to the BIN. In general, these clients will store and retrieve small- to medium-sized files (refer to the discussion of "EMSL User Data" below). At least 100 gigabytes of disk cache and 6 terabytes of near-line storage will be provided to support this class of storage. The second class of storage will be dedicated to high-performance client computing platforms (e.g., HPCS-1) connected to the C2N. Large- to very large-files will constitute the bulk of the data stored and retrieved by high-performance clients. Even so, these clients will generate and manipulate small- to medium-sized files and will require those services from a DBCS-X system. At least 300 gigabytes of disk cache and 14 terabytes of near-line storage will be provided to support the high-performance class of storage.

The connectivity of all components internal to

DBCS-1 will be decided by the system's integrator. The DBCS-1 specifications define the minimum requirements for the BIN and C2N networks with options to select current network technologies available at the time of contract award. The servers and network-attached peripherals must use the networking fabrics available in the EMSL. The key objective in designing the internal and external connectivity of the system is to maximize throughput between the DBCS-1 system and all computing client platforms in the EMSL. This implies that the vendor has successfully minimized the total amount of time required to store/retrieve files of any size to/from DBCS-1, independent of which level of storage the file resides. To accomplish this, the vendor must:

- maximize utilized bandwidth and minimize latency on the given networking fabric
- maximize the aggregate throughput achieved to/from high performance computing platforms with multiple I/O connections
- minimize the I/O latencies associated with staging/migrating files between levels of storage devices within a hierarchy.

The EMSL will be an open facility that is used by resident scientists, as well as many visiting scientists. Visiting scientists must be able to import data they have brought with them into the DBCS-1 system. Likewise, these scientists also need the capability to export data from the DBCS-1 system onto removable media upon completion of their work in the laboratory. A facility for importing and exporting files in DBCS-1 will be provided. DBCS-1 must also provide an import/export facility whereby users will be able to either import data from a given off-line media or export data from DBCS-1 to a given off-line media to take to locations outside EMSL. The facility must be capable of allowing utilities to use non-HSM formatted removable media reads and/or writes to the off-line media (e.g., UNIX tar command). A wide variety of removable media types will be supported by this facility.

EMSL User Data

As shown in Figure 2, the sources of data within the EMSL will be from a wide variety of instruments and computing platforms. The amount of data archived annually by each source is expected to increase significantly between 1995 and 2000. The volume of data produced by the large scale parallel processor and the graphics servers are expected to be dominate consumers of the DBCS-1 storage resources. These client computing platforms must have ready access to all of their data in DBCS-1 with minimum latency, regardless of whether the data resides on the DBCS-1 disk cache or tape storage. The DBCS-1 system must be designed in such a way that minimizes contention for storage resources (e.g., a tape cartridge) to ensure a minimum latency.

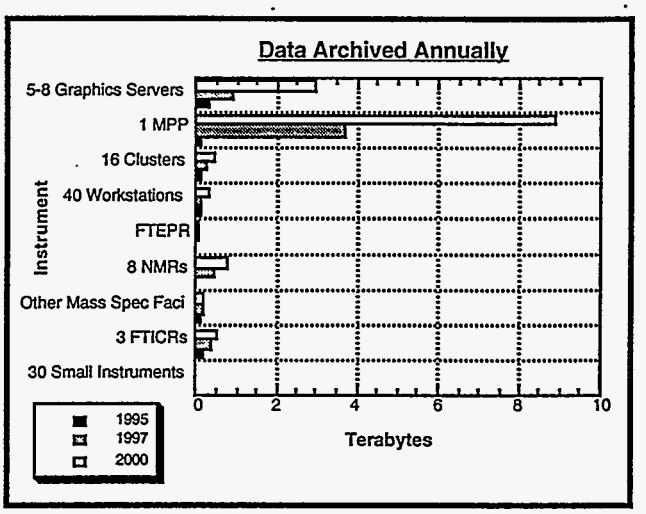

Figure 2. Data archived annually

As depicted in Figure 3, the total amount of data that is archived by all EMSL instruments and computing platforms between the years of 1997 and 2000 is expected to accumulate at a very rapid rate. It is expected that the DBCS-1 system will hold in excess of 40 terabytes of data by the year 2000. (The reader should note that DBCS-0 will be used for archived data in years 1995 and 1996. This data will be moved to DBCS-1 after deployment).

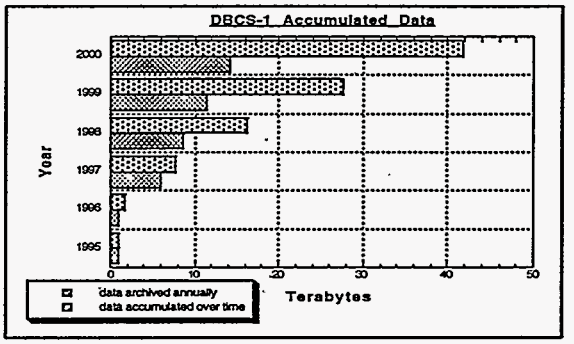

Figure 3. DBCS-1 accumulated data

While it is difficult to predict the total number of files and file sizes that will comprise this vast amount of data, a model based on existing files has been developed. This model accounts for files of various sizes that correspond to a class of storage. For this discussion, files correspond to one of four categories based on their size as follows:

- small files - files sized from 1 kilobyte to 50 megabytes
- medium files - files sized from 51 megabytes to 500 megabytes
- large files - files sized from 501 megabytes to 1

gigabyte

- very large files - files sized from 1 gigabyte to multi-gigabyte

Figure 4 provides a graphical representation of the files that will likely exist in DBCS-1. This figure identifies the volume of DBCS-1 capacity that a particular category of files will consume, and identifies the quantity of files that constitute each category. The medium-performance class of storage will house small- and medium-sized files and the high-performance class of storage will house large- and very-large files.

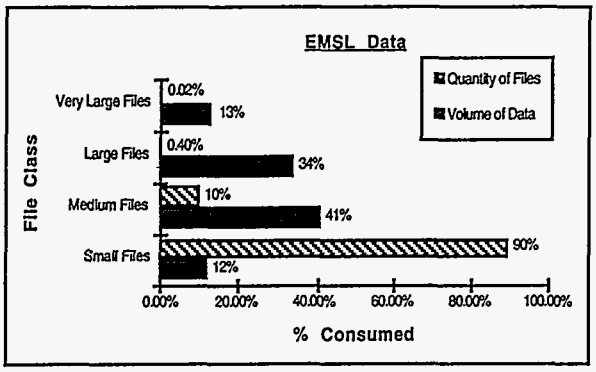

Figure 4. EMSL data: quantity versus volume

Once the system is "populated" with user data, it is expected that the access pattern of DBCS-1 users in EMSL will exhibit periods of peak activity for store and/or retrieve operations. Based on intuition, possible peak period times may be in the early morning when users first arrive at work, near lunch time, and at the end of the day just before users leave work. DBCS-1 must provide sufficient performance during a peak period of access by:

- providing the throughput required to meet users' storage and retrieval demands
- being capable of maintaining a sufficient amount of free space on the DBCS-1 disk cache(s) to accommodate the amount of data stored/retrieved (e.g., reduce the I/O bottleneck that exists between the disk cache and removable media levels within a hierarchy)

Scientific Data Management Software

The architecture, hardware, and HSM software described above provides the foundation for a scientific data management system. However, the data management requirements of scientific applications and instruments in EMSL are only partially satisfied by hardware and driver software. To fulfill EMSL data management requirements, a software system that integrates a database management system with the mass storage system must be developed. This section describes a core component of the scientific data management software development.

Scientific data management researchers and scientists are aware of shortcomings of using file systems, such as the UNIX file system, for long-term storage of data files. A few of the reasons file systems are inappropriate for managing scientific data files are their insufficient storage capacity and long-term unreliability. Mass storage systems offer solutions to storage capacity limitations and provide reliable long-term storage of data files. When considering data file management for scientific applications, however, the issue is not only the storage of data files but also the efficient access, browsing, and retrieval of data files and their contents. With nothing more than file system type functionality, the onus of file management falls upon the scientist; essentially requiring scientists to spend an ever increasing portion of their time *managing* (storing, organizing, searching, and retrieving) data files. Some examples of the things we have observed scientists doing in attempts to manage their data are:

1. Describing a data file's contents in the file name: The file 18c6_Cs_2water_631pgs_hybrid.log describes molecules (8c6_Cs_2water) and experiment parameters(631pgs_hybrid).
   Problems: Most systems have a limited file name length. Not all metadata can be placed within a file name. For example, the file name given does not indicate the application that produced the file (NWCHEM). It is difficult to develop a naming convention that will convey all metadata that might be of interest. Encrypted file names inhibit the sharing of data files.
2. Describing file format (file *type*) in file name extension: A file named ethane.car would be interpreted as Biosym CAR formatted file.
   Problems: Extensions don't distinguish different versions of a single format. File format alone doesn't describe the data contained within the file or the conditions that were used to produce the file. Some extensions, such as ".log", are used by many legacy applications. Output file format for many scientific applications is determined by input parameters, so a single file extension per application would not sufficiently describe a file's format.
3. Spending a significant amount of time searching for data values:
   One chemist, when asked for an optimized ethane molecule geometry, took 15 minutes searching through 3 directories and 2000 files for less than 1K of data.
   Problems: Eventually any data search resorts to scanning a set of files to determine the one that contains the data (file names alone are not selective enough).
   Our project, as other projects (Intelligent Archive[2],

OPTIMAS[3,4]), is attempting to address some or all of the shortcomings not solved by HSM software alone. Our approach, as in other approaches, uses a database management system to fill in gaps not usually handled by mass storage systems. We present our approach as an evolution from managing files, to adding simple file-type metadata to files, to full integration with an experiment management data model. Our development activities, however, do not follow the evolutionary path, but are targeted directly at the final integration of experiment management data model with combined database-HSM software.

One of the obvious goals in designing the integrated database-HSM software layer is to create a design that is independent of underlying technologies. The underlying technologies that we are required to include are the UNIX file system, the HSM software, and the Object-Oriented database management system software.

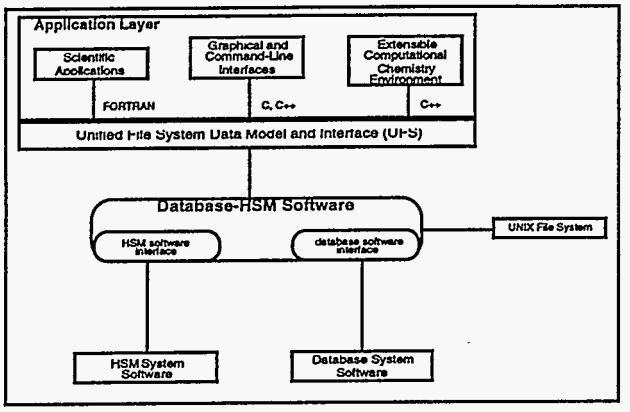

Figure 5. High-level architecture for Database-HSM (DBHSM) software

An overview of our high-level design is given in Figure 5. The uppermost layer, or application layer, provides the sole access path to the Database-HSM (DBHSM) system. We provide command-line utilities similar to UNIX file system commands, graphical user interfaces, and applications interfaces (APIs) in C++(including C access) and FORTRAN. All end-user and administrator utilities are built upon a single API to the DBHSM component. The single DBHSM API is expressed as an object-oriented data model of a unified file system (UFS). We will describe the UFS data model in greater detail in the following section.

Between the external applications and the underlying third-party components is the DBHSM software layer we are currently developing. The DBHSM component manages external requests, handles the translation of requests to underlying component interfaces, and provides management capabilities that enhance the underlying system's capabilities. Underneath the DBHSM software is the software provided by the archive vendor, OODBMS vendor, and the UNIX file system. Currently the underlying software systems are NSL-UniTree and ObjectStore.

Any combined mass storage-database system strategy for scientific data management must address the problem of maintaining the consistency and integrity of information duplicated in both underlying systems. In our approach, the HSM and database software is not under our control; therefore, we are required to maintain the consistency between files placed in the mass storage system and information contained in the database system. The only reasonable way to guarantee the integrity of the system is to require that all access to the DBHSM system occur only through our interfaces. If, for example, data is placed into the mass storage system through the HSM software rather than through the UFS interface, it falls outside of our system. We have chosen to design a UFS interface as an object-oriented data model. Three key advantages of using an object-oriented data model as our system's interface are:

1. We can describe the functional requirements any underlying HSM or database software must provide for our architecture to work.
2. The approach described above enables us to design a system where we can later extend the system to work with other HSM or database system software.
3. We can change or extend implementations without affecting applications and minimizing the effects on the DBHSM software.

Although our UFS data model design is still evolving, we provide a sketch of our current progress.

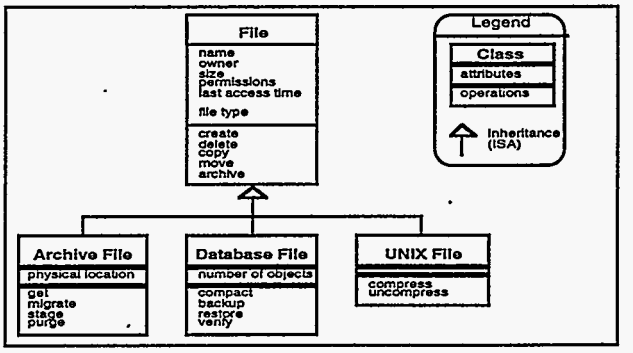

Figure 6. Unified File System data model

Figure 6 gives a data model description of the UFS data model in Rumbaugh[6] notation. First, we observe that any underlying system, mass storage system, database, or UNIX file system, contains a common list of capabilities and attributes. For example, all three systems have:

- the concept of Files and Directories
- attributes - name, owner, size, permissions, and last

access time
- functions - create, delete, copy, move, and archive.

Attributes and operations common to all systems are modeled by a superclass that defines the capabilities required of all file systems. The next level in the UFS data model defines the requirements for each functional type of underlying software system. For HSM system software, this is shown as adding:
- attributes - physical location
- functions - get, migrate, stage, and purge

Similarly, files in the OODBMS must add:
- attributes - number of objects (contained in the database)
- functions - compact, backup, restore, and verify

Our database system software, ObjectStore, uses C++ as the data-definition language (DDL) and data-manipulation language (DML) for the database management system, so C++ is our natural choice for implementing our UFS data model. By implementing the UFS in C++, we immediately have the ability to make file objects persistent and, therefore, have specified the "file-metadata" we will manage in the DBMS.

The UFS data model addresses what we refer to as **file-metadata;** that is, data that we choose to track in the database system that mirrors the data managed by file systems. Even if we were to only handle the file metadata, we could still provide more efficient tools than the UNIX file system. Specifically, tools for searching large groups of files with complex selection criteria would benefit from the query capabilities of the OODBMS. With indices, for example, the following query could be made quite efficient:

find all experiments conducted by chemist "I.M. Curious" before Jan. 1, 1990, that involved crown ether combined with Cesium.

It should be apparent that, for large numbers of files spread across multiple file systems, a OODBMS optimized query would be more efficient than a UNIX "find" command . When combined with files placed into the mass storage system, we expect queries involving file-metadata to give the scientists even greater storage and retrieval speed benefits. One more important feature to note is, because these queries are dependent only on attributes found in the abstract file specification, they are valid independent of which system the file is stored in. Based on the **File** part of the UFS data model, our DBHSM software provides a tracking and reporting software component. The tracking and reporting software provides tools to:
- generate administrative reports containing: space usage, frequency of access
- generate user reports
- notify users or administrators when files are fetched, files are deleted, unauthorized access occurs.

Using file-metadata and the UFS data model we have immediately gained complex querying capabilities, independence from underlying HSM and OODBMS software, the ability to easily extend the system, and a model upon which to base many useful utilities.

While there are benefits in handling file-metadata, the data model approach allows us to also capture metadata about the contents of data files. As our data model diagram shows, we require each data file to contain a *type* attribute. The type attribute is a key to the format contained within the file, and is quite similar to the definition of MIME types. However, file types can specify either a format standard (including version) or the name and version of an instrument that produced the data. When data files of new formats get introduced to the system, the instrument or format get added to a system maintained list. This extensible list allows system administrators and end-users to add formats as they see fit. The delivered system will provide a list containing over 40 computational chemistry codes and their formats used at PNL as well as entries for the instruments housed in the EMSL. Based upon the definition of file format, the DBHSM software provides a component capable of importing and exporting different file formats. The Import/Export module uses an object-oriented data model representation of scientific data types as the intermediate format when importing and exporting data.

An example of the import and export capabilities we will provide is the ability to parse the output file from a computational chemistry code (NWCHEM) to retrieve the optimized geometry of a molecule. Here is an example of a file-type table that describes the contents (parseable data objects) and the procedures to invoke when importing data from a NWCHEM output file:

file-type	NWCHEM Output
object / parser	Optimized Geometry / parse_geom
	Molecular Orbitals / parse_mo
file-type	ARGUS Input
object / parser	Geometry / export_geom

From the parsed geometry, the Import/Export module can write the geometry into an input file for an analysis program such as ARGUS. The same list that enumerates the file formats contains user or system-defined parsers; one for each embedded data type. For example, the computational chemistry code NWCHEM would have the parsers shown in the table above. The entry would also contain export routines for writing the same data types (objects) into the NWCHEM input file format. Eventually, we would expect the UFS data model to be extended to handle new file formats. Instead of requiring new formats, parsers, and exporters to be registered in our *type-table*, sophisticated users would create new sub-types of the **File** class in the UFS data model. The parsers and

exporters would become the operations of the new class and attributes unique to each file format could be added.

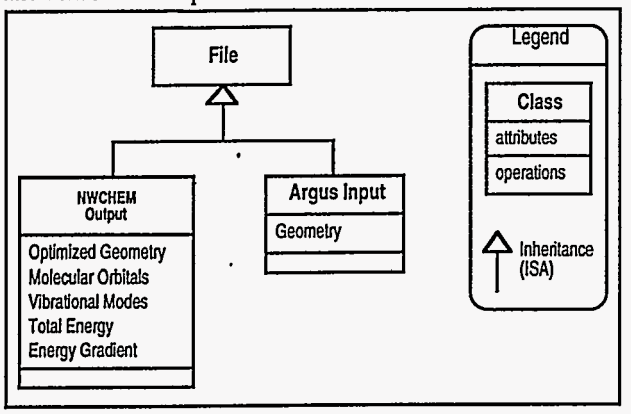

Figure 7. Extending the UFS data model

Figure 7 depicts the extended UFS data model with a new type of file called **NWCHEM Output**. The attributes are the logical objects contained in each file. Accessing those attributes would invoke the parsing operations. In our example, the Import/Export software would request the **optimized geometry** attribute of a **NWCHEM Output** file and use it to set the **geometry** attribute of the **Argus Input** file. Adding file types to our file-metadata approach adds the benefits of extending data file searches to include the application or instrument that created them and forms the foundation for inter-application data transfer.

We have justified the storage of what could be called "redundant" information in our database. There is a higher purpose, in terms of describing a file's metadata, for developing the UFS data model. In a coordinated effort, another team of developers at PNL are developing the Extensible Computational Chemistry Environment (**ECCE'**). ECCE' is a desktop workstation software system for experiment management. This effort is similar to earlier work in Desktop Experiment Management[5] The ECCE' software will provide an extensible framework allowing scientists to set up, run, analyze, browse, combine, and query complex experiments. Initially ECCE' focuses specifically on high-performance computational chemistry experiments. These experiments can range from attempting to determine the optimal geometry of a simple molecule to computing a protein dynamics trajectory describing different conformations of a protein folding. These are exactly the experiments (instruments) that are producing files being placed into the archive.

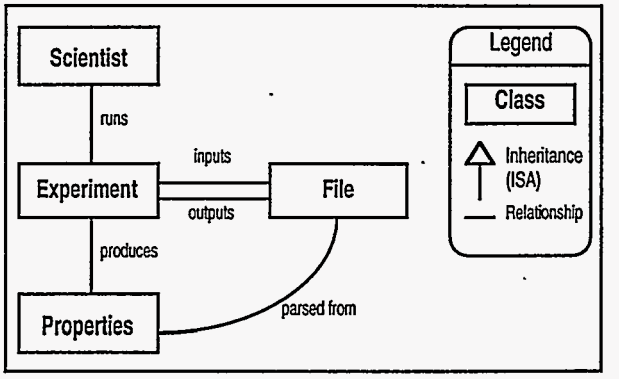

Figure 8. Extensible Computational Chemistry Environment data model

Figure 8 shows a simplistic overview of both the EMSL experiment management data model and the EMSL chemistry data model. These data models are dependent on the UFS data model and functions for managing data files. We are faced with two fundamental requirements in managing experiments:

- Capture sufficient information to be able to exactly reproduce the experiment, and
- Capture enough metadata to allow scientists to make their own judgments regarding the validity and quality of the experimental results.

The experiment management data model shown in the figure is an abstraction of the current EMSL data model. It captures the basic notions that a **scientist** conducts an **experiment** that produces a number of **results**. In the process of producing those results, the experimental instrument may take **files** as input and may produce files containing the results as output. The joining of an experiment management data model to the UFS data model expands the scope of queryable attributes tremendously. For instance, scientists may now have access to their results without having to access output files, because key results contained in output files have been duplicated as objects in the OODBMS. Queries about classes of experiments based on the parameters used or the experimental conditions are then possible.

We are currently engaged in detailed design of the DBHSM software component. Both the abstract portions of the UFS data model (**File** objects) and the experiment management data model have been designed and implemented. Within the next six months we will have a completed, extensible file typing system and sufficient means to register new applications and instruments. The software developed on DBCS-0 will be ported to DBCS-1 after delivery and installation in the summer of 1996.

ACKNOWLEDGMENTS

Pacific Northwest Laboratory is a multiprogram national laboratory operated for the U.S. Department of Energy by Battelle Memorial Institute under contract DE-AC06-76RLO 1830.

The National Storage Laboratory is a collaborative effort of various industry partners and DOE Laboratories at Lawrence Livermore National Laboratory (LLNL). LLNL is operated for the U.S. Department of Energy under contract W-7405-Eng-48.

BIBLIOGRAPHY

[1] P. Berard, "Value Added Data Archiving", *Proceedings, Third NASA Goddard Conference on Mass Storage Systems and Technologies*, NASA Conference Publication 3262, October, 1993.

[2] C. Hunter, "Intelligent Archive: Integrated Information, Application, and Metadata Management at the Scientist's Desktop", *http://www.llnl.gov/liv_comp/ia.html.*

[3] L.T. Chen, R. Drach, M. Keating, S. Louis, D. Rotem, A. Shoshani, "Efficient Organization and Access of Multi-Dimensional Datasets on Tertiary Storage Systems", *Information Systems*, 19(4), 1994.

[4] L.T. Chen, D. Rotem, "Optimizing Storage of Objects on Mass Storage Systems with Robotic Devices", *Extending Database Technology* , 1994.

[5] Y. Ioannidis, M. Livny, E. Haber, R. Miller, O. Tsatalos, J. Wiener, "Desktop Experiment Management", *IEEE Data Engineering Bulletin*, 16(1), March 1993.

[6] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, *Object-Oriented Modeling and Design*, Englewood Cliffs, New Jersey: Prentice-Hall, 1991.