Database Support for Computational Chemistry

Judith Bayard Cushing

Oregon Graduate Institute
Department of Computer Science
and Engineering
19600 N.W. von Neumann Drive
Beaverton, OR 97006-1999 USA

Technical Report No. CS/E 92-001

May 12, 1991
February 24, 1992 (revised)

Database Support for Computational Chemistry

Judith Bayard Cushing

May 12, 1991
January 23, 1992 (revised)

Abstract

In common with other computational science applications, computational chemistry applications have the need both for increasing the speed of calculations and for storing and viewing large amounts of specialized information. This paper addresses the latter of these two needs: a review of current literature in scientific data management shows computational chemistry to be fairly typical among scientific applications in its use of flat files as opposed to database systems. Good initial choices for certain input parameters would greatly improve the individual chemist's research efficacy and even the accuracy and performance of these computationally intensive experiments. Unfortunately, because of the high data management overhead, computer-readable results that could help in setting up future experiments are typically discarded once an experiment is complete. While the object-oriented paradigm appears adequate for modeling the required computational chemistry information, specific database technology necessary for implementing the application may as yet be lacking. To test this hypothesis, I performed both a conceptual database design and a functional specification for a computational chemistry database browser. This report describes those specifications, and identifies some challenges and research opportunities suggested by the computational chemistry information model.

1 Introduction

Computational chemistry is one of the emerging computational sciences that bring together applied mathematicians and computer scientists with scientists from domains such as environmental science, biology, chemistry or physics. The computational sciences have in common a need for (1) increasing the speed and precision of computation so that operations not now practical can be performed, (2) promoting the sharing of scientific data, and (3) providing better support for the individual scientist's research activities, e.g., help in managing an increasingly high volume of data, providing visualization and analysis facilities, and promoting easily used computer program libraries. This report describes an effort to learn how computational chemists conceptualize and use scientific structures, preparatory to exploring how database technology can ameliorate the problems stated above.

Computational chemistry applications, an area to which database technology has not to my knowledge been applied, share many of the critical characteristics of scientific data management identified by the 1990 NSF Workshop on Scientific Database Management (see Figure 1). Clear specifications of the required modeling and behavioral functionality for computational chemistry will help indicate if current development and research efforts in computer science address the needs of scientific data management in general. If scientific data management needs are not being met, then such studies as this one, spread over several scientific disciplines, could lead to data structures and data management capabilities and policies adequate for many sciences.

$\boxed{3}$User Interfaces
$\boxed{3}$More Flexible Representational Structures
$\boxed{2}$Appropriate Analysis Operators
$\boxed{2}$Special Concurrency Support
$\boxed{3}$Data Citation Standards
$\boxed{3}$Data Interchange Standards
$\boxed{3}$Metadata
$\boxed{2}$High Volume, Multi-Level Store, Indefinite Retention
$\boxed{1}$Fast(er) Dataset Transmission
$\boxed{3}$Comparability of Multiple Datasets
$\boxed{1}$Interoperability of Multivendor DBMSs
$\boxed{3}$Quality Assurance of Datasets

A rating of 3 indicates issue highly relevant to Computational Chemistry, 2 relevant, and 1 only slightly relevant.

Figure 1: Scientific Database Issues as per 1990 NSF Workshop

Computational chemistry applications, an area to which database technology has not to my knowledge been applied, share many of the critical characteristics of scientific data management identified by the 1990 NSF Workshop on Scientific Database Management (see Figure 1). Clear specifications of the required modeling and behavioral functionality for computational chemistry will help indicate if current development and research efforts in computer science address the needs of scientific data management in general. If scientific data management needs are not being met, then such studies as this one, spread over several scientific disciplines, could lead to data structures and data management capabilities and policies adequate for many sciences.

This report covers the information modeling phase for an *ab initio*[1] computational chemistry database project. The objective of this work was to establish that conceptual structures from this application area stretch the object-oriented paradigm and suggest new object-oriented structures that might be generalized to other scientific applications, perhaps to other heterogeneous distributed data intensive computing environments. In effect, I propose to develop more useful information structures and programmatic interfaces for computational chemistry based on observational analyses of how scientists working in this field use existing programs and information structures. I have worked closely with computational chemist David Feller at the Applied Physics Center of Battelle Pacific Northwest Laboratories, observing and analyzing tasks performed both by chemists and programs. To the best of my knowledge, the data management requirements of this application area have not been described from the point of view of computer science.

Sections 2 and 3 present related work, implementation alternatives for scientific databases, and a brief argument for using an object-oriented database for developing an exploratory prototype in this application area. Section 4 includes a functional description of tasks routinely performed by computational chemists and goes on to outline the computational chemistry database project. Section 5 presents the information model in detail. Finally, Sections 6, 7, and 8 outline opportunities for future research offered by the study of computational chemistry data and show how the proposed research could contribute to the application area in the longer term. The proposed database prototype meets some immediate needs of chemists, coincidental to giving computer science researchers insight into data structures and data access patterns.

[1] *Ab initio* computational chemistry involves the computation of chemical properties from first principles alone.

2 Related Work

The Invitational NSF Workshop on Scientific Database Management held in March of 1990 brought together about forty computer scientists and domain scientists to address data management problems facing scientific researchers. The workshop report corroborates other work in the scientific database area: many scientists still, by and large, manage their information through programs reading and writing flat files. Almost every scientific domain has an extensive software investment in programs (usually FORTRAN) that use flat files and that have evolved over a number of years. While database management systems would ultimately improve the reliability, availability, concurrency levels, and programmability of scientific applications (just as for any application area), scientists now attempting to use databases find that current technology does not match their needs. Even if current technology were adequate, changing from flat file access to database access would involve retraining highly skilled programmers and extensive conversion of existing programs and files to database representations [FJP90a, FJP90b].

Molecular scientists have been among the first to use private and public databases, perhaps because this data is (at least on the surface) easily represented as ASCII character strings [Bur89, Boa90, FB90]. While the past four years have seen "only" a 7-fold increase in the number of nucleotides in the centralized DNA databases (from 3 million to 21 million), and the data is accumulating at "only" 7 million nucleotides per year, automated sequencing methods promise to increase this rate by an order of magnitude [Wat89]. Of particular interest are the Human Genome project [LPS90] and three protein structure databases Compo-OWL [BW90], BIPED [Gar89], and P/FDM [GPKF90], each of which attempts to use different methods to represent similar structures. Other work in protein sequencing [FB90, HS86, LWS87, PL88] gives insight into the interplay between parallel algorithms and innovative data structures.

Most work by computer scientists concerning scientific databases can be classified as (1) general characterizations of scientific databases [Olk86b, Olk86a, SOW84], (2) studies of scientific data structures [Bel83, BP87, Bel88], and (3) specifications of future data-intensive scientific application systems, such as the Earth Observing System [Che90, Doz90]. Other database research areas, such as geographical information systems [WK90], temporal data structures [GS90, SS88], and statistical databases [Gho88, RS90a, RR90, KR88] exhibit important similarities to scientific databases.

Within the mainstream of database research, special attention should be paid to innovative work in two areas. (1) Both semantic and object-oriented data modeling techniques are important to scientific data management because of the importance of representing complex scientific objects and their associated behavior independently of any physical schema [BS85, MD90, AG89, BBMA89, Mai89]. (2) Research on personal databases and laboratory notebooks is relevant because of the interplay between an individual scientist's private databases and laboratory-wide or public databases [LM84, Wei89, BB87].

Scientific applications of interest to the research outlined in this paper fall into three categories: computationally intense programs, scientific visualization systems, and data interchange programs. (1) A basic understanding of commercially available and public domain programs by domain scientists is relevant to scientific data management research because commonly used programs define the relevant data modeling entities of interest and will constitute the ultimate clients for a data repository. Commonly used programs that perform *ab initio* chemistry computations include Gaussian (which includes a data browser capability) [Gau89], GAMESS [GAM90], HONDO [Dup90], and MELDF [F$^+$91]. (2) Scientific visualization tools are also likely clients for scientific data repositories; many chemists use graphical molecular display and editing packages such as Tektronix' CAChe [CAC90] and Chem3D [Che89] to prepare molecular structures input to computational programs.

Specialized toolkits such as Daylight Tools [DAY91] and AVS Chemistry subsystems [Van90] that allow visual rendering of molecular structures are available to the developer of application programs in this area. (3) A scientific data repository system should be prepared to accept data from and generate data for data exchange formats commonly used in the respective scientific domains. Like all scientists, computational chemists typically share both programs and data. Data interchange efforts strive to make data prepared for interchange "self describing" and thus readable by heterogeneous systems. The NetCDF project, aimed at atmospheric data, is an example of just such an effort [Ful91].

3 Paradigm Choices for Scientific Databases

The relational model, developed by Codd and based on set-theoretic notions, organizes data into flat relations called tables [Cod88]. Relationships between tuples (or rows of tables) within two or more relations are represented via shared attribute values. Relationships are typically materialized via joins. Even though typical scientific data management transactions require many more joins than typical business transactions, relational database systems do offer two distinct advantages over object-oriented database systems: (1) The relational model is well-understood, well-grounded in theory, and well-documented. (2) Mainstream relational systems are robust. Unfortunately, however, the relational model seems inadequate for directly representing and implementing some important structures and functionalities required for scientific applications, such as ordered sequences and temporal relationships. Supporting large complex transactions, representing spatial information, and maintaining complicated ancillary information (also known as "metadata") are difficult to provide in an application that runs on top of a relational system [FJP90a, FJP90b, MS90]. Several DNA research centers are attempting to reap the advantages of the relational model and overcome its deficiencies by using a relational system for a data repository with a more flexible object-oriented system as the interface to application programs. In addition to the obvious disadvantages of maintaining two database designs and two databases, this approach risks losing information when moving data from the object-oriented database back to the relational [Pec91].

Protein sequence databases, used to predict and compare protein structures, can be stored in relational systems, but for performance reasons implementers often introduce considerable data redundancy (with fewer tables come fewer joins), thus undermining the strict relational approach. Queries not anticipated at the database design stage and basic behavioral functionality both present difficulties. For example, because of the difficulty of combining complex calculation with retrieval, the Biped database implemented in Oracle is used simply to store and retrieve protein sequences. In a separate step, programs read the textual output of Biped's Oracle queries, perform complex calculations, and only then display superimposed motifs [Gar89, GPKF90]. For examples of shortcomings of the relational paradigm specifically with respect to computational chemistry, see Section 6.1.

While many scientific database researchers believe that relational systems as now implemented are inadequate to the task at hand, they feel that the following options are still too immature or too expensive to use for major implementation efforts [FJP90a, FJP90b, Pec91, SOW84]: special purpose data management facilities [BW90], extended relational [Jag89, RS90b, H^+90], extensible tool kits [C^+90], logic databases [GPKF90], and object-oriented databases [AMKP90, GH91, ZM90].

4 The Computational Chemistry Database Project

To explore the applicability of object-oriented database management systems for managing scientific data, computer scientists from the Scientific Database Laboratory at the Oregon Graduate Institute and computational chemists and computer scientists at the Molecular Sciences Research Center (MSRC) and Applied Physics Center (APC) at Battelle Pacific Northwest Laboratory (PNL) in Hanford, Washington, have developed an initial working relationship. *Ab initio* computational chemistry has been identified as a specific initial research area within which to explore the applicability of emerging database technology to high performance scientific applications. David Feller, an active researcher in computational chemistry and a co-author of the computational chemistry program MELDF, is the primary domain scientist collaborator at PNL[DF86, FBD87, FD90, F$^+$91]. After evaluating several object-oriented database systems for use in this application area, I intend eventually to generalize results to other scientific domain areas.

4.1 Computational Chemistry

The emergence of quantum theory during this century is still changing the ways in which chemists work. As early as 1929, it was clear that quantum chemistry calculations could predict molecular properties and structure, but most scientists believed that adequately precise calculations would be impossible. By the 1950's, approximate methods had been developed, but most scientists felt even these would be impractical for molecules of any size or complexity. Until recently, *ab initio* methods have been of interest primarily to theoretical chemists: only semi-empirical methods, considerably less accurate, could be used for molecules larger than 50-100 atoms. Rapid increases in computing power, however, will soon make *ab initio* methods applicable to much larger molecules, including those of interest to molecular biologists [Sal87, Cam70, Lev83, PB70].

The computational chemist's laboratory is his or her computer, where numerical experiments based on quantum theoretical models compute chemical properties, i.e., structure, dynamics and molecular properties, for a molecule under investigation. *Ab initio* molecular orbital methods apply quantum mechanical techniques to molecular structure and energetics, solving the Schöedinger equation to various levels of approximation. Most chemists construct each experiment in an iterative manner and use more than one library of programs, since each offers slightly different capabilities. The chemist will rerun the programs several times, adjusting various parameters and tuning the programs for each molecule studied [Fel91].

An investigation is over when the chemist has completed a run that adequately models the molecule in question. An admitted problem with the current methodology is that little or no information associating parameters with molecules is captured from the experimental process in a way directly applicable to later experiments with other molecules. In addition, the chemist's work on even relatively small molecules (many fewer than 50 atoms) is hampered by the amount of information he or she can store, search, and manage effectively. Being able to run experiments on molecules significantly larger than 50-100 atoms (as is expected in the next five years) will exacerbate these data management problems.

4.2 User Scenario for a Computational Data Browser

In this section, I describe a typical scenario for the use of the computational database and its browser which are central architectural components of the chemist's workbench. (See Figure 2.)

1. Using a molecular structure editor (and templates of similar molecules from the database), the chemist will graphically define the subject molecule within a "private" database (*aka* a

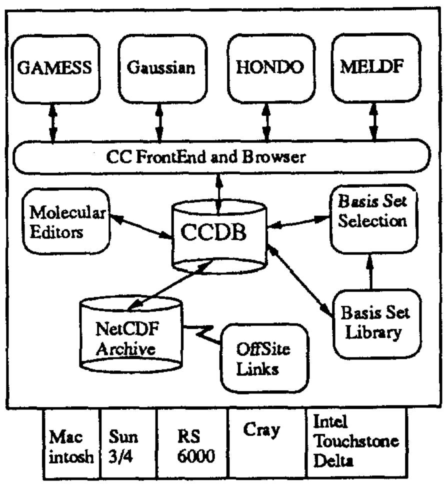

Figure 2: Computational Chemistry Data Browser Architecture

Personal Laboratory Notebook). The chemist may also perform simple structure optimizations. During the course of the investigation this geometry will likely be radically changed. The chemist will usually save only the final, optimized structure for a molecule. Sometimes a sequence of structures, for example those associated with a chemical reaction, may be of interest.

2. The chemist may consult the database for computational and laboratory experiments on molecules of similar structure and properties. Selected experiments may then be loaded into the personal database. Because the chemist may not be familiar with all of the computational programs or apparatus on which the selected experiments were run, associated data must be available to help interpret input parameters and results. Since data may be physically stored in the database, archived at the site, or resident at another location, the chemist must be presented with a consistent view of data irrespective of its physical location or the format or machine on which it was generated or is stored.

3. While consulting the public database, the chemist may determine that important information from the literature ("property data" [SOW84, Dub91]) is missing. The chemist should be able to add such information to the public database, assuming of course some quality assurance measures.

4. Using the molecular properties in which the chemist is interested and the experiments retrieved above, the browser then assists the chemist in choosing an appropriate set of parameters, a highly specialized and critical activity. Since at least some of the computational chemistry programs likely to be encountered by a scientist wishing to perform calculations will be little more than research devices, it may be difficult for a "bench" chemist[2] to properly prepare input, due to the combination of mathematical sophistication and research orientation of the codes. The consequences of a mistake may be a failed run, a much-too-long run or, what is undoubtedly worst, a run that produces seemingly plausible results that are inherently incorrect.

[2]The term "bench chemist" refers to an experimentalist (i.e., nontheoretician) involved in synthetic work or spectroscopic analysis.

The database will hold input templates for each computational program, as well as methods for converting to formats appropriate to a particular parameter from the internal formats. Using templates and information gleaned from previous "like" experiments, the browser will assist the chemist in determining an appropriate first cut set of parameters to use, as well as a target machine on which to run the the experiment.

5. Using the first cut set of parameters and a set of codes that meet the chemist's needs, an available target machine on which to run the experiments is selected. The chemist is given an estimate of how long the computational experiment will run and (if relevant) how much it will cost. As a result of this information, the chemist may further optimize the molecular structure or modify parameters before asking the system to schedule and run the experiment. Once the experiment is run, results are placed in the chemist's personal database.

6. The above process may be repeated many times before the chemist is satisfied with the results. The computational experiment may be run on several sets of codes, and the results compared with each other and with those of previous experiments. A browser interface to analysis packages such as "S" [BCW88] would be ideal, but even the side by side viewing of different computational experiments would be helpful. The database system must provide the capability of capturing results from different programs in comparable form viewing and analysis.

7. Once the chemist has successfully completed an investigation and published the results, data may be added to the public database or sent to colleagues at outside laboratories.

A successful computational chemistry experiment results in long term storage of only about two megabytes of data, but may be preceded by as many as hundreds of "unsuccessful" runs, each resulting in the short term storage of about two megabytes. The personal database, a kind of "laboratory notebook", is proposed to hold this work in progress. In addition, a run can generate up to 5 or 10 gigabytes of intermediate data, written temporarily to disk, and used in the solution of the problem or restarting an interrupted run. A laboratory such as Battelle's Environmental and Molecular Research Laboratory at Hanford now generates per year about 1.2 gigabytes of data that are candidates for permanent archiving. This laboratory-wide scientific reference material should be accessible by casual and off site users, and will likely be archived or compressed. In addition, data from other chemistry laboratories will be imported to this repository.

4.3 Computational Chemistry Data Browser

The initial project is to provide a database and browser for the chemist who uses *ab initio* techniques. While no current database system adequately addresses all anticipated needs, existing database technology could effectively enhance the chemist's computing environment and give computer scientists important insight into additional database functionality that may be needed. A prototype database browser will enable chemists at PNL to query data for both ongoing and archived experiments. Presently, once work on a particular molecule has been completed these data are for all practical purposes discarded, primarily because maintaining all the data for each experiment represents too great a data management overhead for the researcher. Data management barriers also hinder comparison of runs even within the same investigation.

A prototype data browser should offer the following immediate benefits to the chemist: (1) repository for computational chemistry experiment data (until the data can be archived), (2) repository for computational chemistry experiment metadata (for browsing archived data and online data),

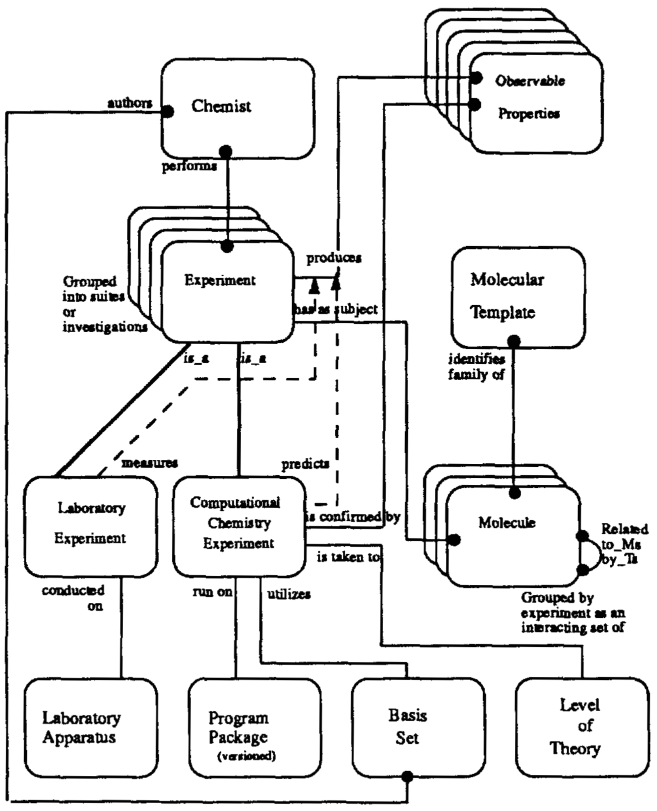

Figure 3: Computational Chemistry Data Model

and (3) help in the selection and development of parameters for computational chemistry programs. Support for chemists performing computational chemistry experiments involves: setting up a personal database corresponding to the current set of experiments; side-by-side viewing of parameters and results for a number of runs, or of the current experiment and those in the main repository; comparison and analysis of computational results with those of laboratory experiments; and assistance in archiving successful results in the main repository.

For the reasons given in Section 3 above, current relational database systems seem unsuitable for this application class. Object-oriented database technology shows promise for handling the data types and queries for supporting these computational experiments. The requirements analysis of the chemist's information needs are here presented independently of any particular object-oriented database management system. From this information model specific (physical) database schema can be developed. Once it has been determined whether the object-oriented database paradigm fits the application in general, a chief research objective will be to determine any inadequacies of the paradigm and to design extensions as needed.

5 Computational Chemistry Database Information Model

Figure 3 mirrors my current overall understanding of the information model for computational chemistry experimentation. The model is expanded and elaborated in sections that follow.

5.1 Chemists and Experiments

The model includes simple identifying information about chemists whose experiments or basis sets are included in the database so that users can contact a chemist about his or her work.

Experiments are arranged in an *is-a* hierarchy. An experiment, perhaps a collaborative effort of more than one chemist, is either a laboratory experiment or a computational chemistry experiment. Experiment attributes include simple identifying information such as name, e.g., a run title or a textual annotation, date-begun, date-completed, and meta-information such as a citation for the data used as source of the experiment. Experiments will be accessed through links with root entities chemist or molecule.

5.2 Computational Experiments

A chemist performs a computational chemistry experiment on a given molecule using a program package, specifying as many as 200 parameters, including initial estimations of molecular structure, basis set and level of theory. Selection of input parameters is critical not only to how rapidly the numerical experiment will run, but also to the accuracy of the results. The syntax of parameters varies from program to program and sometimes from one version of the programs to the next.

Computational chemistry experiments can require, as input, somewhere between 50 and 200 "numbers" and produce, as output, several thousand additional "numbers". Between the input and output may lie several trillion other "numbers" needed to solve the mathematical equations. Among the "numbers" that appear in the output are some that correspond to physically observable properties of a molecule, and some that are simply artifacts of the equations that were solved. The most important output value is the total energy of the molecule. This energy can be determined by a variety of different techniques, e.g., Hartree-Fock self consistent field theory or configuration interaction theory. An example of an artifact of solving the equation is the set of molecular orbitals, an array of numbers whose size varies approximately with the square of the number of atoms in the molecule. Although molecular orbitals cannot be physically observed, they can be combined with input parameters to produce a map of the electron density about a molecule and, as such, have meaning beyond the particular program that produced them [Fel91].

Molecular properties are generated directly from the detailed molecular structure derived in an experiment. An experiment is "successful" if these computed properties agree with properties measured in laboratory experiments, e.g., by x-ray crystallographic methods.

Chemists study many kinds of combinations of the atoms making up a molecule. A "suite of experiment" is an aggregation of computational experiments that reflects such combinations. For example, in studying transition states from hydrogen and oxygen to water, four computational experiments might be grouped together into a suite of experiments (see Figure 4): Two initial experiments model the stable states of hydrogen and oxygen molecules. An intermediate experiment models the unstable state of these elements at the energy level required for the transition. A fourth experiment models the final and stable state of the water molecule. This suite of computational experiments predicts molecular properties of the hydrogen and oxygen atoms at the points where the energy values, computed and stored as "results" of the computational experiment, are minimal (at the stable states), and maximal (at the unstable state). Energy curves, however, are not typically smooth as depicted in Figure 4; an improper or careless determination of molecular structure or other parameter can cause the computation to converge only to a local minima or local maxima.

A series of computational experiments can also be grouped into an "investigation", as follows: In the course of computationally determining a molecule's structure, a chemist performs several experiments on that molecule; only a few (perhaps one or even none) will ultimately be archived.

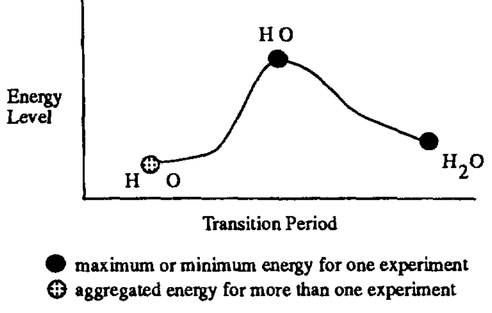

Figure 4: Transition of hydrogen and oxygen to water.

Until deemed "successful", such experiments should be marked as private and available only to the performing chemist(s). In the course of such an investigation, the chemist uses results of intermediate experiments to identify trends, isolate local minima, and then iteratively tune input parameters. Storing inputs and results of different iterations of an investigation supports the ongoing experimental process. (For a discussion of how the model supports the startup of an investigation, i.e., the selection of initial parameters, see Section 5.7 below.)

Viewing results of two or more runs side by side would constitute an improvement over the existing practice. Translucent display of molecular orbitals over the original or optimized molecular structure would further aid the chemist in the investigation.

Another measure of the accuracy of the computational experiment is the amount of numerical error. Stability analysis techniques are generally too difficult for a theoretical or bench chemist to apply to a particular set of experiments, although authors of commercially available programs usually perform a general stability analysis for the method. If the experiment object contained some measure of the cumulative error introduced by each state of the calculation, an associated browser could upon request perform a rudimentary stability analysis. Stability analyses from different runs could then be compared and contrasted statistically.

5.3 Laboratory Experiments

A laboratory experiment is conducted in a "traditional" chemistry laboratory usually by a bench chemist, using laboratory apparatus such as a mass spectrometer or a cloud chamber, and producing a value or values for a specific observable property.

Laboratory experiments are relevant to computational experiments in at least two ways: (1) A computational chemist typically validates a computational chemistry experiment by comparing the calculated molecular properties to properties physically observed through one or more laboratory experiments, usually performed by another chemist, perhaps at another site. Whether a laboratory experiment "agrees with" a computational experiment is typically a matter of judgement, not formal analysis. (2) On occasion a laboratory experiment may yield a property value that does not seem to agree with accepted theory. The bench chemist might then set up a computational experiment to explore the apparently anomalous laboratory result.

5.4 Molecules

A molecule is the subject of one or more laboratory and computational chemistry experiments, and can be identified by name or chemical formula or through its corresponding experiment. Molecule

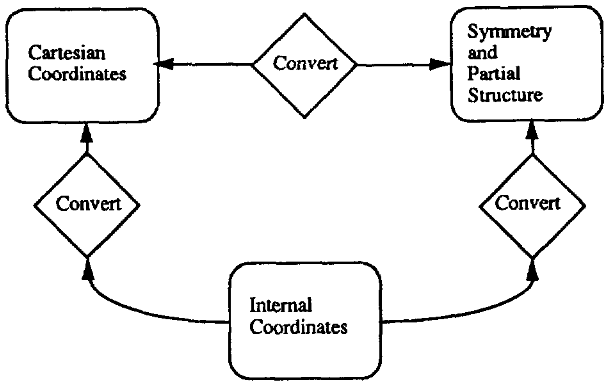

Figure 5: Molecular Structure Representations.

name and chemical formula are examples of information that is extraneous to the computational experiment *per se* but is included for the chemist's use when retrieving data on molecules. Some attributes of the molecule (e.g., name, chemical formula, atomic weight) are functions of the atoms or isotopes comprising the molecule, and, as such, are also independent of a particular experiment. Other attributes, in particular molecular structure, are highly dependent on a particular experiment.

For a computational experiment molecular structure is the only necessary information regarding a molecule. Molecular structure, the location of the atoms in the molecule, identifies the molecule to the application program. Different programs require different representations; sometimes the same program may even require different representations depending on the function to be calculated. Molecular structures are represented in three basic formats:

1. Structure. Three-dimensional Cartesian coordinates, atomic mass and charge are specified for each atom in the molecule.

2. Partial structure with symmetries. Here, only the locations of symmetry-unique atoms of the structure are specified; others are calculated using symmetry rules.

3. Internal coordinates, sometimes called "optimized structure". The molecular geometry is specified using bond lengths and angles instead of Cartesian coordinates. A well-optimized structure can significantly increase the speed and accuracy of an experiment. There is no unique set of internal coordinates corresponding to a given set of Cartesian coordinates, but by conforming to any one of a number of conventions which fix the positions of the first few atoms in the molecule with respect to a fixed Cartesian axis system, it is possible to define the remainder of the molecule in terms of a variety of internal coordinates [Fel91].

A single comutational experiment typically uses only one of the structure representations listed above, but a chemist performing a suite of experiments for different chemical states of the "same" molecule will create several instances of that molecule, each potentially with a different molecular structure. There is no reliable algorithm for converting partial structure or Cartesian coordinates to optimized structure (internal coordinates), but with extreme care *ab initio* programs can be used to optimize structures for use in (other) *ab initio* programs. (See Figure 5.) For further discussion of this issue, see Section 5.8.

The meaning of "molecule" differs depending on context, and often varies with respect to the stage of a particular computational experiment. A suite of computational experiments looking at different states of a molecule must have as subject several spatial variants of that molecule, though always for the same collection or "bag of atoms". Each distinct view of the same molecule necessitates a different conceptual representation of the same physical information: A spatial variant of a molecule is one of several molecular structures in which the atoms of a molecule arrange themselves when in different chemical states. A "bag of atoms" is a collection of atoms and isotopes that carries atom- and isotope-specific information, grouped together by virtue of being the constituent parts of the subject of a suite of experiments. An atom associates a given atomic symbol with a certain nuclear charge. Atomic mass is taken from an instance of isotope of the atom in question.

5.5 Molecular Template

To browse the database for candidate basis sets, a chemist will want to examine experiments run on similar molecules. A molecular template is a way of characterizing a list of similar molecules, or a "molecular family", and defines ordered aggregations of molecules. A given molecule can of course match a number of templates and thus belong to a number of families. A molecular template is a structure much like a molecule that can be rendered either graphically or textually and be matched against molecular structures in the database. A chemist will usually choose to view an instance of a template graphically, in color, rather than textually, as "C_xH_x".

5.6 Program Package

Each computational experiment is run on a versioned instance of program package. A version of program package refers to a release of the program package, compiled by a particular version and release of a compiler, on and for particular platforms. Program packages are thus represented as a version hierarchy of program package, release, architectural platform, and compiler version. For example, a computational experiment might utilize the GAMESS package, release 2.0, compiled under version 3.21 of the Sun 3 FORTRAN 77 compiler. Important meta-information is stored at each level, for example, basic formatting and functional capability of the program package at the highest level, changes in format at the release level, and performance characteristics at the compiler version and platform level. (See Figure 6.)

5.7 Basis Sets and Levels of Theory

A basis set is a set of real functions in three-dimensional space. Basis sets are in effect artifacts of solving the Schröedinger equation, and might have no correspondence to the symmetries in the geometry of a molecule. Basis sets are used in describing the electron density about the molecule. Although the solution of the Schröedinger equation does not explicitly require their use, the overwhelming majority of quantum chemistry techniques are formulated assuming that a suitable basis set is available [Fel91].

A large number of basis sets are in popular use, and these can be categorized according to the families of molecules for which they "work". Determining which basis set to use is an extremely complicated process, even for theoretical chemists. For a new investigation, it is possible that no known basis set is appropriate and the chemist will need to develop his or her own.

The level of theory corresponds to the degree of accuracy with which the motion of the electrons around a molecule will be described. Higher levels of theory usually result in better agreement with experiment, but can cost much more in terms of computer time. Computational chemistry experiments generally become increasingly accurate with "better" basis sets and "higher" levels of

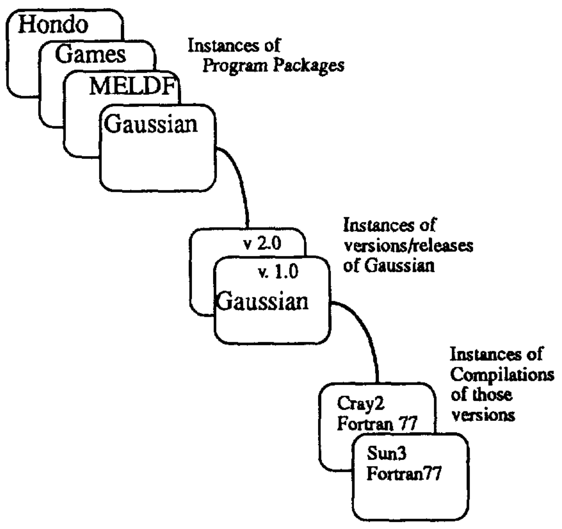
Hondo
Games
MELDF
Gaussian
Instances of Program Packages
v 2.0
v. 1.0
Gaussian
Instances of versions/releases of Gaussian
Cray2 Fortran 77
Sun3 Fortran77
Instances of Compilations of those versions

Figure 6: Program Package Aggregation and Version Hierarchy.

theory, albeit with exponentially higher computational demands and diminishing degree of accuracy. However, choices for level of theory and basis set are not independent: an experiment run with a lower level of theory and more primitive basis set could give more accurate results than one run with higher levels of theory and better basis sets if respective discrepancies in the former case cancel out each other. Basis sets could be paired with levels of theory that "work" for a family of molecules, and thus, in principle, arranged hierarchically .

Formats of both basis sets and levels of theory are specific to the program package chosen.

5.8 Observable Properties

Observable properties for a given molecule are a function of an experiment, not the molecule itself. They are represented as property-unit-value triples and grouped according to the experiment that produced them. Additional property types can be added to the database at any time; data dictionary facilities should support and control this function. Examples of observable properties include:

1. Hydrophobicity.
2. Polarizability.
3. Hyper-polarizability.
4. Anisotropicity.
5. Bond Measurements.
    (a) Bond Distances. Measured in nanometers or Angstroms. Physically measurable by xray diffraction.
    (b) Bond Angle. Measured in degrees, e.g., carbon might be 109. Physically measurable by xray diffraction or spectroscopic techniques.

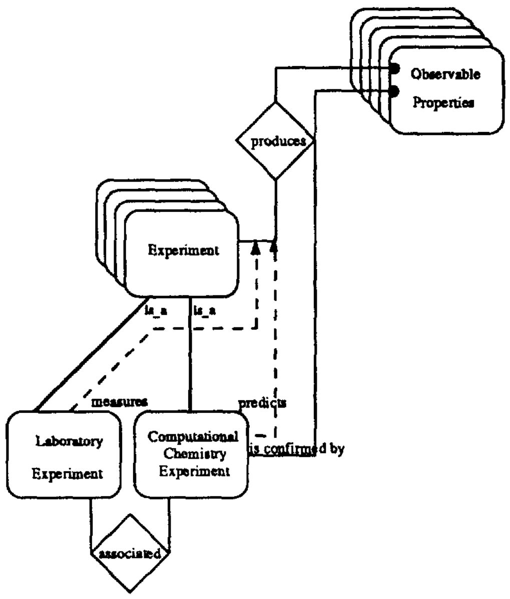

Figure 7: Functional Relationships — Experiments and Properties.

(c) Bond Dipole Moment (Bond Polarity). A measure of the distribution of electrical charge, weighted by mass. Measured in nuclear distance (esu/cm). Bond polarity affects the bonding characteristics of the molecule and is physically measured by ultraviolet spectroscopy.

(d) Bond Energy. Measured in kcalories/mole.

(e) Bond Frequency. Measured in hertz.

The number of property-unit-value triples associated with a given experiment can be a function of the size of the molecule and its structure, e.g., the number of bonds. Properties can be categorized according to whether they refer to atom, bag of atoms, molecule, bond, or atom pair. A molecule might have one bond angle for every pair of bonds, one bond distance per pair of atoms bonded, and one measure of polarizability per molecule.

A computational experiment typically produces many observable properties, but a laboratory experiment usually only one, e.g., polarizability or bond distances. The relationship between experiment and a set of observable properties is modeled by a (stored) function "produces" that, given an experiment and a property, returns a set of unit-value pairs. The relationship between a computational chemistry experiment and the observable properties confirming it is modeled by a stored function "associated" that, given a computational experiment and a property, returns a laboratory experiment. The "produces" function can again be applied, this time to the laboratory experiment, to retrieve the observable properties confirming that computational experiment. (See Figure 7.)

Comparability of observable properties is an exacting task. Computational programs differ greatly in the units they use to report properties and the assumptions they make in calculating properties. While units typically can be inferred from the particular code package used to run that experiment, assumptions are typically a function of the basis functions used.

The information that would be necessary to pin down the results of one program vs. another is quite detailed. Molecular orbitals (MO's) are but one example: The orbitals represent one of several "n^2" data sets, that to an outsider consist of nothing but n^2 double precision numbers. Each of n MO's is a vector of length n. Each number in the vector is a coefficient multiplying a basis function. To make sense of an arbitrary set of MO's stored in the database, one needs to know which basis functions were used and in what order. For example, knowing that the "STO-3G" basis set was used to generate a set of MO's for water is necessary but not sufficient. One must also know the correspondence between the MO coefficients and the basis functions. Program RHFSCF in MELDF lists the basis function and the atom on which it sits. In some cases even that doesn't really pin things down. One might also need to know how the basis functions were normalized. [Fel91]

6 Challenges

The information model described above presents particular challenges to the database system in which the corresponding physical schema is implemented. In addition, there remain nuances within the information model itself that require further work if the database is to represent accurately the computational chemistry research enterprise.

6.1 Challenges to the Target Database System

The computational chemistry information model confirms suspicions that currently available relational products will not meet the needs of this application class. For example, abstract data structures such as lists and bags are needed to model experiments, molecules and atoms. Furthermore, without matrix and vector representations, modeling computational chemistry results and basis sets will be needlessly complex. Families of molecules should be ordered using a molecular template into lists according to some goodness of fit criteria. Currently available commercial relational products do not support abstract data structures such as lists, bags, vectors and matrices.

Note also that the "predicts" and "confirms" relationships, linking computational and laboratory experiments to observable properties constitute derived attributes for these two kinds of experiments. Another derived value of importance is program performance, calculated using data spread across disparate entities: program package, experiment, basis set, size of molecule, level of theory. I believe that derived attributes of this sort are considerably easier to model and implement in the object-oriented paradigm than in the relational.

6.2 Remaining Challenges in Information Modeling

Using an object-oriented product for building the computational chemistry database will alleviate some of the problems outlined above. However, key challenges remain before a computational chemistry database system can build an infrastructure for future research. These challenges include:

1. Semantically overloaded terms. The application abounds with highly context-dependent terms, each use of which is really a different aggregation or view of an entity in the model. Molecule and experiment are prime examples of this overloading.

   Sometimes "molecule" means any collection of specific atoms. For example, "Retrieve all experiments on the water molecule" means retrieve all experiments for which the molecule entity has two hydrogen atoms and one oxygen atom. "Molecule" can also mean particular spatial groupings of atoms associated with the input of an experiment; these spatial groupings can be in any of three (or more) formats. Sometimes, a molecule display should include the electron cloud, and sometimes not.

Molecular structure in terms of bond lengths and angles is both an attribute of the molecule (and hence an input to a set of programs) and an observable property (an output from a program package). The output of one computational experiment could be used as the input to another, although this is not a straightforward process:

> If one performs a calculation on water with Gaussian and wants to use the molecular orbitals in GAMESS, one must have some detailed knowledge of the format in which Gaussian stores orbitals and the conventions it follows. The orbitals have a meaning beyond the bounds of a particular code: They are mathematical entities resulting from the way the Schröedinger equation is solved. In principal it is possible, but difficult, to compute orbitals with one program and use them in another. First, one needs the equivalent of the metadata. Second, one then must construct a conversion utility to reformat the orbitals from one program's format to another. [Fel91]

"Experiment" sometimes means one run of a program package, sometimes a set of runs modeling states of one molecule, sometimes a set of runs modeling transition from one or more molecules to one or more different molecules. "Experiment" can also be an aggregation of any of these, in the sense that when solving a molecular structure, the chemist groups a number of runs together. I have used the terms "experiment", "suite of experiments", and "investigation" to denote these different senses of the term "experiment", but there is considerable subtlety remaining in how this term is used by the practicing scientist.

2. Representing relationships or groupings within a domain. A single instance of Molecule, Basis Set, and Computational Experiment can be grouped into one or more collections. For example, a molecule can belong to none, one, or several different families of molecules depending on how many molecular templates it matches and how well. A single instance of Basis Set could belong to several basis set hierarchies, and be placed at a different level in each hierarchy depending on how well it solves the structure problem for several families of molecules. The energy levels of several computational experiments, for some queries, might be aggregated and reported as a single value; see, for example, the energy values of hydrogen and oxygen depicted in Figure 4 in Section 5.2.

3. Populating the database. Once a physical schema has been prepared, there remains the problem of populating the database with enough data to test the schema design. For an initial verification, the database can be populated by hand. However, populating the database by hand with enough instances to give chemists a realistic feeling for its use, or to measure performance, is not a practical alternative. Automating the database population, however, presents significant challenges: Few computational experiments are archived with all the meta information needed to populate the database. Data for experiments run on different programs have not been converted to common units.

7 Future Work

Opportunities for future research in this area grow directly out of the information modeling phase. Once an initial database is built, populated, and in preliminary use, the software infrastructure will be in place for pursuing research in one of several areas:

1. Generalizing results to other application areas. Bell's work with particle physics programs suggests that at least some of the underlying structures will be similar enough to abstract more general (higher level) types and operations [Bel88]. Our own work suggests that, within computational chemistry itself, structures for holding data from *ab initio* programs could be generalized to semi-empirical programs. Computational chemistry structures bear similarity to materials science and other molecular sciences.

2. Object-oriented database system issues. Effectively generalizing a model containing complex data types and associated behaviors requires better data modeling constructs for specifying the logical database designs than are now available.

3. User interface design issues. Can a data modeling environment rich enough to describe computational chemistry data types and associated operations be made simple enough for a practitioner to use when accessing data or designing a personal research database? Important areas for research are the design of effective user interfaces and visualization and analysis support for large amounts of scientific data within specific application areas such as computational chemistry.

4. Database system interface to operating systems and compilers. Recent advances in processing power have rendered this application class I/O intensive as well as computationally intensive [HS86]. Information about the structure of the persistent data could perhaps be passed from the database to the operating system to enhance the performance of application programs by better data caching policies.

   Computational chemistry programs have not yet been effectively parallelized, but there is evidence that the data could be partitioned to this end [Fox90, Tay91]. Many scientific data collection and analysis tasks involve the parallel treatment of large numbers of similar data items. Compiler methods of data dependence might be extended to include analysis of persistent data, providing hints to the compiler for efficient partitioning of data into parallel processors and reconstruction of results from parallel computations.

8 Conclusion

This report describes an initial information model for a computational chemistry database browser, the first phase of a longer-term research project to explore data management support for this application area. I have thus far gained a basic understanding of the structure of molecule and experiment objects and confirmed the hypothesis that computational chemistry is a fertile field for research in scientific data management. Work to date also corroborates the hypothesis that computational chemistry applications seem better suited to object-oriented database management systems than to current relational systems. Several aspects of the information model will require further exploration, however, and may prove difficult to represent using even the relatively flexible object-oriented paradigm.

Implementation of the information model in an object-oriented database system and of a rudimentary data browser is targeted for the immediate future [CMR91]. Releasing a populated database to the chemists and exploring possible interfaces of computational chemistry utilities and programs will give us insight into the data access patterns of this application class. Preliminary efforts suggest four general areas for future research: generalizing computational chemistry data types, object-oriented data modeling extensions to cover behavior as well as structure, user interface issues specific to scientific data management, and enhancing the performance of computational chemistry applications through database interface to the operating system and compiler.

9 Acknowledgements

This work is supported financially by grants from the Oregon Advanced Computing Institute (OACIS) and Battelle Pacific Northwest Laboratory's (PNL) Molecular Sciences Research Center (MSRC) and Applied Physics Center (APC).

Individuals at both PNL and OGI provided ongoing guidance: From PNL, James Thomas provided the original spark that ignited my interest in computational chemistry, and Michael DeVaney the absolutely essential week to week suggestions. Their encouragement and support is gratefully acknowledged. The information model results largely from length discussions with computational chemist David Feller of PNL and my advisor computer scientist David Maier of OGI. The vision they hold of how a computational chemist might benefit from better data management facilities has been invaluable. Any errors in the model are, of course, due to my own ignorance of computational chemistry and object-oriented data modeling.

References

[ACM89] ACM. *Proceedings ACM SIGMOD*, volume 18, New York, June 1989. ACM Press.

[AG89] R. Agrawal and N. H. Gehani. ODE (Object Database and Environment): The language and the data model. In *Proceedings ACM SIGMOD* [ACM89], pages 36–45.

[AMKP90] H. Afsarmanesh, D. McLeod, D. Knapp, and A. Parker. An extensible object-oriented approach to databases for VLSI/CAD. In Zdonik and Maier [ZM90], pages 607–618.

[BB87] J. Biskup and H. H. Bruggemann. The personal model of data - towards a privacy oriented information system. Technical report, Hochschule Hildescheim, Hildescheim, West Germany, 1987.

[BBMA89] A. Borgida, R. J. Brachman, L. McGuinness, and L. Alperin Resnick. CLASSIC: A structural data model for objects. In *Proceedings ACM SIGMOD* [ACM89], pages 58–67.

[BCW88] R. A. Becker, J. M. Chambers, and A. R. Wilks. *The New S Language.* Wadsworth and Brooks-Cole, 1988.

[Bel83] J. L. Bell. *Data Structures for Scientific Simulation Programs.* PhD thesis, University of Colorado, Boulder, CO, 1983.

[Bel88] J. L. Bell. A specialized data management system for parallel execution of particle physics codes. In *Proceedings ACM SIGMOD*, volume 18, pages 277–285. ACM, ACM Press, June 1988.

[Boa90] Computer Science and Technology Board. *Computing and Molecular Biology: Mapping and Interpreting Biological Information, a CSTB Workshop.* NRC, Washington, DC, 1990.

[BP87] J. L. Bell and G. S. Patterson, Jr. Data organization in large numerical computations. *The Journal of Supercomputing*, 1:105–136, 1987.

[BS85] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9:171–216, 1985.

[Bur89] C. Burks. Genbank: Current status and future directions. Technical Report LA-UR-89-1154, Los Alamos National Laboratory, Los Alamos, NM, April 1989.

[BW90] A. J. Bleasby and J. C. Wootton. Construction of validated, non-redundant composite protein sequence databases. *Protein Engineering*, 3(3):153–159, 1990.

[C+90] M. J. Carey et al. The EXODUS extensible DBMS project: An overview. In Zdonik and Maier [ZM90], pages 474–499.

[CAC90] Tektronix, Beaverton, OR. *CAChe: Modeling for the Experimental Chemist*, 1990.

[Cam70] J. A. Campbell. *Chemical Systems.* Freeman, San Francisco, 1970.

[Che89] Cambridge Scientific Computing, Inc., Cambridge, MA. *Chem3D*, 1989.

[Che90] R. M. Chervin. High performance computing and the grand challenge of climate modeling. *Computers in Physics*, pages 234–238, May/June 1990.

[CMR91] J. B. Cushing, D. Maier, and M. Rao. Computational chemistry database prototype: Objectstore. Technical Report CS/E-92-002, The Oregon Graduate Institute, Beaverton, OR, 1991.

[Cod88] E. F. Codd. A relational model of data for large shared data banks. In Michael Stonebraker, editor, *Readings in Database Systems*, pages 5–15. Morgan Kaufmann, San Mateo, CA, 1988.

[DAY91] Daylight Chemical Information Systems, Irvine, CA. *The Daylight Toolkit*, 1991.

[DF86] E. R. Davidson and D. Feller. Basis set selection for molecular calculations. *Chemical Review*, 86:681–696, 1986.

[Doz90] J. Dozier. Looking ahead to EOS: The earth observing system. *Computers in Physics*, pages 248–259, May/June 1990.

[Dub91] B. Dubrovsky. Universal data access for time series analysis. *Pixel*, pages 42–44, March/April, 1991.

[Dup90] M. Dupuis. *HONDO-8 User's Guide.* IBM Center for Scientific and Engineering Computations, Kingston, NY, 1990.

[F+91] D. Feller et al. *MELDFX User's Guide.* Molecular Science Research Center, Battelle Pacific Northwest Laboratories, Richland, WA, April 1991.

[FB90] J. W. Fickett and C. Burks. Development of a database for nucleotide sequences. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 1–35. CRC Press, 1990.

[FBD87] D. Feller, C. M. Boyle, and E. R. Davidson. One-electron properties of several small molecules using near Hartree-Fock limit basis sets. *Journal of Chemical Physics*, 86(6):3424, 1987.

[FD90] D. Feller and E. R. Davidson. Basis sets for *ab initio* molecular orbital calculations and intermolecular interactions. In K. B. Lipkowitz and D. B. Boyd, editors, *Reviews in Computational Chemistry*, pages 1–43. VCH, New York, 1990.

[Fel91] D. Feller. Personal communications, Dec 1990 - Apr 1991.

[FJP90a] J. C. French, A. K. Jones, and J. L. Pfaltz. NSF scientific database management workshop (final report). Technical Report TR-90-21, University of Virginia, Charlottesville, VA, August 1990.

[FJP90b] J. C. French, A. K. Jones, and J. L. Pfaltz. NSF scientific database management workshop (panel reports and supporting materials). Technical Report TR-90-22, University of Virginia, Charlottesville, VA, August 1990.

[Fox90] D. J. Fox. Gaussian 90, applying quantum chemistry to biology. *Cray Channels*, pages 19–21, Fall, 1990.

[Ful91] D. W. Fulker. Unidata strawman for storing earth-referencing data. *Proceedings of the Seventh International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography, and Hydrology*, pages 210–217, 1991.

[GAM90] Department of Chemistry, North Dakota State University, Fargo, N.D. *GAMESS User's Guide*, 1990.

[Gar89] J. M. Thornton and S. P. Gardner. Protein motifs and database searching. *TIBS*, 14:300–304, July 1989.

[Gau89] Gaussian, Inc., Pittsburgh, PA. *Browse: Quantum Chemistry Database System*, 1989.

[GH91] R. Gupta and E. Horowitz, editors. *Object-oriented Databases with Applications to Case, Networks, and VLSI/CAD.* Prentice Hall Series in Data and Knowledge Base Systems, Englewood Cliffs, New Jersey, 1991.

[Gho88] S. P. Ghosh. Statistical relational model. In Rafanelli et al. [RKS88], pages 338–355.

[GPKF90] P. M. D. Gray, N. W. Paton, G. J. L. Kemp, and J. E. Fothergill. An object-oriented database for protein structure analysis. *Protein Engineering*, 3(4):235–243, 1990.

[GS90] H. Gunadhi and A. Segev. Temporal query optimization in scientific databases. In Ozsoyoglu [Ozs90], pages 27–34.

[H$^+$90] L. M. Haas et al. Starburst mid-flight: As the dust clears. *IEEE Transactions on Knowledge and Data Engineering*, 2(1), March 1990.

[HRSP86] W. J. Hehre, L. Radom, P. Schleyer, and J. A. Pople. *Ab Initio Molecule Orbital Theory.* Wiley, 1986.

[HS86] W. D. Hillis and G. Steele, Jr. Data parallel algorithms. *CACM*, 29(12):1170–1183, December 1986.

[Jag89] H. V. Jagadish. Incorporating hierarchy in a relational model of data. In *Proceedings ACM SIGMOD* [ACM89], pages 78–87.

[KR88] J. C. Klensin and R. M. Romberg. Statistical data management requirements and the SQL standards. In Rafanelli et al. [RKS88], pages 19–38.

[Lev83] I. N. Levine. *Quantum Chemistry.* Allyn and Bacon, 1983.

[LM84] P. Lyngbaek and D. McLeod. A personal data manager. *Proceedings of the International Conference on Very Large Databases*, August 1984.

[LPS90] S. Letovsky, R. Pecherer, and A. Shoshani. Scientific data management for human genome applications. In Ozsoyoglu [Ozs90], page 51.

[LWS87] R. H. Lathrop, T. A. Webster, and T. F. Smith. ARIADNE: Pattern-directed inference and hierarchical abstraction in protein structure recognition. *CACM*, Vol 30, No 11:909–921, November 1987.

[Mai89] D. Maier. Why isn't there an object-oriented data model? Technical Report CS/E-89-002, Oregon Graduate Institute of Science and Technology, Beaverton, OR, 1989.

[MD90] F. Manola and U. Dayal. PDM: An object-oriented data model. In Zdonik and Maier [ZM90], pages 209–215.

[MS90] D. Maier and J. Stein. Development and implementation of an object-oriented DBMS. In Zdonik and Maier [ZM90], pages 167–185.

[Olk86a] F. Olken. Physical database support for scientific and statistical database management. Technical Report LBL-19940 (rev2), Lawrence Berkeley Laboratories, Berkeley, CA, May 1986.

[Olk86b] F. Olken. Scientific and statistical data management research at LBL. Technical Report LBL-21623, Lawrence Berkeley Laboratories, Berkeley, CA, June 1986.

[Ozs90] Z. M. Ozsoyoglu, editor. *Data Engineering (Special Issue on SSDBMS)*, volume 13, Washington, DC, September 1990. IEEE Computer Society.

[PB70] J. A. Pople and D. L. Beveridge. *Approximate Molecular Orbital Theory.* McGraw-Hill, 1970.

[Pec91] R. Pecherer. Personal communication, March, 1991.

[PL88] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. (U.S.)*, 85:2444–2448, April 1988.

[RH70] W. G. Richards and J. A. Horsley. *Ab initio Molecular Orbital Calculations for Chemists.* Clarendon Press, Oxford, 1970.

[RKS88] M. Rafanelli, J. C. Klensin, and P. Svensson, editors. *Fourth International Working Conference on Statistical and Scientific Database Management (SSDBM)*, volume 339. Springer-Verlag, June 1988.

[RR90] M. Rafanelli and F. L. Ricci. A visual interface for statistical entities. In Ozsoyoglu [Ozs90], pages 35–44.

[RS90a] M. Rafanelli and A. Shoshani. STORM: A statistical object representation. In Ozsoyoglu [Ozs90], pages 12–18.

[RS90b] L. A. Rowe and M. Stonebreaker. The Postgres data model. In Zdonik and Maier [ZM90], pages 461–473.

[Sal87] L. Salem. *Marvels of the Molecule.* VCH Publishers, Inc., 1987.

[SO89] A. Szabo and N. S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory.* McGraw-Hill, 1989.

[SOW84] A. Shoshani, F. Olken, and H. K. T. Wong. Characteristics of scientific databases. *Proceedings of the Tenth International Conference on VLDB*, pages 147–159, August 1984.

[SS88] A. Segev and A. Shoshani. The representation of a temporal data model in the relational environment. In Rafanelli et al. [RKS88], pages 39–61.

[Tay91] P. Taylor. Computational chemistry well-suited to supercomputers. *On Line*, 13(1):1–3, January 1991.

[Van90] M. VandeWettering. The application visualization system — AVS 2.0. *PIXEL*, July/August 1990.

[Wat89] M. Waterman. Foreword. *Bulletin of Mathematical Biology*, 51(1):1–4, 1989.

[Wei89] R. F. E. Weissman. In search of the scholar's workstation: Recent trends and software challenges. *Academic Computing*, pages 28–31,59–64, September 1989.

[WK90] A. J. Westlake and I. Kleinschmidt. The implementation of area and membership retrievals in point geography using SQL. In Ozsoyoglu [Ozs90], pages 4–11.

[ZM90] S. B. Zdonik and D. Maier, editors. *Readings in Object-oriented Database Systems.* Morgan Kauffman, San Mateo, CA, 1990.