LBL-17582

c.2

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Computing Division

Presented at the 10th International Conference on Very Large Data Bases, Singapore, August 27-31, 1984

CHARACTERISTICS OF SCIENTIFIC DATABASES

A. Shoshani, F. Olken, and H.K.T. Wong

April 1984

LBL-17582 c.2

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

LBL-17582

Characteristics of Scientific Databases

Arie Shoshani, Frank Olken, and Harry K.T. Wong

Computer Science Research Department
University of California
Lawrence Berkeley Laboratory
Berkeley, California 94720

April, 1984

This research was supported by the Applied Mathematics Sciences Research Program of the Office of Energy Research, United States Department of Energy under contract DE-AC03-76SF00098.

LBL-17582

CHARACTERISTICS OF SCIENTIFIC DATABASES

Arie Shoshani, Frank Olken, and Harry K.T. Wong

Computer Science Research Department
University of California
Lawrence Berkeley Laboratory
Berkeley, California 94720

ABSTRACT

The purpose of this paper is to examine the kinds of data and usage of scientific databases and to identify common characteristics among the different disciplines. Most scientific databases do not use general purpose database management systems (DBMSs). The main reason is that they have data structures and usage patterns that cannot be easily accommodated by existing DBMSs. It is the purpose of this paper to identify the special database management needs of scientific databases, and to point out directions for further research specifically oriented to these needs.

In the past, we have studied "statistical databases", which are databases that are primarily collected for statistical analysis purposes. We have found that the observations and techniques developed for statistical databases are useful for scientific databases. The reason that common characteristics exist is that many scientific databases are often subject to statistical analysis. However, we found that scientific databases have additional stages of data collection and analysis that induce more complexity.

We discuss the different types of scientific databases, and list the properties identified for them. Ten examples are then analyzed with respect to the types of data and their properties, and summarized in two tables. Conclusions are drawn as to the preferable data management methods needed in support of scientific databases.

1. INTRODUCTION

This document is a result of numerous interviews with scientists, mostly from Lawrence Berkeley Laboratory, spanning several different scientific disciplines. The purpose of these interviews was to examine the kinds of data and usage of scientific databases in order to identify common characteristics among the different disciplines.

Most scientific databases do not use general purpose database management systems (DBMSs). The main reason is that they have data structures and usage patterns that cannot be easily accommodated by existing DBMSs. It is the purpose of this document to identify the special database management needs of scientific databases, and to point out directions for further research specifically oriented to these needs.

In the past, we have studied "statistical databases", which are databases that are primarily collected for statistical analysis purposes. A summary of work in statistical databases can be found in [Shoshani 82]. We expected that some of the observations and techniques developed for statistical databases will be useful for scientific databases. Indeed, we found this to be the case. The similarities are pointed out throughout this document where appropriate. It should not be surprising that common characteristics exist, because many scientific databases are often subject to statistical analysis. However, as discussed below, scientific databases have additional stages of data collection and analysis that introduce more complexity and challenges.

In section 2, different types of scientific databases are described. In order to describe the common characteristics between several example applications, a list of properties for the different types of data are given in section 3. In section 4, several examples of scientific applications are delineated with respect to the list of properties. The properties of these example applications are summarized in two tables, which appear at the end of this document. In section 5 we discuss the implications of our observations to desirable database techniques for scientific databases, and proposes areas for further investigation. Section 6 is a short summary section.

2. TYPES OF SCIENTIFIC DATA

The scientific databases described to us during the interviews were analyzed in order to identify similar data structures, data characteristics and data usage among different applications. We found it convenient to distinguish between different types of scientific data. The important features for each type were identified, and different examples of scientific data were categorized accordingly. This categorization helped us to identify the most common characteristics and usage of scientific data, and provided the direction for future research. In section 4 several examples of scientific applications are described in order to demonstrate the different types of scientific data. In this section we describe the data types and their main features.

Most scientific data result from experiments and simulations. Data from experiments are usually measurements of some physical phenomena, such as the collision of particle beams, or the spectra generated by molecules in a strong magnetic field. Data from simulations typically result from complex computations derived by using values from the previous time interval. Both experiment and simulation data have similar characteristics, and therefore are considered jointly. In order to simplify the terminology used here, we refer to such data as "experiment data", regardless of whether they are experiment or simulation data. Experiment data can be classified according to three characteristics: regularity, density, and time variation.

1) Regularity

This characteristic refers to the regularity of the points or coordinates for which values are measured or computed. For example, in physics experiments, detectors are placed in a specific configuration. If the configuration describes a regular grid or some other geometric structure, the experiment is said to have (spatial) regularity. Similarly, many simulations assume some regular grid for which values are computed, and therefore have spatial regularity. In addition, if values are measured or computed at regular time intervals, then time can be considered as another regular coordinate of the data.

In general, regularity implies that a mapping between the coordinates of measured values and the storage locations of these values can be made by means of a computation (such as "array linearization", which is simply a mapping from multi-dimensional space to linear space, similar to FORTRAN array mapping). Therefore, in such cases it is not necessary to store the coordinate values with each measured data value, resulting in storage savings and fast random access. On the other hand, when spatial irregularity exists it is necessary to enumerate the data points, and store their identifiers with the data values.

2) Density

This characteristic indicates whether all the potential data points have actual values associated with them. For example, simulation data of fluid motion computed on a regular grid would have data values (for velocity, direction, etc.) computed for each point of the grid, and therefore the data is considered dense. On the other hand, in many experiments a large number of measurements that are below a certain threshold are discarded and never recorded. In fact, the level of sparseness can be quite high, i.e. only a small fraction of the potential data points have recorded values. For example, in physics experiments of colliding particle beams, the measured data is only for resulting sub-particles, which occur over a small portion of the detectors that are distributed in space.

It is important to identify this characteristic because sparsity implies a large number of null values which may be compressed out. The compression technique chosen should depend on the access patterns to the data. Access patterns are discussed in the next section.

3) Time variation

This characteristic refers to the change of coordinates over time; i.e. the points for which data values are measured or computed change their position from one time unit to another. For example, consider some material that is bent in the course of an experiment. Before the experiment starts a set of points is selected for measuring the material's behavior (such as stress, voltage, temperature). During the experiment the selected points may change their position as a result of the bending action. Time variation is a characteristic found mostly in simulations where a mesh of points are allowed to change their position over time during the simulation process. These simulation methods are generally called adaptive mesh techniques.

The reason for considering time variation an important characteristic is that it adds an additional requirement. In addition to storing the coordinates of points for every time interval, it is necessary to maintain the relationships between the points as they existed in the original mesh. This is needed in order to be able to reconstruct the time sequence of points that correspond to the same original point, and in order to find neighboring points to a given point at any given time.

In addition to the experiment data discussed above, there exist data in support of the experiments, and data that are generated from the experiment data. Support data fall into two types which we call configuration data and instrumentation data. Similarly, generated data fall into three types: analyzed data, summary data, and property data. These types are discussed below. To distinguish these additional data types from the experiment data, we refer to them collectively as "associated data".

1) Configuration data

Configuration data are data that describe the initial structure of an experiment or simulation. For example, in simulating heat transfer through buildings, the building layout has to be described. Similarly, the configuration of an experiment describes the position of different devices and detectors. The configuration layout actually determines the regularity (or irregularity) of the experiment data mentioned above. Usually, it does not change in the course of the experiment or simulation. However, it can change between experiments or simulations. It is important to keep track of these changes and to associate the correct configuration data with the corresponding experiment data.

2) Instrumentation data

Instrumentation data consists of descriptions of the different instruments and substances used in an experiment, and their changes over time. This data is crucial for the correct analysis of the experiment data. It includes information such as the pressure and temperature of a gas used in an experiment and their changes over time, drift of voltage over time, and the characteristics of detectors and devices as measured before each experiment or a series of experiments. It also includes the log of experiment operations, such as the time that a defective analog-to-digital converter was replaced, and who was in charge of it. Unfortunately, some of this information is collected into unrelated files and log books, thus making their association with the experiment data a tedious task that is prone to errors.

3) Analyzed data

The previous two data types are essential in order to support the analysis of experiment data. The analysis process produces many databases that also need to be managed along with their relationships to the experiment data and to each other. The analysis process may require several steps. For example, in physics experiments of colliding particle beams, a preliminary histogram over the experiment data can be done in order to estimate parameters that are later used to interpret the calibration data in the next step of the analysis. For each collision, called an event, the tracks of sub-particles produced are reconstructed and kept in a database. From the track data, another database for the event data can be derived, describing the kind of sub-particles produced and their characteristics. Additional steps use databases from this and earlier stages to generate yet more data. It is important to capture the analysis process, the input and output databases of each step, and the relationships between the steps.

4) Summary data

Similar to "statistical" databases, which deal with statistical summaries (aggregations) of data sets, scientific databases are often aggregated. For example, in experiments of heat transfer in buildings, the amount of heat lost or gained can be averaged over several points of a wall, summed over entire rooms, or aggregated over days into months. Another example, is the generation of histograms from many experiments to determine the likelihood of a certain phenomenon. As in the case of statistical databases, there is a need to organize, search and browse collections of summary data, and to preserve their relationship to lower level data from which they were derived.

5) Property data

In any scientific field, the summary of information learned over the years is useful to the community at large. There is a substantial amount of work devoted to the organization and classification of properties of materials, substances, and particles. For example, there are several systems devoted to the staorage and retrieval of chemical substance properties. Many property databases cannot now be accessed on-line. The data is only available in periodically published books, and may not be up-to-date. Property data is non-uniform: it contains numeric, text, and bibliographic data, as well as images and graphs. This is one of the reasons that for each scientific area special purpose systems have been developed. General purpose data management systems that can deal with such diversity of data types are not generally available. In addition, because of the complex terminology involved with such data, sophisticated search and browsing capabilities are needed.

3. PROPERTIES IDENTIFIED FOR SCIENTIFIC DATA

Using the classifications of data types described in the previous section, it was easier to identify common characteristics and usage of the data. For each classification we have looked for certain properties that seem to exist across scientific applications. These properties are described in this section. Since the properties of experiment data are not necessarily the same as those of associated data, they are described separately. In the next section, we describe example applications in terms of these properties. The terms that are used for each property are shown in *italics* letters in the text below. The reader may refer to the leftmost columns of table 1 and table 2 for the list of properties of experiment data and associated data, respectively.

3.1. PROPERTIES OF EXPERIMENT DATA

1) Identifier

The identifier is that part of the data that identifies each data point uniquely (also called a key). In the case of experiment data the identifier is usually a composite key of spatial coordinates and a time coordinate. Since the identifier has multiple dimensions, the properties of regularity, sparsity, and time variation, discussed in the previous section, apply naturally. The concept of a multi-dimensional identifier is similar to that of category attributes in statistical databases [Shoshani 82]. This concept is quite dominant in experiment data (as was the case with statistical databases) because the data is mostly accessed with respect to its identifier. We expand on this point below in the section on access patterns.

The identifier is said to be *regular* if each of its dimensions are ordered in regular intervals; it is *sparse* if only a fraction of the points in the cross product of the dimensions have data associated with them; otherwise it is *dense*. *Time variation* implies that the identifier, regardless whether it is regular or irregular, varies over time.

2) Access pattern

Access pattern refers to the most typical forms of data access. For example, an analysis program may follow a track of a sub-atomic particle, or a simulation program may need its nearest neighbors in order to calculate the next data point. Note that in these examples the access of points is relative to the (spatial) identifier coordinates, and not the measured or calculated data values. This is typical of the access pattern of experiment data. The reason for distinguishing between the different types of access patterns is that they imply different requirements for physical data base organizations, as discussed in the implications section.

We distinguish between two aspects of the access pattern. The *access type* is the type of access of a single query (or a step of the computation). The *access sequence* refers to the relationship between queries, i.e. whether the selection of a query depends on previous queries.

2a) Access type

There are three access types that we found useful to identify. An *exact match* means that the identifier of a point was specified precisely in the query, (Usually, a single point is identified). A *range* type implies that a range of possible points were identified. Since the identifier is multi-dimensional, each dimension is involved in the specification of the range. A *proximity* type indicates that the neighboring points around a given point are desired.

2b) Access sequence

Given a query of a particular type, the access sequence indicates whether the identifier(s) of the next query relate to the identifier(s) of the previous query. A *local* access sequence implies that the identifier(s) of the current query are close to the identifier(s) of the previous query. For example, following a particle track involves a local access sequence, since each successive point is close to the previous point. A *non-local* access sequence means that there is no relationship between the identifiers of successive queries. This means that the points of each query need to be found using a random access search. In a *linear* access sequence, the sequence of the identifiers of successive queries follows successive intervals of the dimensions of the identifier. For example, following the points of a mesh according to the regular intervals of the dimensions of the mesh is considered a linear access. An *arbitrary* access sequence indicates that the order of processing the data points is unimportant. Such access is usually used when the entire data set (or some large subset) need to be processed for analysis or summary statistics.

Access sequence should be thought of in conjunction with access type. For example, searching for a particular point in space where this point is not related to points of the previous query, implies an exact access type and a non-local access sequence. However, searching for a collection of points in the same neighborhood while following a certain path, implies a proximity access type and a local access sequence.

3) Database size

Experiments are often repeated in order to verify a certain phenomena, to determine the statistical behavior of the experiment, or to discover a rare event that occurs only in a small fraction of the experiments. In many cases the results of each experiment can be processed independently. We call each independent part of an experiment a *unit*. An example of an independent unit is a single collision (event) in particle physics, or a single time step calculation of a simulation. It is important to identify such units and to determine their size because they can be processed independently of other units and often in parallel. In addition, if units are small enough they can be processed entirely in main memory, rather than brought piecewise from secondary storage.

Analysis and summarization of experiment data is usually performed over a *collection* of experimental units. The size of a collection is significant because it refers to the quantity of data that analysis queries may need to access. Such queries may select a portion of the collection, or may process the entire set to derive summaries or statistics. A collection may be very large, as is the case with experiments that are run over a period of months because the desired event is rare, because a large number of runs is desirable for statistical analysis, or because extensive parametric studies are desirable.

There is no logical limit to the total amount of data that can be collected by repeating experiments and simulations. The limitations are usually cost and resources. Nevertheless, it is interesting to identify the total amount of data that scientists keep active and available. This category is simply referred to as *total* size. All size figures shown in table 1 are only intended to show order of magnitude.

4) Associated data

The different categories of associated data mentioned here: *configuration*, *instrumentation*, *analyzed*, and *summary*, simply indicate whether such data exists for the different example applications. Note that property data is not mentioned since property data is not usually associated with a single experiment, but rather summarizes data over many experiments.

3.2. PROPERTIES OF ASSOCIATED DATA

We chose to emphasize somewhat different properties for associated data, because their structure and usage is different from experiment data. The access pattern and size properties are similar to those of experiment data, but the identifier properties are more diverse. They are described as part of the data modelling properties. We also added usage properties and non-standard data types.

1) Access pattern

The access pattern properties of associated data fall into similar categories as those of experiment data. However, while access patterns of experiment data refer to accessing data points with respect to their identifiers, the access patterns of associated data are with respect to any attributes, whether they are thought of as identifiers or measured data. The reason is that in associated data the concept of an identifier (or category attributes) is not so dominant. For example, when the experiment data of a particle physics experiment are analyzed, the resulting database represents tracks and events rather than the individual data points. The identifiers of the original data points no longer exist in the analyzed data. Instead the tracks and events may be given an identifying number or some combination of the measured values (such as mass and momentum) may be thought of as the identifier.

The categories assigned to access patterns of experiment data above apply to access patterns of associated data as well. However, we found it necessary to add a *partial* access type, because it is common to access associated data (especially analyzed and summary data) by specifying predicates (selection criteria) only on part of the attributes. For example, finding all particles with a mass in a certain range that generated a certain number of sub-particles.

2) Data modelling

The data modelling capabilities chosen here are either common to many examples of associated data, or are included because of their importance. *Geometric* modelling is the capability to describe the geometry of an object (such as an airplane wing), or a collection of objects (such as the position of detectors). The term *entities* refers to the need to distinguish between multiple entities, which is a basic assumption in all database models (such as relational, hierarchical, etc.). There are situations where the concepts of entities are not naturally applicable, such as with summary data (e.g., a co-variance matrix).

The terms *hierarchical* and *networks* refer to the relationships between entities. A hierarchical property obviously implies a one-to-many relationship between entities of successive levels of the hierarchy, but also implies the possibility that the identifiers (keys) of higher levels propagate down to lower levels. For example, a particle identifier usually propagates down to its sub-particles level, and is concatenated with the sub-particle identifier to form a unique key. A network property indicates the existence of a many-to-many relationship between entities.

We use the term *generalization* in the sense described in [Smith & Smith 77]. Briefly, it is the capability of describing generically the properties that apply to an entire set of objects. For example, the common properties that describe all analog-to-digital converters of a certain type used in a certain experiment should be described only once. Each individual converter can have its own specific characteristics, but the generalization capability allows the common properties to be "inherited" by each individual converter.

The existence of multi-dimensional data was explained before in the context of the identifier of experiment data. Although not as common in associated data, the capability to support multi-dimensional data is nevertheless important, especially for analyzed and summary data. We refer to this property as *N-dimensional*. *Meta-data* refers to the information necessary to describe the data. However, the intent here is to emphasize the information that is beyond the usual data definition capability provided by most data management systems. An example of such additional information is the source from which an analyzed database was derived, and the person who derived it.

3) Usage

It is often necessary in the analysis process to change the definition of the database schema, such as to add new attributes (columns) or to calculate new attributes from previous attributes. The ability to support such changes dynamically is referred to here as *schema variation*. Supporting *historical* data implies the maintenance of the history of changes made to the database, and not only the latest updated version. In the implications section we discuss the different aspects of historical data needed for associated data. An important property of a database is its *stability*, i.e. infrequent updates. In physical database design there is usually a trade off between the efficiency of retrieval and the efficiency of updating. One can take advantage of stable databases to employ more efficient retrieval algorithms in exchange of slower updating.

4) Non-standard data types

The results of the analysis of scientific data are often presented as *graphs*. By *text* we mean not only the usual ability to support character strings of limited size, but also the support of unlimited text, such as article abstracts or manual information. The *time series* data type is important in scientific databases (as it is for statistical databases) because special statistical analysis techniques can be applied to time series. The ability to represent the *molecular structure* of materials is a special requirement of scientific data. It cannot be thought of as graphs or images, because it is necessary to be able to refer to the details of the structure, such as "double bonds between certain atoms". We did not include this category in Table 2 because our examples did not have such a property, but it is a well known requirement for chemical property data as can be found in many chemical property publications (e.g. The Journal of Chemical Information and Computer Sciences). There is also a need to represent *special symbols* which requires the support of a large character set. By *non-scalar* data type we mean vectors, matrices, and combinations of these. The ability to refer to such objects by name, to refer to element of the objects (such as the i,j element), and storing newly generated non-scalar objects as part of the database is an essential capability for scientific data.

Non-standard data types are discussed in [Hampel & Ries 78].

5) Database size

The size figures shown in Table 2 are intended to show the amount of associated data that is required to support or is generated from a collection (described in section 3.1, part 3) above) of experiment data. The *bytes* figures represent an approximate upper bound, and the *percentage* figures show the approximate size percentage relative to the size of the experiment data collection.

4. EXAMPLES OF SCIENTIFIC DATABASES

In this section we discuss different scientific applications with respect to their properties as defined in the previous section. For each example there are three parts: description, experiment data, and associated data. We have tried to select these examples so that they cover a diverse range of applications. They include simulations, experiments, as well as property data. Some of the applications are described in much detail because we felt that they were representative of many similar applications.

There are ten examples altogether. For the sake of organization, we grouped the experiments and simulations into three sections called: regular-sparse, regular-dense, and irregular-dense according to their identifier properties. There is no irregular-sparse group because such applications are not likely and we did not have such examples. In addition to the three groups above, there is a separate section on property data.

4.1. REGULAR - SPARSE DATA

4.1.1. Time Projection Chamber

1) Description

The Time Projection Chamber (TPC) is a device used in high energy physics experiments to record the sub-particles resulting from particle beam collisions. In a typical experiment, two particle beams are accelerated to very high speeds, and made to collide. Each such collision, called an event, may produce sub-particles that scatter in different directions at different speeds. Often the particles only graze each other and do not produce the sub-particles desired. Because some events are very rare, and because of the need to be statistically accurate, the collision experiment is repeated millions of times.

It is not important here to describe the details of the TPC device, but it is important to understand its operation in order to describe the data generated by it. The TPC is essentially a large cylinder filled with a certain gas. the collisions occur in the center of the cylinder. When particle (or sub-particles) travel through the gas they ionize the gas, leaving "tracks" where they pass. In order to distinguish between positive, negative, and neutral particles, the TPC is subjected to a magnetic field which causes the charged particles to travel in circular patterns which depend on their charge. At the two ends of the cylinder electrostatic fields are applied and cause the ionized tracks to drift to the ends. Special detectors detect the position and time of the drifting tracks, and measure the magnitude of ions reaching them. From the position of the detector, the x and y coordinates are determined. From the recorded drift time, the z coordinate can later be calculated. The data is collected through special hardware in a binary form onto tapes.

2) The experiment data

2a) Identifier

Each data point of the experiment data consists of a pulse measurement of a certain detector at a certain time. The identifier of each data point consists, therefore, of the position of the detector and the time. The reasons for considering the identifier as having regularity and sparsity are explained next.

The detectors are placed on the two circles at the ends of the TPC cylinder on concentric circles at regular intervals. Because the intervals are regular one can compute the actual x-y position of the detectors by knowing the concentric circle number and the ordinal number of the detector on the circle. The identifier points are said to be *regular*, because their position can be computed from ordinal numbers, similar to what can be done for a mesh of points.

Readings exist only for the points representing the tracks of the event. Thus, most of the detectors readings are null (in reality, below a very low threshold). Only about one percent of the potential data points have readings. Thus, the identifier is said to be *sparse*.

There are several techniques that can be used to store identifier data that is regular and sparse. they are discussed in the implications section. the most obvious technique is to throw away the null points and to store the identifier of the non-null points with the data values. This is indeed what is currently done for the TPC experiment.

2b) Access pattern

The first step required before the data can be analyzed is to reconstruct the tracks from the experiment data. The method used is to compute each potential track path using physics properties of particle, and to verify that data points exist for it. The process of verification involves a search of points along the presumed path. For each such point, the neighboring points are also needed because a pulse has a certain width (for each pulse about 4-6 neighboring data points exist).

The above process exhibits the following access pattern. The access type is *exact* match and *proximity* search, because for each pulse one looks for a particular point and a collection of points around it. The access sequence is mostly *non-local*, because each successive collection of points (representing a pulse) are not necessarily close to the previous search. Once a few point are found, the rest of the points are searched along the presumed path. In this case, the access sequence is *local*, because each successive collection of points would be close to the previous collection.

2c) Size

In a typical 6 months period, data about 4 million events are collected. An event is run about once per second, and generates an average of about 28k bytes. A (particularly interesting) large event may generate about 120k bytes. Thus, the total volume of data for this period is about 10^{11} bytes. This data is stored on about 1350 magnetic tapes. The main difficulty in dealing with such a large volume of data is the mounting and management of tapes for processing. A mass storage system would be most useful for such an application.

Since the data for every event can be analyzed independently from the other events, they can be considered a separate *unit*. The process of track reconstruction needs only a single unit at a time. However, as discussed later there are other processes that need to be run over a large number of events.

3) The associated data

3a) Configuration data

Although the configuration data does not explicitly exist as a database, it nevertheless exists in the programs analyzing the experiment data. This data corresponds to the description of the physical configuration of the detectors on the TPC device. It consists of mapping information between the identifiers of detectors as stored with each data value and their x-y coordinates. It also includes the mapping of the time measurement to the z coordinate.

3b) Instrumentation data

The instrumentation data is quite extensive and has many components. There is calibration information for each of the 16,000 channels associated with each of the detectors. This information is used to adjust the readings of the detectors. There is other information representing the distortions due to imperfections in the magnetic field, the changes in the electric fields over time, etc. All this information is necessary in order to calibrate the experiment data.

The total amount of instrumentation data is a few megabytes. It is not very large to manage, but it is complex since it contains many components. It is not obvious how to best organize such information in a database management environment.

3c) Analyzed data

The analysis process has many steps that necessitate a number of passes over the experiment data. Each step generates data files that are used in later steps. For example, one of the passes generates histograms over the experiment data. These histograms are used to determine constants for further analysis. A set of (multi-dimensional) histograms is taken over a collection of about 2000 event, and occupies about 400 kbytes. There are about 2000 such sets over the experiment data. These histograms are examples of non-standard data types that require the capability of managing an entire data set as a single item.

The final result of this analysis process is to produce summaries about tracks that belong to events. These summaries form the databases that need to be searched for interesting phenomena. Typically, the access type is a *range* search over some particle measures such as mass and momentum. The access sequence is *non-local* since there no a priori correlation between successive queries.

3d) Summary data

Further analysis over the track and event data usually results with graphs and histograms. These files need to be managed as non-standard data types.

4.1.2. Limited track reconstruction

1) Description

Limited Track Reconstruction (LTR) is another example of a physics experiment that produces data whose identifiers are regular and sparse. However, unlike the previous Time Projection Chamber example (TPC), where the main interest was in mapping precisely the tracks of sub-particles, the main interest in this example is to get detailed, high resolution measurements on the properties of particles. One can think of these two examples being at two ends of a spectrum. TPC records many point along the track, but for each point it measures very few parameters (the charge of the ionized gas) in low resolution. LTR, on the other hand, takes many measurements (10-30) at high resolution at a single point for each sub-particle. Therefore, the processing requirements of the two example are different.

LTR experiments usually involve a particle beam that collides with a stationary target. The detectors are placed in a configuration of a ball around the point of collision. Many types of detectors are used at each detection point on the ball. Different detectors are needed for the different types of particles that may be produced. The different detectors at each detection point are put into a structure called a "telescope". Each sub-particle reaches a single telescope, and all measurements for that sub-particle are taken by the detectors in the telescope. The reason for the label "Limited Track Reconstruction", is that a track is determined from information gathered at one point only by the corresponding telescope.

2) The experiment data

The telescopes are placed in a regular configuration on the ball. Data is kept only for the telescopes that sub-particles reached. Thus, the data is ***regular*** and ***sparse***. However, each data point has many measurements associated with it.

Since there are no tracks to follow there is no need for exact or proximity access types as observed in TPC. Similarly, there is no locality in the access sequence. Most of the processing can be done for each of the events in an ***arbitrary*** order. There is some level of ***range*** access type when events with certain characteristics are searched.

The size of the data is estimated at an order of magnitude smaller than TPC because the number of points measured is smaller.

3) The associated data

The configuration data is fairly simple because of the regularity of data points. However, the instrumentation data is complex since there are many variables that change over time and need to be recorded and managed. These time dependent variables include currents in some 50 magnets, pressure in the target chamber, slow drift in voltage and temperatures, and drift in the calibration of detectors and other instruments. The analyzed data consists of tracks, events, and their properties. Since there is a large number of measurements for each sub-particle and each event, the analysis pf the data involves range access type over multi-dimensional space, where each dimension is a certain measurement such as energy, charge, mass, etc. Summary data is similar to TPC.

4.2. REGULAR - DENSE DATA

4.2.1. Hydrodynamics - Difference Methods

1) Description

Hydrodynamics calculations are concerned with modelling the flow of fluids (liquids or gases). Such problems arise in combustion modelling, magneto-hydrodynamic calculations of laser and magnetic fusion, turbine modelling, air pollution simulation, nuclear weapons simulation, ICBM reentry vehicle design, weather modelling, aircraft design, etc.

Typically these problems involve the numerical solution of certain partial differential equations. This is usually done by repeatedly calculating the values of the fluid flow on a set of points defined by a spatial grid (mesh) at short time intervals. Some subset of the calculated data is saved on disk for subsequent post-processing.

The post-processing typically consists of generating various plots describing some portion of the solution. Typically the plots have lower dimensionality than the problem (i.e., 2-dimensional vs. 3-dimensional + time).

Various hydrodynamic computations may be distinguished by the types of meshes they employ. The value of the fluid flow at each mesh point is calculated from the values of the flow at the previous time step at the same and adjoining mesh points. We refer to the adjacency relations of the mesh points as the "logical" structure of the mesh. The "spatial" characterization of the mesh refers to the relationship of the mesh points to "real world" spatial coordinates. During the computation the logical structure of the mesh is exploited to access adjoining mesh points needed to perform the calculation. The classification of meshes is described below in terms of their logical and spatial structures.

Data management for hydrodynamics is extensively discussed in [Bell 83].

2) Experiment Data

2a) Identifier

Since meshes have both logical and spatial structures, the identifiers of mesh points are expressed in both terms.

The simplest hydrodynamic mesh structure is a spatially ***uniform*** mesh. It is logically ***regular***.

If we continuously deform the mesh, e.g., by wrapping it around an airfoil, then the resulting mesh is spatially ***non-uniform***, but still logically ***regular***, i.e., the adjacency relationships of the mesh points have been preserved. Logical regularity facilitates nearest neighbor searches, without requiring the examination of spatial coordinates.

If we apply finer grained mesh patches to the original mesh to enhance the computational accuracy in certain regions, then the logical structure of the resulting mesh is ***locally regular***. As before, the spatial structure may be either uniform or non-uniform.

If we permit mesh points to move with a turbulently flowing fluid, then the original adjacency relationships will not be preserved. Hence the resulting mesh is logically ***irregular***.

Hydrodynamics data is ***dense***, i.e., there are non-null data values associated with all defined mesh points.

Meshes may either remain fixed over time or they may move (e.g. with the flow). Spatial movement increases data storage requirements and complicates retrieval based on spatial coordinates (during postprocessing). *Time variation* of the mesh always affects the spatial structure of the mesh. It may or may not affect the logical structure of mesh. If the logical structure is not affected (and was initially regular) the logical regularity of the mesh can be used to efficiently access nearest neighbors.

2b) Access Pattern

During the computation phase the access pattern consists of a sequence of clusters of spatially ***proximate*** references (all the nearest neighbors of a point). Regularity in the mesh permits nearest neighbors to be accessed via simple array index calculations. If the logical structure of the mesh is irregular but time invariant, then the nearest neighbors of each point can be found (by means of nearest neighbor searches on the spatial coordinates) and recorded by means of pointers. Hence subsequent accesses can be performed by following a pointer (i.e., constant time). If the logical structure is both irregular and time varying then the nearest neighbors must be determined dynamically, by means of repeated nearest neighbor searches over the spatial coordinates.

In some types of computations the sequence with which mesh points are updated is ***arbitrary***, in others there is spatial ***locality*** in the reference sequence.

There are several different types of postprocessing accesses. Usually these accesses are framed in terms of the spatial and temporal coordinates. Often they consist of requests for a spatial or temporal cross-section of the data. If the data points do not exist on the cross-section, interpolation from adjacent grid points is necessary. This generates special types of proximity searches, e.g. nearest neighbors of a plane (rather than a point).

One way of portraying the fluid flow is to simulate the effect of injecting a dye into the fluid at a particular point. The dye would be carried along with the fluid flow. Since the path of the dye particles is continuous in space and time, the simulation generates ***proximity searches*** on the spatial and temporal coordinates.

All of these data accesses characteristically exhibit ***locality*** of reference in the coordinates of the problem (space and time).

There is some interest in accesses which would identify regions of turbulence or shock waves. Such queries ask about the values of the flow field (curl for turbulence, gradient for shock wave.). This is an unusual example of a search on the data values (instead of the identifiers).

2c) Size

Typically a single hydrocode run might generate one billion words (8 bytes each) of data (i.e., a 1 million word mesh run for 1 thousand time steps). Hence the data size per time step *(unit)* is about 10^7 bytes. Often, if the data is only being stored for graphical post-processing it will be stored in reduced precision (i.e., 2-3 bytes per word instead of

8). Thus the data size is generally 10^9 to 10^{10} bytes per problem *(collection)*. Often only a fraction (5 to 20 percent, i.e., 10^8 to 10^9 bytes) of the time steps will be saved, due to storage constraints. One problem might consume a few hours of CPU time on a powerful machine such as a Cray or Cyber 205. A large site might generate as many as 20 such runs per day, keeping the complete results for several days. Hence the ***total*** size of the database (10^{11} to 10^{12} bytes), and sampled results for several months (100 days), i.e., 10^{10} to 10^{11} bytes.

3) Associated Data

3a) Configuration Data

The geometry, boundary conditions, initial conditions, and initial mesh geometry constitute the configuration data. These data may change from one experiment (simulation run) to another.

3b) Instrumentation Data

There is no instrumentation data.

3c) Analyzed data

The analyzed data consist of the graphics post-processing output. In a two-dimensional computation this would consist of a sequence of contour maps of scalar variables (e.g., pressure), vector plots of velocity vector fields, and streamline plots (e.g., generated from dye injection studies). Sometimes a sequence of plots will be combined to make a movie.

3d) Summary data

Conservation of mass, energy, and momentum are checked by keeping track of the total mass, energy, and momentum in the system at each time step. While the total mass, energy, and momentum in the calculation is conserved, users are often interested in the transfer of energy (or mass, momentum) between portions of the system. Often users will conduct parametric studies of various phenomena, studying how certain figures of merit of some system varies as function of initial or boundary conditions. Examples of summary statistics used in such parametric studies include:

(1) drag coefficients of airplane wings,

(2) location of shock waves on airplane wings,

(3) yields of nuclear explosions,

(4) growth coefficients (e.g. rate of growth of plasma instabilities),

(5) rate of movement of fluid interfaces,

(6) amount of energy transferred into a laser fusion target.

4.2.2. Two Dimensional NMR Spectroscopy

1) Description

Nuclear Magnetic Resonance (NMR) spectroscopy is a technique for investigating chemical structures. A sample is placed in a strong magnetic field so that the magnetic dipoles of the atomic nuclei of the sample are aligned. A radio-frequency (RF) signal is then applied to the sample, causing the nuclear magnetic dipoles to rotate (at their precession (or resonant) frequencies). The RF signal is removed and the RF generated by the rotating magnetic dipoles in the sample is measured (periodically sampled in the time domain). The spectrum is then obtained by applying a Fast Fourier Transform (FFT) to the measured RF time series. The RF spectrum generated by the sample has peaks at the natural resonant frequencies of of the sample. These resonant frequencies are affected by the nuclear and chemical structure of the sample, and can thus be used to investigate these structures.

In NMR spectroscopy it is customary to wait for a period of time (called the *evolution time*) after the RF excitation has been applied to the sample (and turned off) for the sample to "evolve". Then the RF generated by the sample is repeatedly measured at fixed time intervals, called the *detection times*. In two-dimensional (2D) NMR spectroscopy a set of NMR spectra are repeatedly taken at fixed increments of the evolution time. A two-dimensional spectra is formed by first applying an FFT to each RF time series (i.e., rows of the data matrix) producing a second matrix (each of whose rows now contains the NMR spectra for a particular evolution time). Then a second FFT is applied to the resulting columns, each of which contains the same frequency component for all evolution times. The resulting two-dimensional spectrum provides information on the "evolution" of the sample, i.e., how energy is transferred between resonances.

2) Experiment Data

2a) Identifier

The experiment data consists of the a set of time series of RF measurements taken at fixed time increments of detection time. A separate RF time series is taken at each increment of evolution time. Hence the identifiers of the experiment data consist of the detection times and the evolution times.

Because of fixed sampling intervals in detection time and evolution time the experiment data is entirely regular, comprising a completely full (dense) rectangular array of data.

2b) Access Pattern

Before any FFTs are performed the input is smoothed by computing moving averages of the each detection time series of RF measurements (row). This entails *linear* access to the data, row-wise. Then an FFT is done on each row, then on each column of the result. Each FFT generates a *local* access sequence on a different dimension, i.e., first local to a row, then local to a column. Hence, if the array will not fit into memory, the computations consists of performing row-wise FFTs, then transposing the array, and performing the second set of FFTs. The transposition generates *non-local* access sequences. At present the data analysis is disk bound, i.e., limited by the disk accesses required to transpose the array.

2c) Size

Currently a typical experiment consists of 256 by 256 data points where each data point is comprised of a complex number (i.e., two scalar values) measured by 10 bit analog-to-digital converters. Currently each scalar value is stored in 16 bits (2 bytes). Thus the experiment data occupies 2^{18} bytes (.25 MB) per experiment (*collection*). An experimenter might conduct an average of one experiment per day. Data acquisition is presently constrained by analysis capabilities (see below), not experiment equipment. It is expected that future experiments will go to 2K by 2K data points, i.e., 2^{24} bytes (16MB). It is commonplace to keep hundreds (perhaps a thousand) of such spectra, so that the *total* size of the spectral data will range from 10^8 to 10^{10} bytes.

3) Associated Data

3a) Configuration Data

The configuration data consists of a specification of the sample (its chemical composition, temperature, etc.), the excitation RF pulse sequence, and the repetition rate for the spectral measurements.

3b) Instrumentation Data

The instrumentation data consists of calibration data for various analog-to-digital converters, RF probes, temperature probes, etc.

3c) Analyzed data

The analyzed data consist of the two-dimensional spectra. This data is comparable in size to the input data. It is usually reported as plots, either contour maps or tree-dimensional drawings. Such graphical data constitute ***non-standard*** data types. However, at present such graphics are not stored digitally. Instead they are redrawn when needed from the stored spectra.

3d) Summary data

Often curves will be fitted to spectral peaks, and the location, height, and width (scale parameter) recorded as summary statistics. A sequence of experiments with varying chemical conditions or excitation sequences may be reported (via tables or graphs) in terms of the effect on the fitted parameters of various spectral peaks.

4.3. IRREGULAR - DENSE DATA

4.3.1. Heavy Ion Spectrometer System

1) Description

The Heavy Ion Spectrometer System (HISS) is a device used in nuclear physics to study the nucleus break up process in catastrophic collision involving heavy ions. A heavy ion particle beam is accelerated to a high speed, focused, and aimed at a stationary target. Each collision between heavy ions, called an event, produces fragments that are subject to a strong magnetic field and scattered in different spirals depending on their charge, mass, and velocity. Each spiral is detected by two detectors, called drift chambers. Each drift chamber is filled with gas and has a fine mesh of electronic sensors which can detect the ionized gas caused by the passage of the fragments. The sensors can detect the x and y positions of the fragments, as well as their charge. A device, called the "time of flight wall", is put next to the drift chambers to detect the fragments after they leave the drift chambers. The speed of each fragment is determined by the time-of-flight-wall. The data, which include the x, y from the two chambers, speed, charge, and other information is recorded and stored on tapes. A limited track of each fragment can be reconstructed from three points: the x, y of the two chambers and the collision target position.

2) The experiment data

2a) Identifier

The way data is generated in HISS is similar to TPC in that each data point consists of a measurement of charge by a certain sensor. However, we classify the HISS data as spatially *irregular* due to the fact that the drift chambers can be placed in arbitrary locations from experiment to experiment or even during experiments. Since each chamber generates a single x, y position for each fragment, the data is classified *dense*.

2b) Access pattern

The track reconstruction process in HISS is much simpler than TPC since only two points are detected for each fragment. The access type if *exact* match because the points which are used for track reconstruction are discrete in space. Since the reconstruction can start at any one of the three points, the access sequence is classified as *non-local* access.

2c) Size

Each experiment typically runs for a few months for a total of 100 to 150 hours of experiment time. During experiments, a beam is shot at the target every 5 seconds for the duration of one second. During that 1 second period, 100K words of data are collected. There are on the average 100 collisions in a second. Each collision is an independent event, therefore the *unit* size is 10^3 words. From the above figure, a *collection* size (over the few months) is in the order of 10^{10-11} bytes. About 10 such collections are kept available for a *total* of 10^{12} bytes.

3) The associated data

3a) Configuration data

The configuration data of HISS consists of the physical layout of the instruments. This includes the physical position and kind of target, the physical positions of the drift chambers, the kind of gas, its pressure and temperature, the position of the magnet, the time of flight walls, etc. The configuration data is typically small and rather stable. The data is currently embedded directly into the logic of the analysis programs. One important requirement is to isolate this data from the analysis programs in order to remove the redundancy of having the same data appear in each analysis program. Also, this will provide an opportunity to model the data in a more uniform manner.

3b) Instrumentation data

A large part of the instrumentation data for HISS consists of calibration data for the 1200 sensors in the drift chambers. There are also hundreds of power sources, magnetic field and voltage variations, etc., which have to be maintained to calibrate the experiment data. Textual information is also needed to document unusual happenings such as failure of certain instruments. The instrumentation data in HISS is fairly dynamic. As a result, the association of this data with the experiment data is very difficult to handle if the instrumentation data is part of the analysis programs.

3c) Analyzed data

Different groups of physicists produce different sets of event data from the raw track data. Each set is called a "physics file". A physics file, on the average, contains about 10^9 bytes, usually stored on tapes. The access type on these physics files is typically *range* search on the number of tracks, charge of fragments, etc.

3d) Summary data

Histograms and scatterplots are generated from the analyzed data in order to look for statistically interesting event types. These summary data are subject to further manipulation to confirm, predict, or reject nucleus theories.

4.3.2. Passive Solar

1) Description

The passive solar experiment involves a large simulation code of heat transfer to study the sun energy performance in actual buildings such as residences and industry. The input to the code consists of two files, a weather file which describes the initial weather conditions, and a physical building file which specifies for each building, the zones (which can be thought of as rooms), the surfaces of each zone (such as a wall), and parts of surfaces (such as windows and doors). A typical building consists of 10's of zones, 30 to 40 surfaces per zone for residential buildings, 100's of surfaces per zone for commercial building. The code is typically run on thousands of time steps where at each time step, all data associated with each surface in each zone is calculated based on the data of its neighbors.

2) The experiment data

2a) Identifier

There is no regular mesh of data points in this experiment, since the placement of data points depends on the building configuration. The identifiers of the data points have to be enumerated by their positions in the building, hence they are classified as *irregular*. Since every data point always has values associated with it for all time steps, the data is classified as *dense*.

2b) Access pattern

The computation of a point in a particular time step requires the access of all its nearest neighbors. Hence the access type is *proximity* search. The access sequence of data points, however, is *arbitrary*, since the order of access of data points is not predetermined.

2c) Size

A typical run of the simulation generates a *collection* of 10 million numbers, and consists of 9000 time steps. The *unit* size is in the order of 10^{4-5} bytes. Such simulations are run up to several thousand times a year. So the *total* data size is in the order of 10^{10-11} bytes per year.

3) The associated data

3a) Configuration Data

The configuration data consists of the weather file (about 10^5 bytes), representing the boundary conditions of the simulation, and building description file (about 10^4 bytes).

3b) Analyzed data

Subsets of the simulation results are stored primarily to avoid having to rerun the simulation up to a certain time step. But sometimes it may be less costly to rerun the simulation than to save the intermediate data generated because of the number of time steps in each simulation run.

3c) Summary data

Summary data is generated from the raw data. Several levels of summarization exist. They range from a low level of summarization where the average temperature is determined for each time step, each building, and each zone, to the high level of summarization where the total energy needed to heat or cool a building is computed.

4.3.3. Vortex modelling of turbulent flow

1) Description

Random vortex modelling is an unusual method in that it does not use a predetermined geometry such as a mesh. This technique was developed for modelling turbulence flow because this phenomena is so complex that regular techniques which solve the appropriate set of differential equations require such mass of data and computation that overwhelms any computer capability available currently. Such a modelling method is very important because nearly every physical phenomena, e.g cloud motion, ocean waves, engine combustion, water flow, etc., involves turbulence.

It is not necessary here to understand the method; it suffices to know that the method models a collection of vortices in space. Each vortex is modeled as a point in space with values that represent the vorticity of that point along with other measurements such as temperature.

2) The experiment data

The experiment data consists of the vortex points. These points are placed in space in a pattern that is *irregular*. In areas where the turbulence is high (e.g. near walls, or where fluid is injected) there is a large number of vortices in order to represent the turbulence more accurately. In low turbulence areas only a small number of vortices is required. As the simulation progresses the vortices move around. Thus, the vortex points are *time varying*.

The computation of points for the next time step requires looking for neighboring vortices. Thus the access type is *proximity*. At times a certain region of the space is needed for further analysis, and a *range* search is performed. The analysis of flow involves techniques such as simulating "dye injection" (described previously in the Hydrodynamics example), where the imaginary movement of the dye is followed. Such techniques imply a *local* access sequence because each successive step needs points that are close the the points of the previous step.

The size of the database is limited by practical considerations of cost and processing time. It is always possible to refine the simulation by introducing more points. The size of an entire simulation (*collection*) is in the order of 10^8 bytes. there are 100-1000 time steps, but because of space limitations only part of the steps are stored. The size of a time step (*unit*) is about 10^6 bytes. The *total* amount data kept is in the order of 10^{10}.

3) The associated data

The *configuration* data can be quite complex depending on the shape of the boundaries assumed. Since it is a simulation there is no instrumentation data. The *Analyzed* data consists of results representing the flow movement. The results are best displayed in a graphical form, and require the ability to store such images. Even a greater need is the display of the movement of flow in real time. This cannot currently be done because of the large amount of data that has to be fed to the display and because of cost limitations.

4.3.4. Laser Isotope Separation

1) Description

The purpose of the Laser Isotope Separation experiment is to develop a technique for recovering the reusable isotopes from nuclear waste materials. The nuclear waste material is ionized and vapors come off. A laser beam is directed into the vapor and causes the reusable isotopes to separate from the non-usable isotopes. There are sensors that measure the properties of the isotopes after they were separated. Each sensor is connected to an analog-to-digital converter. Each converter can handle several sensors. Measurements are taken every second, and a typical experiment runs for 12 hours.

2) The experiment data

The data collected from the sensors make up the experiment data. The sensors can be placed in any position desired by the experimentor. Therefore, the identifier is considered *irregular*. The data from all the sensors is recorded continuously. Thus, the data is *dense*.

A typical analysis of the data consists of selecting a subset, looking for trends, and interpolating results. The selection of subsets implies a *range* access type. In order to support the interpolation a nearest neighbor search is performed. This suggests a *proximity* access type and a *local* access sequence.

There is no size per *unit* in this example because the experiment is not made up of independent units. The size of a single experiment, running over a 12 hour period, is about 10^{7-8}. There are scores of experiments in total, so the *total* size is in the order of 10^{10} bytes.

3) The associated data

The configuration data consists of the positions of the sensors and the information of the converters that they connect to. Both can change within an experiment. The sensors can be moved around to achieve better readings, and they may be moved from one converter to another if a malfunction occurs. This requires that the history of changes is recorded carefully, and available during the analysis process.

The instrumentation data is fairly extensive. It consists of the information about each sensor and each converter (e.g. bias and amplification). This information is continuously monitored and recorded in order to insure the proper adjustments of the measured data. The experiment data is subjected to repeated analysis and summary.

4.4. PROPERTY DATA

4.4.1. Particle Data

1) Description

The Particle Data Group (PDG) is a member of the international High Energy community whose purpose is to compile data and information in a highly evaluated and summarized form. The best known of these compilations is the *Review of Particle Properties*. Other compilations include the current experiments in elementary particle physics, and a large bibliographic database containing data in elementary particle physics. In addition to the common database management requirements such as retrieval, data definition, and data manipulation, there are needs for efficient literature-searching capability, sophisticated text processing with mathematical symbols and fonts that are maintained by the database system, and very high quality phototypesetting for their publications.

2) Access patterns

The access to the database is a mixture of *exact* and *partial* match. An example of the former is to look for all the data associated with a particular particle; an example of the latter is to retrieve the citations in the database containing some relevant keywords. Another access type is hierarchical browsing where the database is traversed to look for desired items. The access sequence is typically non-local. Publication of particle property data involves primarily linear access sequences.

3) Data modelling

Particle properties data is primarily hierarchical. An example is that a particle can have several deflecting angles called branching ratios, and each branching ratio can have multiple decay modes. There are also many-to-many network relationships among data items. An example is the relationship between publications and particles, since one publication can include data on many particle properties, and data on one particle property can appear in many publications. Another example is the relationship between decay modes and branching ratios which represents a many-to-many relationship.

4) Non-standard data types

The dominant data type is the variable-length text for the bibliographical data. Other data types include special symbols for the mathematical notations and graphs for storing data such as scatterplots.

5) Usage

The usage of the particle property data includes interactive user access, literature searching, analysis program access, update of particle data, appending of new data, and data transfer to text processing systems.

6) Size

Particle data is relatively small. The bibliographic database is about 75 MB. The particle property data is only about 4 MB.

4.4.2. Nuclear Structure Data

1) Description

These activities are concerned with recording, evaluating, and tabulating data about the structure of atomic nuclei, the allowable states of the nuclei, and the reactions by which nuclei decay from one state to another (possibly different nuclei).

The database consists of three types of data:

(1) raw nuclear data

(2) evaluated nuclear data

(3) bibliographic citations.

Much of the activity of the database maintenance and publication consists of evaluating the various experiment data to construct a "best estimate" of various data values. The evaluation process is partly subjective, and partly statistical.

The compilation of nuclear structure data is used for research in the systematics of nuclear properties, to study particular reactions, and for nuclear engineering (reactor and waste disposal calculations).

Until recently the primary form of dissemination consisted of printed publications, which remain important. Recently interactive access to the database has been provided to general users.

In addition to the need for data management functions, this project requires efficient literature-searching capability, sophisticated text processing with mathematical symbols and fonts that are maintained by the database system, and very high quality phototypesetting for their publications.

2) Access pattern

There are three types of queries:

(1) finding the value of some property of a particular isotope,

(2) spectral matching - given an observed set of decay products, identify the isotope,

(3) and bibliographic searches.

Property retrievals are usually either *exact* or *partial match* on the isotope and property name. Identification searches, in which one attempts to identify nuclear isotopes from decay products, lead to *proximity* searches (nearest neighbors) or *range* queries (fixed radius neighbor searches). Bibliographic searches are typically partial match on authors, keywords, or exact match on citation (accession numbers listed with experimental data).

Browsing access tends to generate *non-local* access sequences. *Linear* access sequences are generated by some evaluation and tabulation (for publication) activities.

3) Data modeling structures.

Nuclear data is primarily hierarchical. However, there are also many-to-many network relationships among data items similar to those found in the particle data discussion above. Such relationships exist between publications and nuclear properties, and between decay modes and branching ratios.

4) Non-standard data types

Non-standard data types include *textual* annotations, bibliographic citations, and *graphs*. The textual data includes *special symbols*, e.g., Greek and mathematical notation.

5) Size

The database size is approximately 40 MB.

5. IMPLICATIONS

The implications derived in this section can be best followed by referring to Table 1 for experiment data and Table 2 for associated data. The organization of these tables was designed after the information on the different applications was collected in order to clarify its presentation. However, we believe that these table structures can be used to classify additional applications. Once the appropriate entries are filled for an application, one could quickly draw conclusions on its requirements and the possible data management techniques to support it, along the lines discussed below.

5.1. EXPERIMENT DATA

We discuss the entries of table 1 by referring to its rows because the rows represent observations about each property. The sections below are organized according to the row groups in the tables. The first section labeled "experiment/simulation" is merely to identify whether each example is an experiment or simulation for later reference and for clarity.

1) Identifier

Identifiers whose dimensions have a *regular* structure are quite common. The main reason is that simpler algorithms can be developed for them, and that the data can be organized in an orderly fashion. The simplest case exists when the configuration of the experiment or simulation forms a multi-dimensional mesh. In such a case there is no need to store the identifiers of the data because the position of each data point can be calculated using the "array linearization" technique mentioned in section 2. Indeed, the array capabilities of programming languages have been used extensively by scientific application. This suggests that an array linearization access method would be most desirable in a scientific data management system. The advantages of such an access method is that it requires no storage for the identifiers and provides a very efficient random access (a simple computation) to the data points.

The situation is more complex when the configuration is not simple, such as representing an airplane wing or the shape of a combustion chamber. In such cases a mesh that covers the entire configuration can be imposed, and all the points outside the configuration boundaries are considered null. This approach introduces a certain level of sparsity in the data points. We will discuss sparsity below.

Other forms of regularity may exist. One is the regular placement of points along some geometric shape, such as concentric circles. Another occurs when two kinds of regular structures co-exist, such as having a finer mesh in certain regions of the configuration. In such cases the mapping algorithm of logical points into a linear sequence is more complex than array linearization, but they still provide storage savings and more importantly a fast random access to the data points.

As can be seen from Table 1 there are several examples of *irregular* identifiers. Irregular identifiers do not necessarily have to be explicitly stored with each corresponding data value. Rather, the irregular dimensions (such as the different walls of a building) can be stored once and enumerated. Thereafter, the identifiers can be calculated using array linearization over the enumerations. Irregular dimensions are most common in statistical databases (such as state, race, sex, and cause of death for mortality data), where the enumeration of each dimension and array linearization over them is a most effective method.

Data *sparsity* means that only a fraction of the points in the full cross product of the dimensions have actual values associated with them. There are basically two options: either to store the identifiers of the valid data points, or to compress out the non-valid (null) data points. Compression methods, such as run length encoding (which introduce a count into the data stream in place of each sequence of null points) can be quite effective, especially when the null points are clustered to form long sequences. However, such compression methods (as is the case with storing the identifiers) require sequential scanning of the data in order to select a particular point randomly. Indexing methods require too much space for large databases and may be prohibitive.

In [Eggers & Shoshani 80] a compression technique, called header compression, which provides fast (logarithmic) access was proposed for statistical databases. It basically organizes the run length counts into a separate header, in such a way that the header can be searched in logarithmic time with respect to the number of counts. This technique can be applicable for sparse scientific data as well, since it used effectively with multi-dimensional data.

When the identifiers are *time varying* there is no choice but to store them, since they change from one time step to the next. In the case that the data is also regular, there is an additional requirement that the original relationship between the points is maintained. To see this point, one can imagine a mesh of points connected by rubber strings. The entire structure can then be stretched and compressed in successive time steps. The maintenance of these relationships can be achieved with techniques applicable to regular data. When data is irregular and time varying, the relationship between the data points changes from one time step to the next, and has to be deduced from the stored identifiers.

Time varying applications are not as common as other applications, but they represent an important class of modelling techniques.

2) Access pattern

From Table 1 it can be seen that the access types of *exact* match and *proximity* search are important. Exact match implies, in general, the need to access specific data points randomly. To accommodate such a requirement some kind of an indexing or a hashing technique is required. Fortunately, one can take advantage of the multi-dimensionality of the data. The mapping of multi-dimensional space to linear space discussed above (e.g. array linearization) provides a key-to-address mapping that is equivalent to hashing. In addition, some multi-dimensional to linear mappings provide advantages for proximity search as discussed below.

To support proximity search it is necessary to preserve logical locality in physical storage. That is when points are logically close to each other in the multi-dimensional space, it is desirable that they are physically close in physical space. This suggests the organization of physical storage into cells along the dimensions of the identifier. The data points within a cell will satisfy the proximity requirement. For elements on the borders of cells it is necessary to access adjacent cells, and therefore the placement of cells in physical storage is also important. The mapping of multi-dimensional space to linear space mentioned above works well with such a cellular organization because it does not disturb the logical proximity of the data points. An arbitrary hash mapping would place data points into cells (buckets) which would not necessarily preserve logical proximity.

The *range* access type does not seem to be as important. Nevertheless, the cell organization should benefit range access because the data is organized according to the dimensions for which ranges are specified.

Referring again to Table 1 it seems that *local* access sequence is also important. The cell organization is also helpful here because local points are likely to be in the same cell. The question of how to organize the cells arises here again. If the paths of local access sequences are known or predictable, then the cells should be organized along these paths. The benefits of such ideas need to be investigated.

Non-local access sequence is not as prevalent as local access sequence. However, it can be supported well with cell organization. The reason is that it complements the requirements of exact match, since it implies the need for a random access of the data points. *Linear* access sequence conflicts with the idea of a cell organization, because the linear sequencing of the data is broken. However, it does not seem to be an important requirement. If data was organized "linearly" to accommodate this requirement, proximity search and local access sequence will be performed less efficiently.

An *arbitrary* access sequence is quite common. It usually implies that the entire data set needs to be processed, and that the order of points is irrelevant. This suggests that parallel processing can be performed over the data. This only complements the cell organization approach, since the cells could be placed on parallel devices for parallel processing.

In summary, it seems that the cell approach is most desirable since it accommodates the most important requirements. The organization of cells should be along the dimensions of the identifier, since they preserve logical locality. The approach of mapping the multi-dimensional space into linear space only complements this cell organization. There are several papers that discuss the organization of data into cells [e.g.

Nievergelt et al 84]. They mostly emphasize the compression of sparse data into cells. While such an approach is important for sparse data, it is unclear how well such methods will work for the access requirement mentioned here, such as proximity search and local access sequence.

3) Size

The most important observation that can be made from the size figures in Table 1, is that although scientific databases are large, they can often be partitioned into small independent *units*. The units are small enough that much of the processing can be done in main memory. In general, experimental units can be processed in parallel, since they are independent of each other. Simulation units (time steps), on the other hand, usually follow each other in sequence. Note that simulation units are typically larger than experimental units.

Unit processing is only one part of the analysis process. Other types of processing need to search and access entire *collections*. As can be seen from the collection figures in Table 1, some collections are so large that they cannot be practically stored on magnetic disks. In such cases, the data is currently stored on tapes and the mounting of those tapes becomes a major problem. Current solutions are to process the data sequentially once, to collect interesting subsets, or to break the data into redundant smaller sections. It is obvious that larger secondary storage devices (such as optical disks) could be helpful.

4) Associated data

Associated data is discussed in the next section. The different types of associated data were included in Table 1 in order to point out their importance and prevalence. Nearly all applications have all types of associated data. The obvious exception is that simulations do not have instrumentation data.

5.2. ASSOCIATED DATA

Table 2 summarizes our observations on the different types of associated data. We could discuss these observations by row for each class of properties or by column for each type of associated data. A close observation of the table reveals that there are many similarities between the configuration and instrumentation columns, and between analyzed data and summary data columns. This is not very surprising since these two groups represent support data and generated data and should have similar characteristics. In fact, early on we did not make this finer distinction, but later we found that it helped sorting out the different aspects of scientific data.

Accordingly, we will discuss properties in Table 2 in three parts: the support data (configuration and instrumentation data), the generated data (analyzed and summary data), and property data.

1) Support data

The access type for support data is mostly exact match. A typical access involves finding a particular configuration point and the particular instrument associated with it. Proximity search is sometimes needed. For example, if a certain instrument failed, the configuration data may be consulted to find the instruments in neighboring locations. The access sequence is mostly non-local, which indicates that successive queries are unrelated. Thus, the access requirement for support data is mainly random access.

The data modelling requirements are fairly conventional, i.e. modelling of entities that have hierarchical or network relationships. The relationships between the different instruments and detectors are part of the configuration data. Generalization is an important modelling tool for instruments, as generic information can be represented once and inherited by each particular instrument in that class.

An important exception to the conventional modelling requirements mentioned above is geometric modelling of configuration data. In many examples the geometry is quite regular and could probably be modelled with simple types (points, lines, circles, etc.). However, geometric shapes may be complex enough to require special modelling techniques similar to those required in engineering databases [Lorie 82].

Another major requirement is for the support of historical data. Instrumentation data change continuously over time, and the entire history of changes has to be recorded. In addition, logs of the operation, such as when an instrument failed, who was in charge at the time, etc. need also be recorded. The time element can be thought of as another dimension orthogonal to the structure of the database. It requires special storage techniques and special operators such as "after" and "during". Several recent works have dealt with this topic [e.g. Anderson 81, Bolour et al 82]. The history of configuration data changes also needs to be recorded, but not as often as instrumentation data because they usually occur only between experiments.

Support data may have some text that describe procedural instructions or configuration descriptions. Instrument data are usually polled at regular time intervals, and could benefit from a time series data type. The size of the data is relatively small, and constitutes only about 1% of the experiment data.

In conclusion, we believe that support data can be managed for the most part with conventional data management systems. The databases are relatively small. The requirement for random access can be accommodated with conventional indexing methods such as hashing. The two most important exceptions that require special attention are historical data support and geometric modelling.

2) Generated data

The access pattern of generated data is similar to statistical databases. That is, it is mostly range and partial match queries. As with statistical databases, the generated data is repeatedly analyzed in order to discover patterns, statistical behavior, or a rare event. Many subsets are generated and need to be kept track of. The access sequence is mostly non-local, although locality exists when analysts refine their queries. From time to time an entire set of analyzed data is processed to generate summaries. This is indicated as an arbitrary access sequence in Table 2.

The most prominent data modelling property is that generated data is multi-dimensional. Unlike experiment data where the dimensions are mostly spatial coordinates, the dimensions of generated data are the properties of the data (e.g. charge, temperature, mass). Thus, the number of dimensions can be in the order of ten, which presents a special challenge for its efficient support. In some instances it is useful to view analyzed data as entities and hierarchical relationships (for example, events and their corresponding sub-particles).

Another important modelling requirement is for meta-data. The requirements of meta-data management include data definition facilities not only for field descriptors (such as type, size, and acronym), but also the description of the origin of the data, how it was collected, when it was generated or modified, and the identity of the person responsible for its collection. It should be able to describe complex data types such as times series, matrices, and multi-dimensional categorical data.

It is necessary to organize and manage meta-data, just as is the case with data. One should be able to retrieve and search meta-data, index keywords, and browse through the meta-data structures.

Meta-data is also necessary for keeping track of the different subsets produced, dates of their creation, methods used, etc. The management of subsets also requires that their historical aspects are maintained. It is necessary to record and maintain the ancestors of each subset produced. The analysis process, similar to statistical analysis, can be modelled as a tree structure. The analyst can generate subsets, observe their patterns, and choose to go back to a previous set and follow another path of analysis. The above requirements are similar to many aspects of the meta-data management for statistical databases [McCarthy 82].

It is often useful in the analysis process to add new fields to the database or to compute new fields from other fields. This is referred to in Table 2 as schema variation. However, except for such additions during the analysis process, the generated databases are quite stable. The support of non-standard data types is most important. Generated data can be expressed as graphs, vectors, matrices, and time series. Finally, the size of the data is substantial, and although it is limited to fit on disks, it is large enough to benefit from data management techniques that minimize disk storage and access time. The total amount of generated data may be of the same order of magnitude as the experiment data it was derived from, because a large number of subsets are usually produced.

In conclusion, generated data have many properties in common with statistical databases. We believe that special techniques for the management of multi-dimensional data developed for statistical databases could be applied to support analyzed data.

3) Property data

Since property data is a summary over many experiments and general knowledge of a subject area, it has properties more akin to bibliographic databases. However, in addition to managing bibliographic data on books, articles, authors, etc. they contain summaries of information extracted from the articles. Many property databases are presently only available in periodic publications.

All access types are needed. An exact match may be required to locate a specific entry. A range query (which may be partial) could be used to find a desirable subset of the data. A proximity search will locate entries with properties as close as possible to the specified parameters. The access sequence is usually non-local. Linear access sequence is needed for generating the periodic reports, or some other requested report. In the examples that we observed it is not possible to issue ad-hoc queries or to browse the database for information since the databases are not available on-line. An on-line version should provide such facilities.

The databases are organized logically as entities and mostly hierarchical relationships between them (for example, particles with a certain mass at the top level of the hierarchy, and their derivatives at lower levels). There are some network relationships (permitting a many-to-many association between the entities) such as the relationship between papers and particles. A single paper may describe many particles and a particle may be described in many papers.

Property databases contain graphs and text which complicate their management a great deal. They also have to have a representation for special symbols that are unique to the scientific field. The total size of the example databases that we observed is not very large and can fit on disk storage. However, property databases can be quite large, as is the case with chemical property databases.

In conclusion, the main difficulties that property databases have stem from the diversity of data. This is probably the reason that only special purpose systems have been developed for the different disciplines. Conventional data management systems cannot support the combination of numeric, character, text, and graphs data.

6. SUMMARY

In this paper we examined the data management requirements and typical usage of several scientific applications. The data used and generated by these applications was classified into types, and a list of properties for describing their characteristics was developed. Different applications were then described in terms of this list of properties. A summary of properties of the different types of scientific data provided the basis for inferring desirable database management techniques for scientific databases.

Some of the more important conclusions are:

(1) Multi-dimensional data are prevalent in scientific databases. Methods for efficiently managing, accessing, and compressing multi-dimensional data are desirable.

(2) Scientific databases are frequently accessed via proximity searches and successive queries often exhibit locality of reference. Techniques of partitioning the data into cells (or grids) along the coordinates of its dimensions seem to be the most promising for efficiently supporting these needs.

(3) Although scientific databases are usually very large, they can be often partitioned into small independent units during early data reduction. This implies that parallel processing can be applied.

(4) Scientific databases include a variety of support data that describe the instruments and the configuration of experiments. Often this data is not explicitly organized but rather made part of application programs, a practice that tends to cause many difficulties. The requirements of such support data can be handled for the most part with conventional database techniques, but need to be integrated with the data that result from experiments. Some configuration data need special capabilities found in engineerinng database systems.

(5) The analysis of scientific data generates many summary data sets which need to be managed. Special techniques for handling analyzed data and summary data are required in order to manage their metadata, to keep track of numerous data sets, and to handle non-scalar data types (such as vectors and matrices).

(6) Historical aspects of scientific databases are important. They range from time series of the measured data, to logs of instruments variation over time, to the historical sequence of generating different summaries of the data.

(7) There are many aspects of scientific databases that are similar to statistical databases; in particular, supporting the multi-dimensional aspects of the data and the handling of summary data.

Acknowlegment

We would like to thank the many scientists who contributed their time and effort to explain their scientific applications, current processing techniques, open problems, and future needs. The list includes: Brandt Anderson, Marsha Berger, Mark Bronson, Eddie Browne, Bill Carroll, Phil Colella, Janis Dairiki, Barbara Garnjost, Van Jacobson, Rowland Johnson, Stu Loken, Gerry Lynch, Michael Munowitz, Creve Maples, Chuck McParland, Alex Pines, Al Rittenberg, Jeff Saltzman, Jamie Sethian, Joe Sventek, Tom Tripp, Tom Webster.

We also like to thank our collegues Paul Chan, John McCarthy, and Doron Rotem, for many fruitful discussions. Special thanks are due to Meri Jones for her text processing help.

This work was supported by the Applied Mathematics Sciences Research subprogram of the Office of Energy Research, U.S. Department of Energy, under contract number DE-AC03-76SF00098.

References

[Anderson 81]
Anderson, T.L., *The Database Semantics of Time*, Ph.D Dissertation, Dept. of Computer Science, Univ. of Washington, 1981.

[Bell 83]
Bell, Jean L., *Data Structures for Scientific Simulation*, Ph.D. dissertation, Univ.

of Colorado, Dept. of Computer Science, 1983

[Bolour et al 82]
Bolour, A., Anderson, T.L., Dekeyser, L.J., Wong, H.K.T., The Role of Time in Information Processing: A Survey, *ACM-SIGMOD Record*, 12, 3, 1982, pp. 27-50.

[Eggers & Shoshani 80]
Eggers, S. J., Shoshani, A., Efficient Access of Compressed Data, *Proceedings of the International Conference on Very Large Databases*, 6, 1980, pp. 205-211.

[Hampel & Ries 78]
Hampel, M., Ries, D. R., Requirements for the Design of a Scientific Data Base Management Systems, Special Report on *Generalized Data Management Systems and Scientific Information*, OECD Nuclear Energy Agency, Paris, 1978, pp. 111-131.

[Lorie 82]
Lorie, R.A., Issues in Databases for Design Applications, *File Structures and Data Bases for CAD*, J. Encarnacao and F.L. Krause (eds), North-Holland, 1982.

[McCarthy 82]
McCarthy J., Meta data Management for Large Statistical Databases, *Proceedings of the International Conference on Very Large Data Base (VLDB)*, 1982.

[Nievergelt et al 84]
Nievergelt J., Hinterberger H., Sevcik K.C., The Grid File: An Adaptable, Symmetric Multikey File Structure, *ACM Transactions on Database Systems 9*, 1, March 1984, pp. 38-71.

[Smith & Smith 77] Smith, J. M., and Smith, D. C. P., Database abstractions: aggregation and generalization, *ACM Transactions on Database Systems 2*, 2, June 1977, pp. 105-133.

[Shoshani 82]
Shoshani, A., Statistical Databases: Characteristics, Problems, and Some Solutions, *Proceedings of the 8th International Conference on Very Large Data Bases (VLDB)*, 1982, pp.208-222.

Table 1: Summary of Properties of Experiment Data

	Regular Sparse		Regular Dense		Irregular Dense			
Properties	Time Projection Chamber	Limited Track Recon-struction	Hydro-Dynamics	NMR Spec-troscopy	Heavy Ion	Passive Solar	Turbu-lent Flow (Vortex)	Laser Isotope Separation
EXPERIMENT/ SIMULATION	E	E	S	E	E	S	S	E
IDENTIFIER (KEY)								
Regular	•	•	•	•				
Dense			•	•	•	•	•	•
Time Variation			(•)				•	
ACCESS PATTERN								
Access Type								
Exact	•		•	•	•	•		
Range		(•)	(•)				(•)	(•)
Proximity	•		•			•	•	(•)
Access Sequence								
Local	(•)		(•)	(•)			(•)	(•)
Non-Local	(•)			(•)	•			
Linear			(•)	(•)				
Arbitrary	•	•	(•)		•	(•)		
SIZE (bytes)								
Per Unit	10^{4-5}	10^{3-4}	10^{7}	–	10^{3-4}	10^{4-5}	10^{6}	–
Per Collection	10^{11}	10^{10}	10^{8-9}	10^{5-7}	10^{10-11}	10^{7-8}	10^{8}	10^{7-8}
Total	10^{12}	10^{11}	10^{11-12}	10^{8-10}	10^{12}	10^{10-11}	10^{10}	10^{10}
ASSOCIATED DATA								
Configuration	•	•	•	•	•	•	•	•
Instrumentation	•	•		•	•			•
Analyzed	•	•	•	•	•		•	•
Summary	•	•	•	•	•	•		•

• - applies often
(•) - applies sometimes

Table 2: Summary of Properties of Associated Data

Properties	Config-uration	Instru-mentation	Analyzed Data	Summary Data	Property Data
ACCESS PATTERN					
Access Type					
Exact	*	*			*
Range			*	*	*
Proximity	(*)				(*)
Partial			*	*	*
Access Sequence					
Local	(*)	(*)	(*)		
Non-Local	*	*	*	*	*
Linear					*
Arbitrary			(*)		
DATA MODELING					
Geometric	*				
Entities	(*)	*	(*)		*
Hierarchical	(*)		(*)		*
Network	(*)				(*)
Generalization		*			(*)
N-Dimensional			*	*	
Meta-Data			*	*	
USAGE					
Schema Variation			*	*	
Historical	(*)	*	*	*	
Stability	*		*	*	
NON-STANDARD DATA TYPES					
Graphs			(*)	*	*
Text	(*)	(*)			*
Time Series		*	(*)	(*)	
Special Symbols					*
Non-Scalar			*	*	
SIZE					
Bytes	10^6	10^7	10^{8-9}	10^6	10^{7-8}
Percentage	1%	1%	50-100%	1-10%	

* - applies often
(*) - applies sometimes