SEPTEMBER 1990, VOLUME 13, NO. 3

a quarterly bulletin of the IEEE Computer Society technical committee on

Data Engineering

CONTENTS

SPECIAL ISSUE ON STATISTICAL AND SCIENTIFIC DATABASES

Data Engineering Bulletin is a quarterly publication of the IEEE Computer Society Technical Committee on Data Engineering. Its scope of interest includes: data structures and models, access strategies, access control techniques, database architecture, database machines, intelligent front ends, mass storage for very large databases, distributed database systems and techniques, database software design and implementation, database utilities, database security and related areas.

Contribution to the Bulletin is hereby solicited. News items, letters, technical papers, book reviews, meeting previews, summaries, case studies, etc., should be sent to the Editor. All letters to the Editor will be considered for publication unless accompanied by a request to the contrary. Technical papers are unrefereed.

Opinions expressed in contributions are those of the individual author rather than the official position of the TC on Data Engineering, the IEEE Computer Society, or organizations with which the author may be affiliated.

Membership in the Data Engineering Technical Committee is open to individuals who demonstrate willingness to actively participate in the various activities of the TC. A member of the IEEE Computer Society may join the TC as a full member. A non-member of the Computer Society may join as a participating member, with approval from at least one officer of the TC. Both full members and participating members of the TC are entitled to receive the quarterly bulletin of the TC free of charge, until further notice.

Letter from the Editor

This issue of **Data Engineering Bulletin** is devoted to Statistical and Scientific Databases. It contains short versions of six papers and two panel descriptions from the *Fifth International Conference on Statistical and Scientific Databases,* which was held in Charlotte, North Carolina on April 3-5, 1990. For interested readers the full proceedings of the conference is also available as:

Proceedings of the Fifth International Conference on Statistical and Scientific Databases, Lecture Notes in Computer Science, Vol. 420, Springer Verlag, 1990.

Also included in this issue is a paper summarizing the *Scientific Database Workshop,* May 12-13, 1990, which was supported by the NSF. This paper was presented in the Statistical and Scientific Databases Conference. The papers included in this issue are recommended by the program committee chairman of the conference Zbigniew Michalewicz, who also contributed a nice overview of both the current research in Statistical and Scientific Databases, and the conference.

I hope this special issue of **Data Engineering** will help broaden and increase the research interest in Statistical and Scientific Databases. I would like to thank each of the authors for writing short versions of their papers, and to Zbigniew Michalewicz for his help in putting this issue together. Since this is the last **Data Engineering** issue that I edit, I would also like to thank the editor in chief, Won Kim for making my term as an editor an enjoyable as well as a rewarding experience.

Z. Meral Ozsoyoglu
Cleveland, Ohio
August, 1990

Current Research in Statistical and Scientific Databases: the V SSDBM

Scientific and statistical databases (SSDB) have some special characteristics and requirements that cannot be supported by existing commercial database management systems. They have different data structures, require different operations over the data, and involve different processing requirements. SSDB's typically contain static data which are comprised of numeric and discrete values. Data tend to be large, they contain a large number of nulls; additional adequate description of the quantitative information must be included. Queries to a SSDB very often are aggregation and sampling, thus requiring special techniques.

Computer scientists, statisticians, database designers and data analysts agree that there are no characteristics which uniquely identify statistical/scientific database management system. However, it can be informaly defined as a database management system able to model, store and manipulate data in a manner suitable for the needs of statisticians and to apply statistical data analysis techniques to the stored data.

At the beginning of the previous decade some researchers began to look at some problems characteristic for statistical/scientific databases. This resulted in a series of conferences on statistical and scientific database management (SSDBM).

The V SSDBM (Charlotte, North Carolina, April 3–5, 1990) continued the series of conferences started nine years ago in California (1981 and 1983), then in Europe (Luxembourg, 1986, and Rome, 1988). The purpose of this conference was to bring together database researchers, users, and system builders, working in this specific area of activity, to discuss the particular issues of interest, to propose new solutions to problems, and to extend the themes of the previous conferences, both from the theoretical and the application point of view.

The Conference was hosted by UNC-Charlotte and sponsored by: UNC-Charlotte; Statistical Office of the European Communities—EUROSTAT (Luxembourg); Ente Nazionale Energie Alternative (Italy); Statistics Sweden; Microelectronics Center of North Carolina; International Association for Statistical Computing; Department of Statistics (New Zealand); and Istituto di Analisi dei Sistemi ed Informatica del CNR (Italy).

The papers presented during the conference cover a wide area of research for statistical and scientific databases: object oriented database systems, semantic modelling, deductive mathematical databases, security of statistical databases, implementational issues for scientific databases, temporal summary table management, graphical and visual interfaces for statistical databases, query optimization, distributed databases, and economic and geographical databases. In addition to traditional topics new topics of growing importance emerged. These include statistical expert systems, object oriented user interface, geographical databases, scientific databases for human genome project.

This special issue contains short versions of some of the papers presented at the conference. These papers reflect the diversity of approaches used to solve considered problems.

The paper, *The implementation of area and membership retrievals in point geography* by Andrew Westlake and Immo Kleinschmidt, identifies the conceptual and implementational problems in geographical database systems. The discussion is based on the experience of the Small Area Health Statistics Unit at the London School of Hygiene and Tropical Medicine which investigate the geographical distribution of some diseases in UK.

In the paper *STORM: A statistical object representation model* by Maurizio Rafanelli and Arie Shoshani, the authors discuss the development of a Statistical Object Representation Model (STORM), which captures the structural semantic properties of data objects for statistical applications. They discuss the deficiencies of current models, and show that the STORM model captures the basic properties of statistical objects in a concise and clear way. They also develop the conditions for a well-formed statistical object.

In *A genetic algorithm for statistical database security* by Zbigniew Michalewicz, Jia-Jie Li, and Keh-Wei Chen, the authors demonstrate the usage of genetic algorithms in enhancing the security of statistical databases. This may constitute the first step towards a construction of Intelligent Database Administrator, built as a genetic process running in the backround of the system. It has a potential of taking some of the responsibilities from database administrator like clustering, indexing, provision of security, *etc.*

In *Temporal Query Optimization in Scientific Databases* by Himawan Gunadhi and Arie Segev, the authors address important issues related to query processing in temporal databases. In order to exploit the richer semantics of temporal queries, they introduce several temporal operators and discuss their processing strategies.

The paper *A visual interface for browsing and manipulating statistical entities* by Maurizio Rafanelli and Fabrizio Ricci, presents a proposal for a visual interface which can be used by statistical users. By means of this interface it is possible to browse in the database, to select topic-classified statistical entities and to manipulate these entities by carring out queries based on an extension of the STAQUEL query language.

In *A Scientific DBMS for Programmable Logic Controllers* by Gultekin Ozsoyoglu, Wen-Chi Hou and Adegbemiga Ola, the authors present the design of a programmable logic controller (PLC) database system which is a single-user, real-time, scalable, and main-memory-only system. PLCs are special-purpose computers commonly used in scientific computing applications.

There were also two panel sessions: the first one on *Scientific Data Management for Human Genome Applications*, chaired by A. Shoshani, the other one on *Expert Statistical Systems*, chaired by Roger Cubitt. Members of the first panel described the data structures and operations associated with Genomic data, which includes DNA structures and various genomic maps. The main observation made was that the support for data of type "sequence" is essential, a concept which is foreign to current commercial relational database technology. Another important observation was the need for extensibility, that is the need for user interfaces, programming languages, and persistent databases with basic object capabilities that can be extended (customized) to the particular domain of Genomic data. The most pressing capability is persistent object storage. The second panel concentrated on problems related to the development of expert systems in the areas of statistical and scientific data processing and analysis. This topic is of growing concern: the Statistical Office of European Communities has launched for 1989–1992 a special program "DOSES" of research specifically in the areas of development of statistical expert systems. The reports from both panels appear in this issue.

On March 12-13, 1990, the National Science Foundation sponsored a two day workshop, hosted by the University of Virginia, at which representatives from the earth, life, and space sciences gathered together with computer scientists to discuss the problems facing the scientific community in the area of database management. During the V SSDBM there was a special presentation *On the NSF Scientific Database Workshop* by James C. French, Anita K. Jones, and John L. Pfaltz (Institute of Parallel Computation). Their paper summarizes the discussion which took place at that meeting.

The proceedings from the conference were published by Springer-Verlag in the Lecture Notes in Computer Science series (Volume 420) and are available from the publisher.

The Sixth International Conference on Scientific and Statistical Database Management (note the change in the order of terms: *statistical* and *scientific*) is scheduled to take place in Switzerland in June 1992. At the end of this issue there is the preliminary call for papers for this conference.

Charlotte, N.C., May 21st, 1990

Zbigniew Michalewicz
V SSDBM Chairman

THE IMPLEMENTATION OF AREA AND MEMBERSHIP RETRIEVALS IN POINT GEOGRAPHY USING SQL

A. J. Westlake and I. Kleinschmidt
Small Area Health Statistics Unit
Department of Epidemiology and Population Sciences
London School of Hygiene and Tropical Medicine

1 Introduction

1.1 Background

The Black Advisory Group [Blac84], in the course of its enquiry into childhood cancers around the nuclear reprocessing plant at Sellafield in Cumbria, experienced delays and difficulties, first in assembling local statistics and then in assessing them in relation to regional and national experience. The group concluded (Recommendation 5) "that encouragement should be given to an organisation [...] to co-ordinate centrally the monitoring of small area statistics around major installations producing discharges that might present a carcinogenic or mutagenic hazard to the public."

Subsequent events have underlined the importance of this conclusion, as other reports have arisen of possible excesses of malignancies near nuclear installations. There are also analogous — and more numerous questions concerning contamination of the environment by industrial chemicals, effluents, and smoke. These too call for a similar system to provide ready access to the health statistics of defined local areas, and means for interpreting them. The need applies primarily but not exclusively to cancers, and it applies to all ages.

Arising from these concerns the Small Area Health Statistics Unit (SAHSU) was inaugurated in the latter part of 1987.

1.2 Data

The UK Office of Population, Censuses and Surveys (OPCS) is the primary source of our data, and we work in close collaboration with them. For events we hold national death certificate data for all deaths in Great Britain (excluding personal identification) from 1981 to 1988, plus similar records on cancer registrations (eventually from 1974). All event data is augmented annually, some 900,000 records for each year.

We hold population data from the 1981 census as aggregate records for each of the 142,000 Enumeration Districts (EDs), with an average population of about 400 persons. Other population data and estimates are held for whatever aggregation units are available.

2 RDBMS or GIS

Computer Scientists and Geographers have developed many methods of physical database organisation for spatial structures (a good summary can be found in [RhOG89]) and some of these methods are implemented in the various specialised Geographical Information System (GIS) products which are commercially available. On the other hand, commercially available relational database management systems (RDBMSs) do not sup- port directly any of these special structures. RDBMS systems have been developed to meet the major commercial market for transaction processing, and so concentrate on efficiency in that application. In consequence, indexed files are the only form of physical storage on which we can depend in an RDBMS.

2.1 Postcodes

The postcode system in the UK was begun about twenty years ago and has been in full operation for the last ten years. All birth, death and cancer registration now carry the postcode of residence of the individual, and have done so since the early eighties. The accuracy of these files is being checked by OPCS, and cancer registrations are being postcoded retrospectivly from 1974, using commercial services and a Post Office directory which links a postcode to each of the 12 million postal household addresses in the UK.

The most detailed codes are related to individual bundles of mail, and identify on average only about 12 households, with 1.6 million codes covering the whole of the UK. Individuals generally know and use their postcodes, since this speeds up postal deliveries. The Post Office and OPCS have produced a Post-Code Directory (the Central Postcode Directory - CPD) which lists all the postcodes in use throughout the UK. The CPD also includes a Grid Reference location (see section 3.4), plus a link to electoral Wards and hence to administrative geography.

2.2 Choice of approach

In many situations the statistical requirements for the storage and retrieval of data with a geographical component can be met efficiently by using a hierarchical organisation, which is easily implemented within the standard relational model. What you miss compared with a GIS is the direct representation of physical structures and the considerable emphasis on the production of maps as output.

The very small postcode areas can be used as building blocks for larger (aggregate) areas. Since all our areas of interest contain at least several tens of postcodes (and often hundreds or thousands) the postcode locations provide a perfectly adequate approximation for representing the location and extent of these larger areas. So after considerable discussion we decided to build the system using a RDBMS, rather than a specialised GIS. Subsequent sections describe how the required structure and retrievals have been implemented within the relational model.

3 Implementation of structure

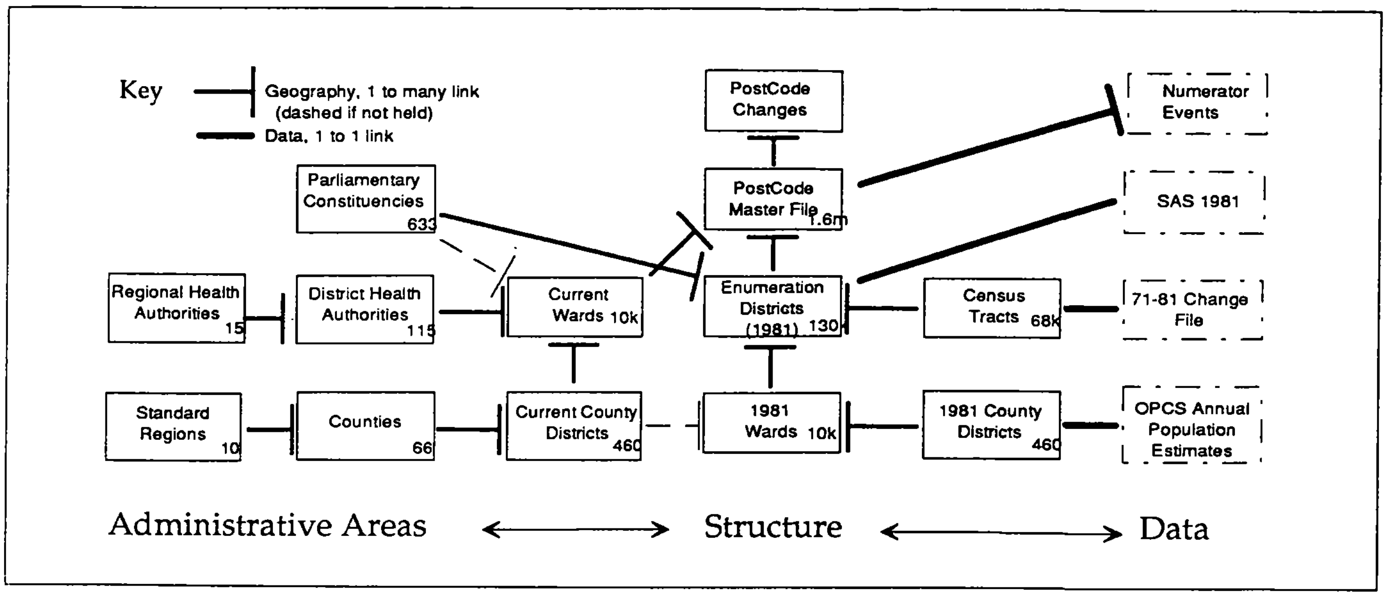
3.1 General structure

The general structure of the SAHSU database is shown in Figure 1.

The geographical structure is centred on postcodes. These have a grid reference which gives them a physical location. Other geographical areas are represented as separate tables, with links representing the hierarchical structure of aggregate areas, i.e. small areas contain fields (foreign keys) containing the ID of the next larger areas of which they are a part.

Note that with this structure we are representing the memberships in the data. We can know that one administrative area is part of another one, but we do not know directly where anything is (except for the individual postcodes).

Event records all contain a postcode field, giving a link to both the administrative geographies of census and local government and to the physical geography of the country through grid references.



Key
Geography, 1 to many link (dashed if not held)
Data, 1 to 1 link
PostCode Changes
Numerator Events
Parliamentary Constituencies 633
PostCode Master File 1.6m
SAS 1981
Regional Health Authorities 15
District Health Authorities 115
Current Wards 10k
Enumeration Districts (1981) 130k
Census Tracts 68k
71-81 Change File
Standard Regions 10
Counties 66
Current County Districts 460
1981 Wards 10k
1981 County Districts 460
OPCS Annual Population Estimates
Administrative Areas
Structure
Data

Figure 1: General view of Geographical Aggregate Structure.

Census records link to the ED table, and other statistical data can be linked to any other part of the structure.

3.2 Resolution

The hierarchy of administrative geographical units (regions, constituencies, districts, wards, EDs, postcodes) form irregular nested tessellations. The spatial resolution of these tessellations is variable, in that districts are of varying physical size. For the small areas (Postcodes, EDs and Wards) the size is chosen to include approximately equal populations, that is inversely proportional to population density. Spatial resolution in our units therefore reflects the population density, with administrative units becoming physically larger where the population is sparse.

Population (census) data are only available for census enumeration districts (EDs) and larger areas. This presents some problems in the analysis stage, which would disappear if we worked with EDs as building blocks (and to do so could also simplify the data structures). We decided, however, to retain the extra precision of postcodes and face the consequent matching and population estimation problems.

3.3 Temporal Stability

A potential drawback with this data structure is that it is data dependent and so can change over time. When reporting for aggregate areas the current area definitions are appropriate, and so these must be updated as they change. Population aggregates must be linked to the areas as defined at the time of data collection. For event records we must know the date for which the postcode applies. We maintain the postcode master table in its current form, with an indicator for the existence of any previous definitions in the postcode changes table.

3.4 The UK Grid Reference system

Latitude and longitude give a world-wide system for specifying location, but have the disadvantage that trigonometrical calculations are needed to convert them to distances on the ground. In Great Britain a grid reference (GR) system is used based on kilometre squares, aligned to true North at the Greenwich meridian, and with an origin somewhere in the Atlantic south of Ireland. This allows the calculation of distances using Pythagorus' Theorem anywhere in GB. Grid references are given as an Easting coordinate and a Northing coordinate to a specified precision (corresponding to cartesian x and y coordinates).

3.5 Data volume

The major components of the storage requirement are shown in Table 1. The main data tables (events, postcodes, census enumeration districts) are very numerous, but the most significant administrative areas (constituencies, health districts) are relatively few, rarely more than a few hundred. Similarly, with rare diseases the areas which we study will usually contain only a few cases. So in some senses the problems we want to study are small even though the overall volume of data is large. The task we face is to design retrievals (and the corresponding database structure) so that the time taken is

proportional to the size of the problem (or the size of the result table) rather than the size of the main database tables.

		Fields			Indexes		
Table	Rows	Num.	Bytes	Mb	Num.	Mb	Total Mb
Cancer	4,000,000	11	39	163.84	2	169.56	333.40
Death	5,850,000	15	52	323.81	2	247.98	571.79
SAS_81	142,000	201	612	96.94	1	3.10	100.04
Tract_81	68,000	101	312	23.21	1	1.48	24.70
PC	1,600,000	11	51	86.24	6	197.66	283.90
ED_81	142,000	7	40	6.06	5	14.97	21.03
Ward	10,000	3	16	0.17	3	0.57	0.74
Ward_81	10,000	6	31	0.33	4	0.78	1.11
Totals				700.60	24	636.11	1,336.71

Table 1: Space requirements

3.6 Relational efficiency

Many systems for physical storage have been proposed and shown to be particularly efficient for particular modes of data usage. Note that such physical query optimisation choices do not violate the relational model, provided they do not exclude other queries (though they may make them less efficient). Unfortunately the option to use such specialised storage structures is not generally available in the available DBMSs, so we are required to work with the features provided. Even when limited to the specification of indexes the designer of the database can significantly affect performance by the choice of physical data organisation. Most real data access makes extensive use of natural joins (equality of key fields) for subsets of records, and for these indexes usually prove acceptably efficient. Thus with the SAHSU data structure we can build indexes to achieve reasonable efficiency for our pre-defined aggregate area retrievals.

Since most of our queries involve small results (being based on relatively small areas and rare diseases) we are confident that the large size of the postcode and event tables will not be a problem. This is certainly borne out by our experience so far, since extending the postcode table from just Scotland to cover the whole of GB (a tenfold increase) had a barely noticeable effect on performance. There is a large cost involved in creating an index for a large table, but this is only done once (since the data are static) and not under the pressure of a real enquiry.

4 Retrievals

Retrievals for administrative areas from the SAHSU database are all of the same general form. For one or more areas of one or more types we must find the events (usually of a selected type) in the area (the numerator), and also the size of the population (denominator) in which those events occurred. We will invariable be interested in classified populations (at least by age and sex) and so the events must be similarly aggregated. The result will be a record for each subgroup containing the numerator count and population, from which a rate can (trivially) be computed.

4.1 Problems with Geographical Retrievals

The exact details of retrievals for arbitrary geographical areas cannot, in general, be anticipated, since a query may be based on any part of the country and may include an area of any shape. It would be theoretically possible to anticipate a number of different types of query and build appropriate retrieval structures for them, such as storing the distances from a grid of significant points, for example,

major towns. For any reasonable level of choice, however, this would involve an unacceptable overhead in storage for the various keys and indexes.

An alternative approach is to use some form of area indexing. The Quad-Tree approach is very attractive [Laur87, Hogg88]. In this model areas are approximated by sets of squares of differing sizes. Any square can be dis-aggregated recursively into four component squares, and so on down to a level of resolution that provides an adequate representation of the area in that location. Each square is represented by a key (based on its location) and a size attribute. This method combines the advantages of regular tessellations with (roughly) equal representational precision for the target characteristic. Unfortunately, the data structure is not provided in most available RDBMSs, and implementation of the algebra required to support the model [LaMi89] did not seem to us to be reasonable with the tools at our disposal. Instead, we decided to look for a simplified method which could give most of the benefits, at minimal cost for implementation.

We restrict attention here to the simplest case of an arbitrary geographical query, namely one based on a circle with arbitrary centre and radius (though small compared with the whole country). As before we need to find the set of postcodes (building blocks) included within the area, and for these retrieve selected events (numerators) and population estimates (denominators). So far this is the only form of query implemented, though our procedures are chosen to generalise to more complex areas.

Our solution is to use a simple and efficient intermediate method to select a small superset quickly, and then to apply an accurate selection to this set. This will now be a small problem so we can afford to do an expensive calculation.

For reliably efficient operation we need to construct a key (which we can index) on which we can do a natural join to select postcodes, since we know that such joins are efficient. Drawing on the quad-tree idea we define a set of covering areas (a tessellation), from which we first select an approximation to the area of interest. We can then find the postcodes located within this set of areas, and finally select more accurately from within the selected set of postcodes. For this to be efficient overall it is an essential requirement that the initial operation of selecting the required set of covering areas can be done cheaply. This requirement is met if we can compute (rather than select) the set.

We required a system which we could implement simply and decided to use equal sized squares based on the grid reference system. After some experimentation we decided to use one-kilometre squares for the grid-square system. The ID for each square is obtained by concatenating the kilometre precision easting and northing coordinates of the south-west corner of the squares.

4.2 Retrievals of postcodes for circles

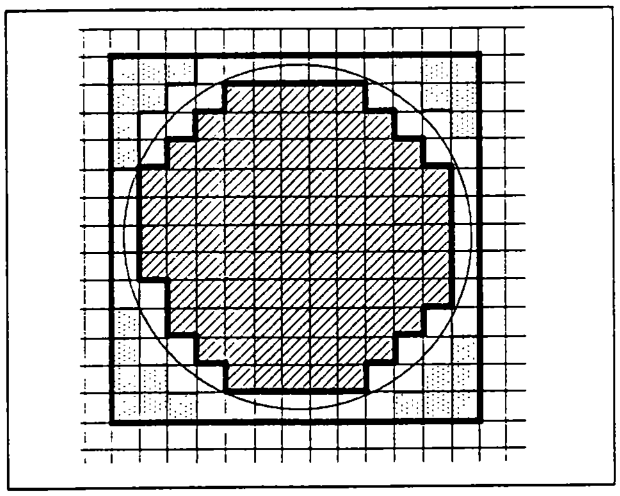


Figure 2: Selection of squares covering a circle.

A circle is specified by its centre and radius. The centre may be given as a grid reference point or as a postcode. The system then finds all the postcodes which have their grid reference within this circle.

It is not feasible to calculate for each postcode its distance from the centre of a specified circle in order to determine its membership of the circle. Indexing on grid references in the postcode table is of no help here, since it is the distance from the (arbitrary) circle centre, rather than the grid reference itself that determines whether a postcode is selected.

Once a circle has been specified a procedure (written in C using embedded SQL) determines all the grid squares that either completely or partially overlap with the circle. A marker is added to show whether a square is cut by the circle. A temporary table is constructed containing the IDs of the included squares and the cutting marker.

This temporary table is then used in a join operation with the postcode table, selecting all postcodes within the contained squares and computing the inclusion criterion for postcodes in the cut squares. This takes the following form:

```
SELECT Postcode, ...
```

```
FROM Postcode, TempSq
WHERE TempSq.GrSq = Postcode.GrSq
   AND (Cut = 0
    OR (Cut = 1
       AND (East - Centre-East)^2 + (North - Centre-North)^2 ≤ Radius^2))
```

The selected postcodes are placed in another temporary table from where they can be used to select deaths in the circle and to control the estimation of the population in the circle.

The set of selected postcodes define an area which is an approximation to the underlying circle. The error in approximation will clearly depend on both the size of the circle and the size of the postcodes. However, postcodes are large where population is sparse so that large circles are needed, with the reverse being true in areas of dense population. It is thus a reasonable rule (in areas of similar population density) that the relative error in the fit of postcodes to the circle depends on the included population rather than the exact dimensions involved. Since we are usually interested in rare diseases for which large populations need to be studied we can be confident that the postcodes will give a (relatively) good fit to the circle.

4.3 Calculation of denominators

As described previously, the smallest area for which denominator data are available are enumeration districts (EDs) - which are made up of about 10 postcodes on average.

Since numerator and geographic data are given by postcode, and the resolution of the circle algorithm is such that it selects postcodes rather than EDs, some method is needed for reconstituting EDs out of their constituent postcodes in order to estimate denominators. If the boundary of a circle cuts across ED boundaries then it will be necessary to decide what population total to estimate for the partial EDs.

We use a simple rule to allocate a proportion of the ED population to the circle, ie to project the ED population down onto the postcodes. In the absence of any other information we use the proportion of the ED postcodes which are actually selected for the circle. The assumption underlying this method is that the population of the ED is evenly distributed throughout its constituent postcodes.

The core of the algorithm is the following SQL statement.

```
SELECT pcs-temp.ED, COUNT (DISTINCT pcs-temp.Postcode)
        / COUNT (DISTINCT Postcode.Postcode)
    FROM pcs-temp, Postcode
    WHERE pcs-temp.ED = Postcode.ED
    GROUP BY pcs-temp.ED
```

This produces a table with one record for each ED which had at least one postcode in the circle, plus the proportion of codes in each ED included in the circle. This description is in terms of the total population, but the system can also operate with classified populations, currently rates for specific age groups, separately for men and women.

5 Conclusions

There are a number of serious problems in making use of locational information in a statistical database. Computer scientists and geographers have developed a number of specialised solutions to these problems, and by studying their work we are able to develop simplified versions which can be implemented efficiently using standard DBMS systems.

The other big problem which we face is the organisation of meta data for the geography, attributes and aggregations in the database, in order to provide simple access to end-users and to control the classification and aggregation processes performed by our front-end applications. This work is not yet complete, and will be the subject of a further paper.

6 References

Blac84 Investigation of the Possible Increased Incidence of Cancer in West Cumbria. Report of the Independent Advisory Group (Chairman, Sir Douglas Black). London: HM Stationery Office, 1984.

Hogg88 J. Hogg. Representing Spatial Data by Linear Quadtrees. Computing, March 10, 1988.

LaMi89 R. Laurini and F. Milleret. Spatial Data Base Queries: Relational Algebra versus Computational Geometry. In [RaKS89].

Laur87 R. Laurini. Manipulation of Spatial Objects with a Peano Tuple Algebra. University of Maryland, Centre for Automation Research Technical Report (1987).

RaKS89 M. Rafanelli, J. Klensin and P. Svensson (Eds.). Statistical and Scientific Database Management, IV SSDBM. Lecture Notes in Computer Science, Vol 339, Springer Verlag, 1989.

RhOG89 D. Rhind, S. Openshaw and N. Green. The analysis of Geographical Data: Data rich, Technology adequate, Theory poor. In [RaKS89].

STORM: A STATISTICAL OBJECT REPRESENTATION MODEL

Maurizio RAFANELLI+, Arie SHOSHANI*

+ Istituto di Analisi dei Sistemi ed Informatica
viale Manzoni 30, 00185 Roma, Italy

* Information & Computing Sciences Division
Lawrence Berkeley Laboratory
1 Cyclotron Road, Berkeley, CA 94720, USA.

Abstract. In this paper we explore the structure and semantic properties of the entities stored in statistical databases. We call such entities "statistical objects" (SOs) and propose a new "statistical object representation model", based on a graph representation. We identify a number of SO representational problems in current models and propose a methodology for their solution.

1.0 INTRODUCTION

For the last several years, a number of researchers have been interested in the various problems which arise when modelling aggregate-type data [SSDBM]. Since aggregate data is often derived by applying statistical aggregation (e.g. SUM, COUNT) and statistical analysis functions over *micro-data* the aggregate data bases are also called "statistical databases" (SDBs) [Shoshani 82], [Shoshani & Wong 85].

This paper will consider only aggregate-type data, a choice which is justified by the widespread use of aggregate data only i.e. without the corresponding micro-data. The reason is that it is too difficult to use the micro-data directly (both in terms of storage space and computation time) and because of reasons of privacy (especially when the user is not the data owner).

Statistical data are commonly represented and stored as statistical tables. In this paper we show that these tables are complex structures that may have many possible representations (e.g. tables, relations, matrixes, graphs). Accordingly, we use the term "statistical object" (SO) for the conceptual abstraction of statistical data.

Various previous papers have dealt with the problem of how to logically represent an aggregate data reality (e.g. [Chan & Shoshani 81, Rafanelli & Ricci 83, Ozsoyoglu et al 85]). Starting from those works, this paper will propose a new "statistical object representation model" (STORM), based on a graph representation. In the subsequent sections, after the necessary definitions, the proposed structure for a SO will be discussed and developed. We follow the definition of the STORM model with an investigation of a well-formed SO, and develop conditions for it.

2.0 PROBLEMS WITH CURRENT LOGICAL MODELS

2.1 BASIC CONCEPTS

We start this section by briefly presenting four basic concepts that are unique to SDBs, and then discuss deficiencies of currently proposed models.

1. *Summary attributes* -- these are attributes that describe the quantitative data being measured or summarized. For example, "population", or "income for socio-economic databases", or "production and consumption of energy data".

This work was partially supported by the Office of Health and Environmental Research Program and the Director, Office of Energy Research, Applied Mathematical Sciences Research Program, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

2. *Category attributes* -- these are attributes that characterize the summary attributes. For example, "Race" and "Sex" characterize "Population counts", or "Energy type" and "Year" characterize the "production levels of energy sources".

3. *Multi-dimensionality* -- typically a multidimensional space defined by the category attributes is associated with a single summary attribute. For example, the three- dimensional space defined by "State", "Race" and "Year" can be associated with "Population". The implication is that a combination of values from "State", "Race" and "Year" (e.g. Alabama, Black, 1989) is necessary to characterize a single population value (e.g.10,000).

4. *Classification hierarchies* -- a classification relationship often exists between categories. For example "Cities" can be classified into "States", or specific "professions" (e.g., "civil engineers", "chemical engineer", "college professor", high school teacher", etc.) can be grouped into "professional categories" (e.g., "engineering", "teaching", etc.)

These basic concepts are addressed in different models currently used to describe statistical data by employing essentially two methodologies: a) 2-dimensional tabular representation and b) graph-oriented representation. We explore below some of the problems encountered using these methodologies in current models.

In the rest of the paper, we define a *STatistical Object Representation Model (STORM)* which is independent from the above methodologies. As a consequence, a SO can have a graphical representation, a 2-dimensional tabular representation, or any other representation preferred by the user (e.g. a "relation").

2.2 PROBLEMS WITH THE TWO-DIMENSIONAL TABULAR REPRESENTATION

The two-dimensional (2D) representation exists historically because statistical data have been presented on paper. This representation, although it continues to be practiced by statisticians today, the semantic concepts discussed above. We point out below several deficiencies.

2.1.1 *The concept of multi-dimensionality is distorted.*

By necessity, the multi-dimensional space needs to be squeezed into two dimensions. This is typically done by choosing several of the dimensions to be represented as rows and several as columns. For example, suppose that we need to represent the "Average Income" by "Profession", "Sex" and "Year" and "Professions" are further classified into "professional categories". Figure 1 is an example of a 2D tabular representation. Obviously, one can choose (according to some other preferred criteria) other combinations by exchanging the dimensions (e.g., "Year" first, then "Sex"), or put different dimensions as rows and columns.

Models using this tabular representation technique improperly consider the different tables to be different statistical objects, while in reality only the 2D representation has changed. In general, the 2D representation of a multi-dimensional statistical object forces a (possibly arbitrary) choice of two hierarchies for the rows and columns. The apparent conclusion is that a proper model should retain the concept of multi-dimensionality and represent it explicitly.

2.2.2 *The classification relationship is lost.*

In the 2D representation, classification hierarchies are represented in the same manner as the multi-dimensional categories. Consider, for example, the representation of "Professions"and"Professional Categories" shown in Figure 1.

Average Income in California			Professional Category					
			Engineer		Secretary		Teacher	
			Profession		Profession		Profession	
			Chemical Engineer	Civil Engineer	Junior Secretary	Executive Secretary	Elementary Teacher	College Teacher
Male	Year	80	*1,841*	*2,285*	*1,733*	*2,600*	*1,038*	*1,541*
		81	*2,012*	*2,411*	*1,819*	*2,678*	*1,090*	*1,641*
		82	*2,199*	*2,637*	*1,910*	*2,758*	*1,166*	*1,747*
		..	....	....	....	......	......	
		88	*3,749*	*4,521*	*2,560*	*3,293*	*1,701*	*2,500*
Sex								
Female	Year	80	*1,669*	*1,825*	*1,698*	*2,522*	*1,027*	*1,525*
		81	*1,825*	*1,996*	*1,783*	*2,597*	*1,079*	*1,624*
		82	*1,994*	*2,184*	*1,872*	*2,675*	*1,154*	*1,729*
		..	.....	......	......	......	......	
		88	*3,399*	*3,744*	*2,508*	*3,194*	*1,683*	*2,524*

Figure 1

As can be seen, there is no difference in the representation of "Sex" and "Year" and the representation of "Profession" and "Professional Category". However, it is obvious from this example that the values of average income are given for specific combinations of "Sex", "Year" and "Profession" only. Thus, "Professional Category" *is not* part of the multi-dimensional space of this statistical object. As can be seen from the above example, there is a fundamental difference between category relationship and multi-dimensionality. Usually, only the low-level elements of the classification relationship participate in the multi-dimensional space. This fundamental difference should be explicitly represented in a semantically correct statistical data model.

2.3 PROBLEMS WITH CURRENT GRAPH-ORIENTED MODELS

An attempt to correct some of the deficiencies of the 2D representation discussed above was made by introducing graph-oriented models. In these models the concepts of multi-dimensionality and classification hierarchies were introduced by having especially designated nodes. For example, in GRASS [Rafanelli 83] (which is based on SUBJECT [Chan 81]) multi-dimensionality is represented by A-nodes (A stands for "association") and C-nodes (C stands for "classification"). Thus, the statistical object of Figure 1 would be represented in GRASS as shown in Figure 2. Note that the node of the type S represents a "summary" attribute.

2.3.1 *Mixing categories and category instances.*

We refer to the classification hierarchy of "Professional Category" and "Profession" in Figure 2. Consider the intermediate node "Engineer". It has a dual function. On the one hand, it is an instance of the "Professional Category". On the other hand, it serves as the name of a category that contains "Chemical Engineer", "Civil Engineer", etc. Note that the category "Profession" is missing in this representation. The reason is that after we expand the first level ("Professional Category") into its instances, the next levels can contain only instances.

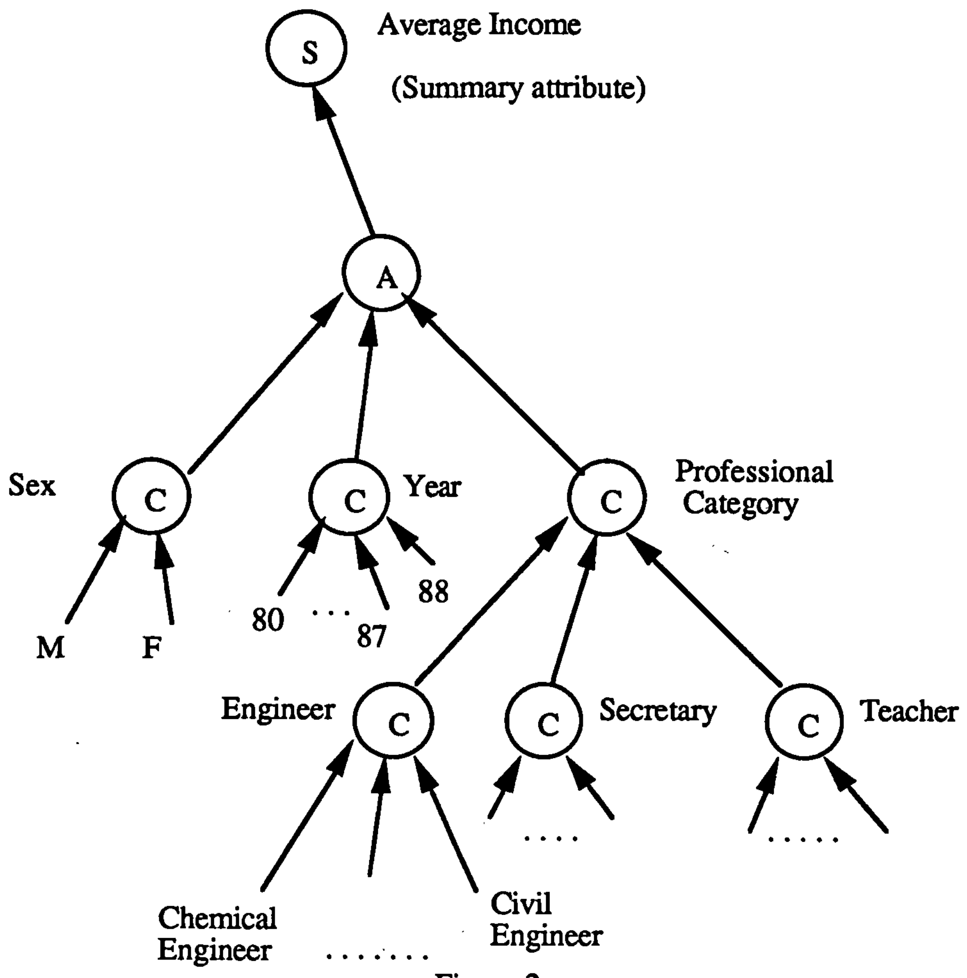


Figure 2

For the above reasons, we have chosen a graph model that separates the categories and their instances into two separate levels. For example, the statistical object of Figure 3 will be represented at the meta-data level (intentional representation) as shown in Figure 3. Underlying this representation the system stores and maintains the instances and their relationship. The instances can become visible to a user by using an appropriate command.

Note that an added benefit of representing only categories as in Figure 3 is its compactness as compared with the previous representations.

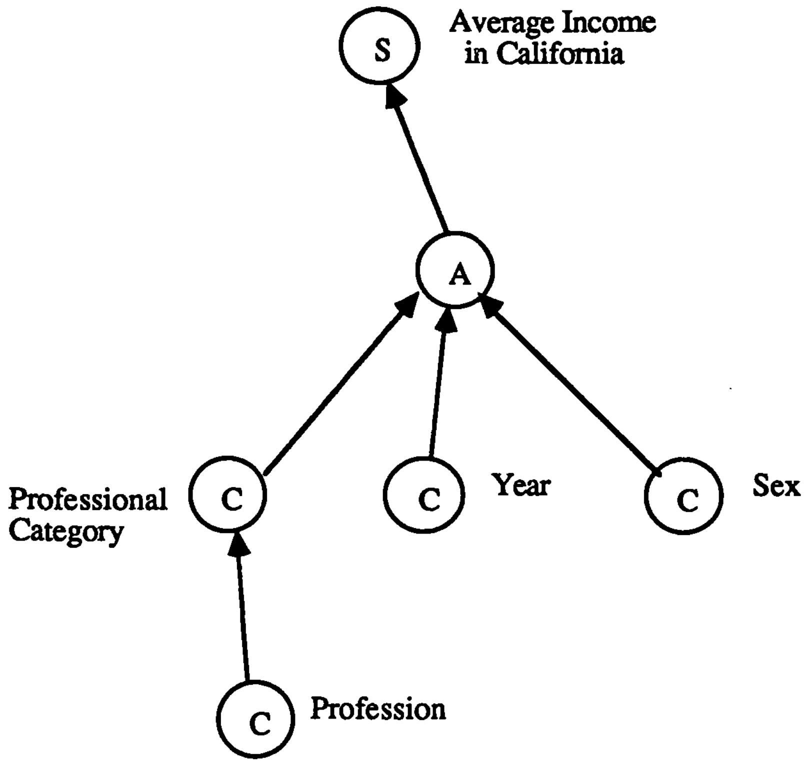


Figure 3

3.0 THE STORM MODEL

We can use the following notation to describe a SO:

$$N\ (C_{(1)}, C_{(2)}, ..., C_{(n)} : S),$$

where N and S are the name and summary attribute of the SO, and $(C_{(1)}, C_{(2)}, ..., C_{(n)}$ are the components of the category attribute set C. There is a function f is implied by the ":" notation, which maps from the Cartesian product of the category attributes values to the summary attribute values of the SO. For example, the following describes a SO on various product sales in the USA:

PRODUCT SALES (TYPE, PRODUCT, YEAR, CITY, STATE, REGION : AMOUNT)

As mentioned in the introduction, a statistical object SO represents a summary over micro-data. That summary involves some statistical function (count, average, etc.), and some unit of measure of the phenomena of interest (gallon, tons, etc.). Accordingly, the summary attribute has the two properties mentioned above: "summary type", and "unit of measure". In the example above, the summary type is SUM (or TOTAL), and the unit of measure DOLLARS. Note that the above SO is presumed to be generated over some micro-data, such as the individual stores where the products were sold.

In addition, we need to capture the structural semantics of the SO, i.e. the relationship between the category attributes as well. In the example above on "product sales", suppose that product "type" can assume the values: metal, plastic, and wood, and that "product" can assume the values: chair, table, bed. How do we know if sales figures are given for products, product types, or both? Further, suppose that we know that figures are given for products, how do we decide whether these figures can be summarized into product type? Similarly, we need to know whether sales figures for cities can be summarized to state levels and to regions. In order to answer these type of questions, we need to capture the structural semantics between category attributes. For that purpose, we use the STatistical Object Representation Model (STORM).

It is best to visualize the STORM representation of a SO in a graphical form as a directed tree. The summary attribute and each of the category attributes are represented as nodes of type S and C, respectively. The root of the tree is always the node S. In addition, another node type is used, denoted an A node, to represent an aggregation of the nodes pointing to it. In most cases the nodes pointing to an A node will be C nodes, but it is possible that an A node will point to another A node. An example of a STORM representation of the SO "product sales" mentioned previously is given in Figure 4. Note that an aggregation node has the domain generated by the cross product of its component domains. Thus, the node A pointed to nodes "type" and "product" in Figure 4, represents combinations of type and product.

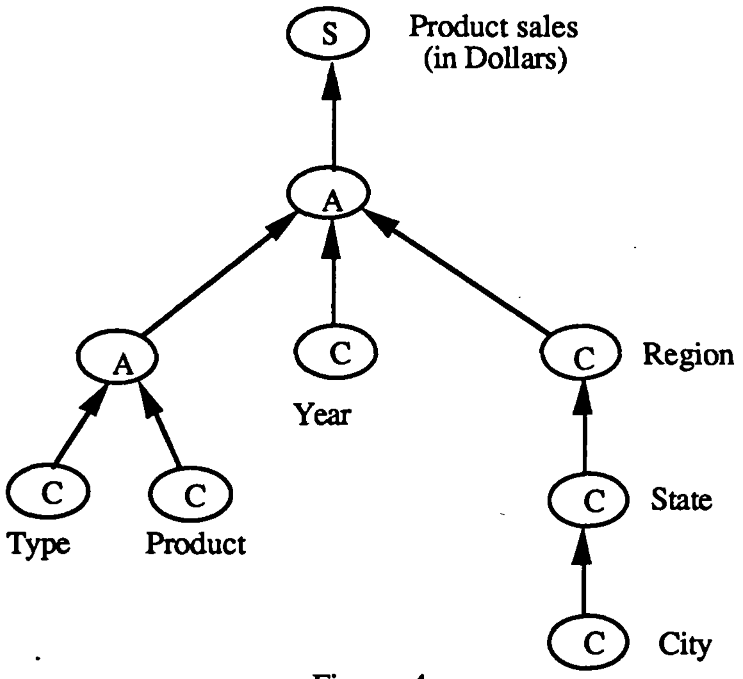


Figure 4

The STORM model is designed to support directly the four basic concepts of statistical data mentioned in Section 2.1. However, it puts limits on the structural interaction between the various constraints. These structural constraints are desirable for the conceptual simplicity of the model, yet are general enough for describing a rich variety of statistical objects. The structural constraints are summarized below.

A STORM representation of a SO is a directed tree of nodes of type S, A, and C, with the following structural constraints:

a) There is only a single S node and it forms the root of the tree.
b) A single A node points to the S node.
c) Multiple C or A nodes can point to an A node.
d) Only a single C or A node can point to another C node.

4.0 PROPERTIES OF STORM STRUCTURES

There are many possible ways of representing the category attributes and their interaction in a STORM structure. How do we choose a desirable representation? We illustrate the answer to this question with several examples.

The STORM representation of a SO implies a mapping between the nodes of the directed tree. We explore here the properties of the various possible mappings between category attributes. We refer again to the example given in Figure 4.

Let us first examine the mapping between "city" and "state". We assume that city names are unique within states, that is, each state can map into a single state. This mapping is therefore "single-valued", or in other words a function. Similarly, if we assume that states are unique within regions, then the mapping between the corresponding nodes will also be single-valued. In this case, the node that should be considered as relevant to the aggregation node A is only "city", because the product sales amounts are given for cities. However, the nodes "state" and "region" exist in that structure to indicate that the two single-valued mappings (city --> state, and state --> region) are also specified as part of the SO description, and therefore the sales amounts for states and regions can potentially be calculated. We call the ability for such summary type calculation "summarizability". Note that single valued mappings imply a classification relationship.

Now, let us consider the branch in Figure 4 that includes "type" and "product". As mentioned above, a product (such as "chair") can be of several types (such as "metal" or "wood"). Such a mapping is called multi-valued (it is obviously not a function). This multi-valued mapping implies that the sales figures are given for the combinations of "product" and "type" (e.g., "wood chairs"). Thus, the A node is needed to represent this multi-valued relationship. Note that in this case it is possible to summarize sales amounts both by "type" or by "product", in contrast to a single summarizability implied by single-valued mappings.

Because of space limitation, we cannot show here the precise arguments for desirable STORM structures. However, from the above examples one can observe the following proposition:

Proposition: A well-formed SO contains no multi-valued mappings along the branches of its tree, and no single-valued mappings between nodes that point to the same A-node.

BIBLIOGRAPHY

[SSDBM] Proc. of the last five conferences on Statistical and Scientific Database Management, 1981, 1983, 1986, 1988, 1990 (1988, 1990 published by Springer-Verlag).

[Chan & Shoshani 81] Chan P., Shoshani A. "SUBJECT: A Directory Driven System for Organizing and Accessing Large Statistical Databases" Proc. of the 7th Intern. Confer. on Very Large Data Bases (VLDB), 1981.

[Ozsoyoglu et al 85] Ozsoyoglu, G., Ozsoyoglu, Z.M., and Mata, F., “A Language and a Physical Organization Technique for Summary Tables”, Proc. ACM SIGMOD Conf., 1985.

[Rafanelli & Ricci 83] Rafanelli M., Ricci F.L. "Proposal of a Logical Model for Statistical Data" Base" in Proc. of the first LBL Workshop on Statistical Database Management, Menlo Park, CA, 1981.

[Shoshani 82] Shoshani A. Statistical Databases: Characteristics, Problems and Solutions" Proc. of the 7th Intern. Confer. on Very Large Data Bases (VLDB), Mexico city, Mexico , 1982.

[Shoshani & Wong 85] Shoshani A., Wong H.K.T. "Statistical and Scientific Database Issues" IEEE Transactions on Software Engineering, Vol.SE-11, N.10, October 1985.

A Genetic Algorithm for Statistical Database Security

Zbigniew Michalewicz* Jia-Jie Li† Keh-Wei Chen‡

Abstract

One of the goals of statistical databases is to provide statistics about groups of individuals while protecting their privacy. Sometimes, by correlating enough statistics, sensitive data about individual can be inferred. The problem of protecting against such indirect disclosures of confidential data is called the inference problem and a protecting mechanism—an inference control. During the last few years several techniques were developed for controlling inferences. One of the earliest inference controls for statistical databases restricts the responses computed over too small or too large query-sets. However, this technique is easily subverted. Recently some results were presented (see [Michalewicz & Chen, 1989]) for measuring the usability and security of statistical databases for different distributions of frequencies of statistical queries, based on the concept of multiranges. In this paper we use the genetic algorithm approach to maximize the usability of a statistical database, at the same time providing a reasonable level of security.

1 Introduction

One of the goals of statistical databases is to provide statistics about groups of individuals while protecting their privacy. Sometimes, by correlating enough statistics, sensitive data about individual can be inferred. When this happens, the personal records are compromised—we say, the database is *compromisable.* The problem of protecting against such indirect disclosures of confidential data is called the *inference problem.* During the last few years several techniques were developed for controlling inferences. One of the earliest inference controls for statistical databases (see [Denning et al., 1979], [Schlörer, 1980], and [Michalewicz, 1981]) restricts the responses computed over too small or too large query-sets; later (see [Denning & Schlörer, 1983]) it was classified as one of the cell restriction techniques. This technique is easily subverted—the most powerful tools to do it are called *trackers* (we will discuss them later in the text). However, query-set size controls are trivial to implement. Moreover, they can be valuable when combined with other protection techniques (see [Denning & Schlörer, 1983]), so they are worth some deeper examination.

A *statistical database* consists of a collection X of some number n of *records*, each containing a fixed number of confidential *fields.* Some of the fields are considered to be *category fields* and some to be *data fields* (the set of category fields need not be disjoint from the set of data fields). It is assumed that for any category field there is a given finite set of possible values that may occur in this field for each of the records. Data fields are usually numerical, *i.e.* it is meaningful to sum them up.

A *statistical query* has the form $COUNT(C)$, where C is an arbitrary expression built up from category-values (specifying a particular value for given category fields) by means of operators $AND(\cdot)$, $OR(+)$, and $NOT(\sim)$. The set of those records which satisfy the conditions expressed by C is called

*Department of Computer Science, University of North Carolina Charlotte, NC 28223, USA
†Department of Computer Science Victoria University of Wellington, New Zealand
‡Department of Mathematics, University of North Carolina Charoltte, NC 28223, USA

the *query set* X_C. The query-set size inference control is based on the following definition of the response to the query $COUNT(C)$:

$$COUNT(C) = \begin{cases} |X_C| & if\ \ k \leq |X_C| \leq n-k \\ \# & otherwise \end{cases}$$

where $|X_C|$ is the size (cardinality) of X_C; k is a certain integer, fixed for a given database, $0 \leq k \leq n/2$; and # denotes the fact that the query is unanswerable, *i.e.* the database refuses to disclose $|X_C|$ for the query. Usually the set of allowable queries in statistical database also includes other queries, such as averages, sums and other statistics, as:

$$SUM(C;j) = \begin{cases} \sum_{i \in X_C} v_{ij} & if\ \ k \leq |X_C| \leq n-k \\ \# & otherwise \end{cases}$$

where j is a data field and v_{ij} is the value in field j of record i. Generally, we will deal with arbitrary *additive queries* $q(C)$ satisfying the condition

$$X_{C_1} \cap X_{C_2} = \emptyset \implies q(C_1 + C_2) = q(C_1) + q(C_2),$$

or equivalently,

$$q(C_1 + C_2) = q(C_1) + q(C_2) - q(C_1 \cdot C_2)$$

(this condition is clearly satisfied for COUNT and SUM).

So the query-set size inference control is the following:

$$q(C) = \begin{cases} |q(C)| & if\ \ k \leq |X_C| \leq n-k \\ \# & otherwise \end{cases}$$

where $q(C)$ is an additive query and $|q(C)|$ denotes the answer for this query.

As we mentioned in the first sentence of the Introduction, the goal of statistical databases is to provide statistics about groups of individuals while protecting their privacy. In other words, we should balance between *usability* (provision of statistics) and *security* (protecting privacy) in statistical databases. These two concepts are essential in evaluating any inference control mechanism and they work against each other: it is intuitively clear that stronger security measures decrease usability of statistical database. In particular, a database which refuses to answer any query (null usability) has perfect security.

Before we proceed further we try to define in a formal way these two fundamental concepts. Having these definitions we will measure the "goodness" of different inference control mechanisms based on the idea of multiranges and (later) we use this measure as an evaluation function for our genetic algorithm.

Let us assume that Q is a set of statistical queries $q(C)$ (we denote by X_C the query set of $q(C)$). Let us denote $f_Q : \langle 0, n \rangle \longrightarrow \langle 0, \infty)$, a function which, for any integer j from the range $\langle 0, n \rangle$, returns the number of queries $q(C)$ from the set Q such that $|X_C| = j$. We will call the function f_Q the query distribution function; later (to simplify derived formulae) we assume that it satisfies $f_Q(i) = f_Q(n-i)$ for all $1 \leq i \leq n$.

We will consider a general case of the query-set size inference control, where the formula for the response to the query $q(C)$ is the following:

$$q(C) = \begin{cases} |q(C)| & if\ \ |X_C| \notin \mathcal{B} \\ \# & otherwise \end{cases}$$

where $\mathcal{B}$ in an arbitrary subset of $\langle 0, n\rangle$ called the *set of restricted ranges.*

The *usability* U of a statistical database is a function of f_Q and $\mathcal{B}$ and is defined as follows:

$$U(f_Q, \mathcal{B}) = \frac{\sum_{j \notin \mathcal{B}} f_Q(j)}{\sum_{j=0}^{n} f_Q(j)}$$

It is clear that $U(f_Q, \mathcal{B})$ gives the fraction of answerable queries from the set of queries Q. In other words it measures the richness of the information revealed to the users.

We do not make any assumptions on the set of queries Q (for example, it can be the set of all statistical queries contained in some applications, against which we build some inference control mechanism). Because of this we will write f instead of f_Q understanding, that the set of queries is fixed (in fact we need not know anything about this distribution of values for query-sets and we can assume, for example, the normal distribution).

Now we discuss a measure of security in a statistical database. It is clear, that (in general) it is not possible to provide an absolute security, however, the greatest danger comes from the existence of (multi)trackers (see [Denning et al., 1979], [Denning & Schlörer, 1980], [Michalewicz & Yeo, 1987], [Michalewicz & Chen, 1988]). Here, by a tracker, we understand any formula T which can be used to find the value for any unanswerable query $q(C)$. Because of our experience with general trackers, double trackers, and multitrackers, we can provide a more detailed description: a tracker is a formula T such that for any unanswerable query $q(C)$ at least one of the queries $q(C \cdot T)$, $q(\sim C \cdot T)$ is answerable, and at least one of the queries $q(C + T)$, $q(\sim C + T)$ is answerable.

Note also that not all of the restricted ranges are equally important. Obviously, the first and the last range should be protected by all means; the protection of all other ranges is not so essential. Indeed, our major reason in introducing multiranges (see [Michalewicz & Yeo, 1987]) was prevention of trackers only, which threaten security in implicit ways. On the other hand, finding a response for unanswerable query in the first or the last restricted range usually compromises security explicitly.

These observations have two consequences:

1. the set of restricted ranges $\mathcal{B}$ should always include the following sets: $B_1 = \{0, 1, \ldots, s\}$ and $B_2 = \{n - s, n - s + 1, \ldots, n\}$ for some (possibly small) parameter s,
2. our modified definition of a tracker is the following:

   A *tracker* is a formula T such that for any unanswerable query $q(C)$, where $|X_C| \in B_1 \cup B_m$, at least one of the queries $q(C \cdot T)$, $q(\sim C \cdot T)$ in answerable, and at least one of the queries $q(C + T)$, $q(\sim C + T)$ is answerable.

Now we will formulate a necessary condition for the existence of a tracker. Later we assume that a statistical database is secure, if the necessary condition for the existence of the tracker is not satisfied. Note again that this requirement will not provide a database an absolute security, but it will prevent users from finding a (multi)tracker, which is the most serious threat to security of a statistical database.

It is clear that we can uniquely divide the set of restricted ranges $\mathcal{B}$ into some number (say, m) of disjoint ranges $I_i = \langle x_i, y_i\rangle$, such that

1. $a \in \mathcal{B}$ if and only if $(\exists j)\ a \in \langle x_j, y_j\rangle$,
2. $y_i \neq x_{i+1}$ for $i = 1, 2, \ldots, m - 1$.

It is easily seen that $B_1 \subseteq I_1$ and $B_2 \subseteq I_m$ and $x_1 = 0$, $y_m = n$.

Let us denote by $|I_i|$ the length of the *i-th* restricted range given by $\langle x_i, y_i \rangle$. Let $\mathcal{M}$ be the maximum length of the first and the last restricted ranges, *i.e.* $\mathcal{M} = \max\{|I_1|, |I_m|\}$, $(\mathcal{M} \geq s)$, and $\mathcal{G}$ be the maximum length of all gaps between restricted ranges, *i.e.* $\mathcal{G} = \max_{1 \leq i \leq m-1}\{x_{i+1} - y_i\}$.

It is quite easy to demonstrate that the necessary condition for the existence of a tracker is $2\mathcal{M} \leq \mathcal{G}$. Indeed, it is clear that for any unanswerable query $q(C)$ such that $|X_C| \in I_1$, the queries $q(\sim C \cdot T)$ and $q(C + T)$ are answerable, provided that the tracker query $q(T)$ is answerable ($q(T)$ lies between ranges I_i and I_{i+1}), and the distance between y_i and $|X_T|$ is not less than $\mathcal{M}$, and the distance between $|X_T|$ and x_{i+1} is not less than $\mathcal{M}$.

The negation of the above necessary condition would provide some reasonable level of security. Thus (to provide some level of security to a statistical database) we impose an additional constraint:

$$\mathcal{G} < 2\mathcal{M}$$

A statistical database is called *secure*, if the above condition is satisfied.

Let us introduce some additional notation. Let

$$F(x) = \frac{\sum_{i \leq x} f(i)}{\sum_{i=0}^{n} f(i)}.$$

$F(x)$ is a cumulative distribution function (in statistical sense). Now we can express the usability $U(f, I)$ of a statistical database in terms of the function$F(x)$:

$$U(f, I) = 1 - \sum_{i=1}^{m}[F(y_i) - F(x_i)]$$

In the paper we attept to maximize usability of a statistical database while $\mathcal{G} < 2\mathcal{M}$.

This short paper is organized as follows: Section 2 gives a general description of genetic algorithms and Section 3 describes our implementation and presents the results. The conclusions are presented in the last Section.

2 Genetic Algorithms

Genetic algorithms ([Davis, 1987], [Holland, 1975]) are a class of probabilistic algorithms which begin with a population of randomly generated candidates and "evolve" towards a solution by applying "genetic" operators, modelled on genetic processes occurring in nature.

For a given optimization problem, at each iteration t of a genetic algorithm we will maintain a population of solutions $P(t) = \{x_1^t, \ldots, x_n^t\}$, where x_i^t is a feasible solution, t is an iteration number and n is arbitrarily chosen length of the population. This population would undergo "natural" evolution. In each generation relatively "good" solutions will reproduce; the relatively "bad" solutions will die out, and will be replaced by the offsprings of the former ones. To distinguish between the "good" and "bad" solutions we will use $f(x_i^t)$ which will play a role of the environment.

During iteration t, the genetic algorithm maintains a population $P(t)$ of some solutions $x_1^t, \ldots, x_n^t$ (the population size n remains fixed for the duration of the computation). Each solution x_i^t is evaluated by computing $f(x_i^t)$, which gives us some measure of "fitness" of the solution (obviously, the lower $f(x_i^t)$, the better). Next, at iteration $t+1$ a new population is formed: we select solutions to reproduce on the basis of their relative fitness, and then the selected solutions are recombined using genetic operators (crossover and mutation) to form a new set of solutions.

The *crossover* combines the features of two parent structures to form two similar offspring. Crossover operates by swapping corresponding segments of a string of parents.

A *mutation* operator arbitrarily alters one or more components of a selected structure—this increases the variability of the population. Each bit position of each vector in the new population undergoes a random change with the probability equal to the mutation rate, which is kept constant throughout the computation process.

The theoretical basis of a genetic algorithm states that, in a given population, chromosomes (solutions) better suited to the environment (evaluation) will have exponentially greater chance of survival, and, therefore, better chance of producing offsprings [Holland, 1976]. Moreover, this genetic search method is far from being a pure *hill-climbing*, for at any time it provides for both exploitation of the best solutions, and exploration of the search space.

A genetic algorithm to solve a problem must have 5 components:

1. A genetic representation of solutions to the problem;
2. A way to create an initial population of solutions;
3. An evaluation function that plays the role of the environment, rating solutions in terms of their "fitness";
4. Genetic operators that alter the composition of children during reproduction; and
5. Values for the parameters that the genetic algorithm uses (population size, probabilities of applying genetic operators, *etc.*).

We discuss these components for our implementation in the next section.

3 Optimal selection of restricted ranges

In the Introduction we defined usability $U(f, \mathcal{B})$ of a statistical database as a function of the set of restricted ranges $\mathcal{B}$ and the cumulative distribution function F. We will try to maximize usability under the following condition:

$$(\star\star\star) \qquad \mathcal{G} < 2\mathcal{M}$$

to provide some reasonable level of security for a statistical database.

In the full version of the paper we considered four different sets $\mathcal{B}$, *i.e.* four different distributions of ranges (classical case—two ranges, uniform distribution, arithmetical, and geometrical) and we gave a formula for usability of a statistical database in each of these cases. In this paper we discuss the optimal set $\mathcal{B}$ found by a genetic algorithm.

We discuss all components of our genetic algorithm (listed earlier) in turn.

a genetic representation of a solution: We have created a solution vector on n bits, $v[1..n]$. A query $q(C)$ is restricted iff $v[|X_C|] = 1$. It means that the vector v determines the set $\mathcal{B}$ defining all restricted points which cluster into ranges.

As discussed in the Introduction, we assume that every solution vector v has the following property:

$$v[i] = 1 \text{ for } i = 1, \ldots, s \text{ and } i = n - s, \ldots, n.$$

It means, that every solution vector restricts queries $q(C)$ such that $|X_C| \leq s$ or $|X_C| \geq n - s$. The number of different solution vectors is 2^{n-2s}. In our experiments we took $n = 100,000$ (the size of the database) and $s = 5$. In such a case the number of different solution vectors is $2^{99,990}$ which excludes the possibility of an exhaustive search (in fact, the number as "small" as 2^{100} would exclude this possibility also). The task of the genetic algorithm is to find a near-optimal solution vector.

an initial population of solutions: We have created an initial population (of size N) of solution vectors. Each bit in a solution vector was initialized to 0 or 1 accordingly to some probability distribution function. In our experiments we divided the whole populatioin into five disjoint sets of equal cardinalities $G_1, \ldots, G_5$ and every solution vector v from the group G_k was initialized in such a way that $Probability(v[i] = 1) = (k-1) \cdot 0.1$ (for $k = 1, \ldots, 5$).

an evaluation function: Obviously, our evaluation function was based on usability of the statistical database. Note that any solution vector corresponds to the set of restricted ranges $\mathcal{B}$ and the query distribution function f_Q is given. Then easily we can determine the usability of the database as $U(f_Q, \mathcal{B})$ (see Introduction).

However, there is no guarantee that the security condition ($\mathcal{G} < 2\mathcal{M}$) is satisfied. So we introduced a penalty function $penalty(v) \longrightarrow R$ with the following characteristic:

$$penalty(v) = \begin{cases} 0 & if\ \mathcal{G} < 2\mathcal{M} \\ c_0 \cdot (\mathcal{G} - 2\mathcal{M})^2 & otherwise \end{cases}$$

Then, for a given vector v, the evaluation function $Eval$ is defined as:

$$Eval(v) = U(f_Q, \mathcal{B}) - penalty(v)$$

genetic operators: In our implementation we used two genetic operators: crossover and mutation.

The crossover combines the features of two parent structures (solution vectors) to form two similar offspring. Crossover operates by swapping corresponding segments of a string of the parents. For example, if the parents are represented by five-dimensional vector, say $(a_1, b_1, c_1, d_1, e_1)$ and $(a_2, b_2, c_2, d_2, e_2)$, then crossing the vectors between the second and the fifth components would produce the offspring $(a_1, b_1, c_2, d_2, e_2)$ and $(a_2, b_2, c_1, d_1, e_1)$. In our implementation we crossover to solution vectors of N bits each. The crossover rate C controls the frequency with which the crossover operator is applied. In each new population, $C \cdot N$ structures undergo crossover.

A mutation operator arbitrarily alters one or more components of a selected structure—this increases the variability of the population. Each bit position of each vector in the new population undergoes a random change with the probability equal to the mutation rate M. Consequently, approximately $M \cdot N \cdot n$ mutations occur per generation.

values for the parameters: As mentioned earlier, we perform some experiments with fixed size of a database ($n = 100,000$) and a given query distribution function f_Q. Initially, we assumed that

$$F(x) = \frac{1}{\varsigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{(t-\mu)^2}{2\varsigma^2}} dt$$

where $\mu = \frac{n}{2}$, and $\varsigma = \frac{n}{c}$ (c is some constant). The equation $\mu = \frac{n}{2}$ is the consequence of our earlier assumption $f(i) = f(n-i)$, and by $\varsigma = \frac{n}{c}$ (*i.e.* by the constant c) we can model the distribution of values for the query-sets.

Then, the formula for usability of a statistical database is

$$U(I) = 1 - \frac{1}{\frac{n}{c}\sqrt{2\pi}} \sum_{i=1}^{m} \int_{x_i}^{y_i} e^{-\frac{(t-\frac{n}{2})^2}{2\frac{n^2}{c^2}}} dt$$

However, the computational effort to find the precise value of integral for each vector was enormous; this resulted in a change of the query distribution function f:

$$f(x) = \begin{cases} \frac{2(b-a)}{n}x + a & if\ 0 \leq x \leq n/2 \\ \frac{2(a-b)}{n}x + (2b-a) & if\ n/2 < x \leq n. \end{cases}$$

which will change a cumulative distribution function F to

$$F(x) = \begin{cases} \frac{an+(b-a)x}{n}x + a & if\ 0 \leq x \leq n/2 \\ \frac{(3b-a)n+2(a-b)x}{2n}(x-\frac{n}{2}) + \frac{(a+b)n}{4} & if\ n/2 < x \leq n. \end{cases}$$

where (in our implementation) $a = 0.000002$ and $b = 0.000018$.

The other parameters were fixed: the population size $N = 100$, the mutation rate $M = 0.001$ and the crossover rate $C = 0.35$. The constant c_0 for the penalty function was $c_0 = 0.00002$.

The table below gives the best values found of usability function for each case, together with the parameters (like q and r in the case of the arithmetical distribution).

Sort of distribution	Evaluation	Comments
Classical case	0.700000	
Uniform distribution	0.666666	
Arithmetical distribution	0.841647	$q = 4$, $r = 910$
Geometrical distribution	0.844444	$q = 22$, $\sigma = 2$
Genetic algorithm	0.887030	penalty $penalty = 0.00648$

Figure 1: The usability of a database, $n = 100,000$

4 Conclusions

If we compare a traditional approach (two restricted ranges, general trackers) and the modified one (multiranges with uniform or non-uniform distribution) with the genetic algorithm approach, we observe that:

- The inference control mechanism based on genetic algorithm can reveal richer information to the users.
- All methods are equally feasible,

- If the user does not know the exact points of restricted ranges $\mathcal{B}$, it should be relatively harder to construct a tracker,
- methods based on multiranges have a simple fault: the boundaries of a range $\langle x_i, y_i \rangle$ need not be integer—it means that a range of the lenght 1.9 may restrict only one query size. These inaccuracies have influence on the final evaluation of the usefulness (usability) of a method. This sort of inaccuracies was eliminated in genetic algorithm approach.

Note also that the adminsitrator of the system can monitor the real distribution of the pattern of queries and than apply the algorithm to maximize the usability of the database. If the pattern of queries changes, a genetic algorithm (as an adaptive procedure derived from principles of evolution) easily adopts to new requirements—such genetic algorithm can run in the background and update the set $\mathcal{B}$ constantly. Also we can tune our penalty function *penalty* which "describes" the importance of the security condition ($\star\star\star$). Additionally, different applications (queries) may have different "degree of importance" which would influence the usability of the database—such modifications are easy in our model.

It seems that a genetic algorithm approach is much more powerful than all other methods studied in this paper as a method for providing security to a statistical databes based on query-set size inference control.

References

[Davis, 1987] Davis, L., (editor), *Genetic Algorithms and Simulated Annealing*, Pitman, London, 1987.

[Denning et al., 1979] Denning, D.E., Denning, P.J. and Schwartz, M.D., *The Tracker: A Threat to Statistical Database Security*, ACMToDS, Vol.4, No.1, March 1979, pp.76–96.

[Denning & Schlörer, 1980] Denning, D.E. and Schlörer, J., *A Fast Procedure for Finding a Tracker in a Statistical Database*, ACMToDS, Vol.5, No.1, March 1980, pp.88–102.

[Denning & Schlörer, 1983] Denning, D.E. and Schlörer, J., *Inference Controls for Statistical Databases*, Computer, Vol.16, No.7, July 1983, pp.69–85,

[Holland, 1975] Holland, J., *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[Michalewicz, 1981] Michalewicz, Z., *Compromisability of a Statistical Database*, Information Systems, Vol.6, No.4, Dec. 1981, pp.301-304.

[Michalewicz & Yeo, 1987] Michalewicz, Z. and Yeo, A., *Multiranges and Multitrackers in Statistical Databases*, Fundamenta Informaticae, Vol. X, No.4, Dec. 1987, pp.41–48.

[Michalewicz & Chen, 1988] Michalewicz, Z. and Chen, K.–W., *Ranges and Trackers in Statistical Databases*, Proc. of the 4-th International Conference on Statistical and Scientific Databases, Rome, 21–23 June 1988, and Springer-Verlag Lecture Notes in Computer Science, No.339, pp.193–206,

[Michalewicz & Chen, 1989] Michalewicz, Z. and Chen, K.–W., *Usability and Security of Statistical Databases*, VUW Technical Report, 1989.

[Schlörer, 1980] Schlörer, J., *Disclosure from Statistical Databases: Quantitative Aspects of Trackers*, ACMToDS, Vol.5, No.4, Dec. 1980, pp.467–492.

TEMPORAL QUERY OPTIMIZATION IN SCIENTIFIC DATABASES

Himawan Gunadhi and Arie Segev

Walter A. Haas School of Business
The University of California and
Computing Sciences Research and Development Department
Lawrence Berkeley Laboratory
Berkeley, California 94720

Abstract. Many statistical and scientific database applications are inherently time dependent, and can be more efficiently modeled and managed by temporal data models. We investigate issues pertaining to query processing of temporal databases in a relational environment. Tuple-versioning of relations is the adopted method of temporal data representation. New operators are necessary in order to exploit the richer semantics of temporal queries. We define four classes of temporal joins -- the theta, time intersection, time union and event joins. We will focus on the temporal equijoin and evaluate strategies for its implementation within the framework of a relational database system.

1. INTRODUCTION AND MOTIVATION

The importance of temporal data models lies in their ability to capture the complexities of real world phenomena which are inherently dependent on time. Econometrics, time-series analysis, surveys, simulations and experimental measurements are examples of applications that are time dependent. Traditional approaches, such as the relational model of data, are incapable of handling all the nuances of such phenomena. Temporal models open up the possibility for new types of operations to enhance the retrieval power of a database management system (DBMS). For example, aggregates, moving averages and joins along the time dimension can be carried out. One of the potential drawbacks of such models is the lack of processing efficiency -- the size of data and the complexity of time-oriented queries may yield unsatisfactory performance.

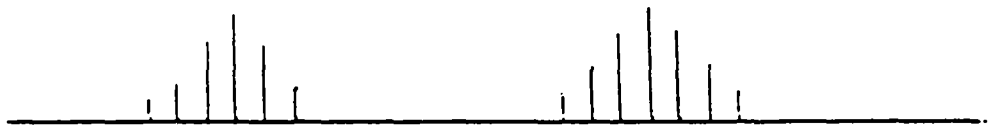
Many papers have been published on logical models that incorporate to varying degrees the time dimension. Most fall into the following categories: (1) Extensions to the relational model, e.g. [Clifford & Tansel 85, Ariav 86, , Clifford & Croker 87, Snodgrass 87]; (2) Enhancements of the Entity-Relationship model, e.g. [Adiba & Quang 86], and (3) Independent modeling such as the concept of the *Time Sequence Collection* (*TSC*) by [Shoshani & Kawagoe 86, Segev & Shoshani 87]. Many operators have been introduced in these papers, although in the relational context, the primary emphasis has been on their integration into the syntax of established query languages, such as SQL and QUEL. This is motivated by the desire to implement a temporal DBMS by minimal modification to current relational technology.

Our approach is to look into the functional requirements of queries on a *temporal relational database*. From there we define a set of fundamental join operators and investigate implementation and optimization strategies. We are motivated in part by the desire to study the feasibility of implementing the *TSC* model in relational form, or on top of an existing relational DBMS. We define four primary types of temporal joins, classified according to the nature of attributes and operators specified in the join predicates. It is our belief that these joins should be capable of capturing the semantics of most, if not all, of the temporal join operators found in the literature. In this paper, we will look at a specific temporal operator, the *temporal equijoin*, and evaluate alternative general strategies for its implementation.

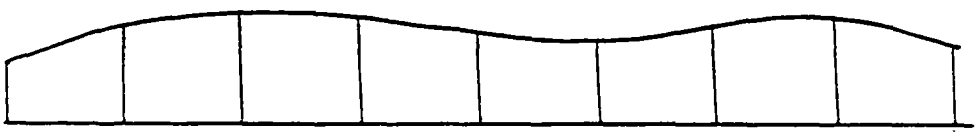
This work was supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy under Contract DE-AC03-76SF00098.

2. MODELING AND REPRESENTATION

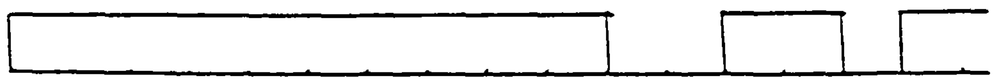
A convenient way to look at temporal data is through the concepts of *Time Sequence* (*TS*) and *Time Sequence Collection* (*TSC*) [Segev & Shoshani 87]. A *TS* represents a history of a temporal attribute(s) associated with a particular instance of an entity or a relationship. The entity or relationship is identified by a *surrogate*. For example, the radiation measurement at a given location is a *TS* with the location ID the surrogate. A *TS* is characterized by several properties, such as the time granularity, lifespan, type, and interpolation rule to derive data values for non-stored time points. Figure 1 illustrates graphically three different time sequences. Figure 1a shows the recorded readings from a detector, which is discrete valued and irregular. By irregular, it is implied that not every data point contains a value for the temporal attribute. Figure 1b shows magnetic field readings, which is a regular and continuous time sequence. The last example pertains to failure data, which is interval based.



(a) Detector data: irregular, discrete



(b) Magnetic field: regular, continuous



(b) Failure data: interval

Figure 1. Examples of Time Sequences

In this paper, for the sake of expositional convenience, we concentrate on two types of data -- *discrete* and *stepwise constant* (*SWC*). *SWC* data represents a state variable whose values are determined by events and remains the same between events; the failure data represents *SWC* data. Discrete data represents an attribute of the event itself, e.g. the detector data. Time sequences of the same surrogate and attribute types can be grouped into a time sequence collection (*TSC*), e.g. radiation measurements for a collection of sites form a *TSC*. There are various ways to represent temporal data in the relational model; detailed discussion can be found in [Segev & Shoshani 88a]. Representations can be different at each level (external, conceptual, physical), but we are concerned with the tuple representation at the physical level. In order to generalize the analysis, we assume *SWC* data using the time-interval representation, as shown in the examples of Figure 2: RADIATION records the radiation levels (on the basis of scale as opposed to actual readouts, which are discrete data) at various laboratory worksites and EMP_LOC keeps track of the location of employees. It should be noted that while the representation in Figure 2 is adequate for the representation of both discrete and *SWC* data, it is not sufficient for continuous data. For such a data type, a function need to be explicitly defined for the assignment of temporal attribute values to a given time interval.

We use the terms *surrogate* (*S*), *temporal-attribute* (*A*), and *time-attribute* (T_S or T_E) when referring to attributes of a temporal relation. For example, in Figure 2, the surrogate of the RADIATION relation is L#, LEVEL is the temporal attribute, and T_S and T_E are time attributes. We assume that all relations are in first temporal normal form (1TNF) [Segev & Shoshani 88a]. 1TNF does not allow a surrogate instance to have more

RADIATION	L#	LEVEL	T_S	T_E
	L1	4	1	20
	L2	1	1	7
	L2	2	8	20
	L3	2	1	7
	L3	5	8	20

EMP_LOC	E#	L#	T_S	T_E
	E1	L3	1	12
	E1	L2	13	20
	E2	L1	1	17
	E2	L2	18	20
	E3	L3	1	20

Figure 2. Examples of Temporal Relations

than one value of a temporal attribute at a given time point. The implication for a temporal relation is that there are no two intersecting time intervals for a given surrogate instance. Whenever it is clear from the context, we will use the term "relation" instead of "temporal relation", although the two are not equivalent.

3. TEMPORAL JOINS

We will define the primary temporal joins, focusing specifically on the equijoin. Details and examples related to the other joins can be found in [Gunadhi & Segev 90a, Segev & Gunadhi 89].

3.1. Basic Definitions

Let $r_i(R_i)$ be a relation on scheme $R_i = \{S_i, A_{i1}, \ldots, A_{im}, T_S, T_E\}$, where S_i is the surrogate of the relation with domain $dom(S_i)$, T_S and T_E are the time-start and time-end attributes respectively, with $dom(T_S) = dom(T_E)$. A_{ij} denotes the attribute with a corresponding domain $dom(A_{ij})$. We distinguish between the surrogate and other non-time attributes for expositional convenience. It is not necessary to distinguish between temporal and non-temporal A_{ij}'s, although one or more should be *time-varying* in order for temporal joins to produce non-trivial results. The characteristics and measures of the time attribute are described in [Segev & Shoshani 87]. It is assumed throughout that we are dealing with a time domain which can be represented as a finite or countably infinite set of integers.

Define $T_i = \{T_S, T_E\}$ as the *time-subscheme* and $R_i' = R_i - T_i$ as the *non-time subscheme* of r_i. Let x_i represent a tuple in r_i, and $x_i(\cdot)$ the projection of x_i on some relational attribute(s). For a given tuple, $[x_i(T_S), x_i(T_E)]$ define a bounded interval, and the time-values immediately preceding and succeeding any of these boundaries are indicated by a decrement or increment of 1 respectively. Define r_1 and r_2 to be *T-compatible* if T_1 and T_2 are defined over compatible domains. Compatibility does not always mean identical domains, but we will assume so in this paper. The *time intersection* operator $x_1(T_1) \cap x_2(T_2)$ (or equivalently, x_1 *intersects* x_2) returns *true* if $x_1(T_S) \le x_2(T_E) \wedge x_1(T_E) \ge x_2(T_S)$, and *null* otherwise, where r_1 and r_2 are *T-compatible*. We shall always assume that any joins on time are always made on *T-compatible* domains. Any *join* between r_1 and r_2 will produce r_3 with scheme $R_3 = R_1' \cup R_2' \cup T_3$, where the derivation of $r_3.T_S$ and $r_3.T_E$ which make up $r_3.T_3$ is dependent on the type of temporal join. Where *null* values are involved, we use $\varnothing$ to indicate the value for a single null attribute, and $\{\varnothing, \ldots, \varnothing\}$ for a set of such attributes.

A *temporal theta-join*, *T*θ*-join*, is made up of the *conjunction* of two sets of predicates, P_T and $P_{R'}$. P_T represents the set of time join predicates, i.e. those defined over time attributes, while $P_{R'}$ represents the set of non-time join predicates. There are three subclasses of temporal joins that are of special interest, based on the specification of join predicates: *time intersection* class, *time union join* and *event-join*. Time intersection type of joins have a time predicate of $r_1.T_1 \cap r_2.T_2$. Where the non-time predicate has an equality operator, the join is called *temporal equijoin*, or *TE-join*, while if it is null, the join is a *time join* or *T-join*. In the event that the predicate is a non-equality type, we group it for processing purposes with the rest of the theta-join class. The semantics of a *TE-join* in the context of -1NF relations is given in [Clifford & Croker 87].

3.2. Temporal Equijoin

In the TE-join, two tuples $x_1 \in r_1$ and $x_2 \in r_2$ qualify for concatenation if the non-time join attributes have the same values and their time intervals intersect. Each concatenated tuple will have time attribute values that define the non-empty intersection of the two joining tuples. Note that the concatenation of tuples is non-standard, since only one pair of T_S and T_E attributes is part of the result tuples. If Y_{ij} are the non-time join attributes, where the subscripts i and j denote the relation number and attribute number respectively, then

r_1 TE-JOIN r_2 on $Y_{11} = Y_{21} \wedge \cdots \wedge Y_{1m} = Y_{2m}$

$= \{x_3 | x_3(R_1') = x_1(R_1') \wedge$

$x_3(R_2') = x_2(R_2') \wedge$

$x_1(T_1) \cap x_2(T_2) \neq \emptyset \wedge$

$x_3(T_S) = max(x_1(T_S), x_2(T_S)) \wedge$

$x_3(T_E) = min(x_1(T_E), x_2(T_E))$

$\}$

Given the following query on the relations of Figure 2: "Find the radiation exposure levels for all employees", we formulate the following join: EMP_LOC TE-JOIN RADIATION on EMP_LOC.L# = RADIATION.L#. The result of the join is shown in Figure 3.

Result	E#	L#	LEVEL	T_S	T_E
	E1	L3	2	1	7
	E1	L3	5	8	12
	E1	L2	2	13	20
	E2	L1	4	1	17
	E2	L2	2	18	20
	E3	L3	2	1	7
	E3	L3	5	8	20

Figure 3. Result of TE-join between EMP_LOC and RADIATION relations

3.3. Time Union Join

The A TU-join is characterized by a union operation on the time intervals. There may be other time predicates specified, and we denote the set of such operators as $\bar{P}_T$. $P_{R'}$ can also be made of any arbitrary predicate. For every pair of tuples x_1 and x_2 that qualify on the other joining predicates, between one and three tuples can be produced, depending on the relationship between the time intervals of the operands. A TU-join is needed if a pair of tuples is considered to satisfy $P_{R'}$ even for cases where $x_1(T_1) \cap x_2(T_2) = \emptyset$. The union join operation is somewhat analogous to a cartesian product operator in the conventional database context. Formally,

r_1 TU-JOIN r_2 on $P_{R'} \wedge \bar{P}_T = r_{31} \cup r_{32} \cup r_{33}$

where

$r_{31} = \{x_{31} | x_{31}(R_1') = x_1(R_1') \wedge$

$x_{31}(R_2') = x_2(R_2') \wedge$

$P_{R'}$ & $\bar{P}_T \wedge$

$x_1(T_1) \cap x_2(T_2) \neq \emptyset \wedge$

$x_{31}(T_S) = max(x_1(T_S), x_2(T_S))$, & $x_{31}(T_E) = min(x_1(T_E), x_2(T_E))$

$\}$

$r_{32} = \{x_{32} | x_{32}(R_i') = x_i(R_i') \wedge$

$x_{32}(R_j') = \{\emptyset, ..., \emptyset\} \wedge$

$P_{R'}$ & $\bar{P}_T \wedge$

$x_i(T_S) < x_j(T_S) \wedge$

$x_{32}(T_S) = x_i(T_S)$ & $x_{32}(T_E) = min(x_i(T_E), x_j(T_S) - 1)$

$i = 1$ or 2; $j = 2$ if $i = 1$ and $j = 1$ if $i = 2$

}

$r_{33} = \{x_{33} | x_{33}(R_i') = x_i(R_i') \wedge$

$x_{33}(R_j') = \{\emptyset, ..., \emptyset\} \wedge$

$P_{R'}$ & $\bar{P}_T \wedge$

$x_i(T_E) > x_j(T_E) \wedge$

$x_{33}(T_S) = max(x_i(T_S), x_j(T_E) + 1)$ & $x_{33}(T_E) = x_i(T_E)$

$i = 1$ or 2; $j = 2$ if $i = 1$ and $j = 1$ if $i = 2$

}

3.4. Event-Join

An event-join groups several temporal attributes of an entity into a single relation. This operation is extremely important because due to normalization, temporal attributes of the same entity are likely to reside in separate relations. When a non-temporal database is normalized, various attributes of a given entity is likely to be grouped as a single relation. If we now define a subset of the attributes to be temporal and they are stored in a single relation, a tuple will be created whenever an event affects just one of those attributes. Consequently, grouping temporal attributes into a single relation should be done if their event points are synchronized. Regardless of the nature of the temporal attributes, however, a physical database design may lead to storing the temporal attributes of a given entity in several relations. The analogy in a conventional database is that the database designer may create 3NF relations, but obviously, the user is allowed to join them and create an unnormalized result. Since many queries require that different temporal attributes be grouped together as a single relation, we have to account for the fact that differences in the behavior of these attributes over time brings up the possibility that *null* values are involved in the join result. Thus the event-join operation combines elements of the temporal equijoin and two asymmetric outerjoins. Let I denote an arbitrary interval $[T_S, T_E]$ over time; for two intervals I_1 and I_2, $I_1 \subseteq I_2$ if $I_1.T_S \geq I_2.T_S$ and $I_1.T_E \leq I_2.T_E$; the cardinality of an interval, $|I|$, is measured as $|T_E - T_S + 1|$. We can now define the operator.

r_1 EVENT-JOIN r_2

$= \{x_3 | x_3(R_1') = x_1(R_1') \wedge$

$x_3(R_2') = x_2(R_2') \wedge$

$x_3(T_3) = x_1(T_1) \cap x_2(T_2)$

for $x_1 \in r_1$ & $x_2 \in r_2$,

$\vee\ x_3(R_i') = x_i(R_i') \wedge$

$x_3(R_j') = \{\emptyset, ..., \emptyset\} \wedge$

$x_3(T_3) = \max\{|I|$, s.t. $I \subseteq x_i(T_i)\} \wedge$

there does not exist x_j such that $x_j(S_j) = x_3(S_1)$ & $x_j(T_j) \cap x_3(T_3)$

for $i = 1, j = 2$ or $i = 2, j = 1$

}

4. IMPLEMENTATION AND OPTIMIZATION OF TE-JOIN

In [Gunadhi & Segev 90b] we provide specific algorithms to process the TE-join, but our objective here is to study the alternatives for implementing the TE-join within the framework of current relational database management systems. As an example, we will use the TE-join previously described in section 3 between the

EMP_LOC and RADIATION relations on L#. Assume the following statistics about the two relations. Relation size in pages, | EMP_LOC | = 2,000 and | RADIATION | = 40, where we assume that each page holds 50 tuples of either relation. Number of unique L# attributes, | EMP_LOC(L#) | and | RADIATION(L#) | both equal to 40. Number of unique E# attributes, | EMP_LOC(E#) | = 5,000. We make the following assumptions: (1) The values of L# is uniformly distributed throughout both relations; (2) Neither relation is sorted or clustered, and join processing is carried out by the nested-loop algorithm with RADIATION as the outer relation; (3) The result relation, RESULT has 120,000 tuples or 2,400 pages; (4) The buffer size in main memory is $BUF = 20$ pages; and (5) No pipelining is used, which means that the temporary results ($TEMP_i$) are written to disk. The cost C_j of step j is measured in the number of disk I/O's.

We consider three approaches to the problem. The first illustrates a naive strategy, which would be the case if a temporal interface were to be built on top of a conventional system. The second strategy employs a standard theta-join operator where the time stamps are treated as ordinary attributes. In this case, a change is needed in the query processor to replace standard concatenation of tuples by its temporal equivalent. The third is an approach specifically designed for the TE-join, and requires a major change to the optimizer.

4.1. Naive Approach

In this strategy, the handling of the time attributes is ignored at the level of the conventional DBMS. Thus a simple equijoin on the non-time joining domains is executed, and the result is retrieved by a special temporal processor which carries out the restrictions over time attributes, creates the new time stamps for qualifying tuples, and projects the final result. In other words, the logical steps carried out are as follows:

Step 1. $TEMP_1 \leftarrow EMP\_LOC\,[L\# = L\#]RADIATION$

Step 2. $TEMP_2 \leftarrow \sigma_{((EMP\_LOC.T_S \le RADIATION.T_E) \wedge (EMP\_LOC.T_E \ge RADIATION.T_S))}(TEMP_1)$

Step 3. $TEMP_3 \leftarrow \prod_{(TEMP_2 - T_{EMP\_LOC} - T_{RADIATION})}(TEMP_2)\ CONCATENATE$

$\{TEMP_3.T_S = max(EMP\_LOC.T_S, RADIATION.T_S),$

$TEMP_3.T_E = min(EMP\_LOC.T_E, RADIATION.T_E)\}$

Step 4. $RESULT \leftarrow \prod_{(NAME,\, L\#,\, T_S,\, T_E)}(TEMP_3)$

We divide the operation into four steps for clarity of exposition. The *CONCATENATE* operator in Step 3 is introduced to allow the appending of attributes not directly created by a join or cross product. Note also that in Step 3 we distinguish between the similarly named time-stamps in the temporary relation by qualifying them on their original relations. The I/O cost is computed in the following manner. For step 1, $C_1 = |RADIATION| + (|RADIATION| \,/\, BUF \times |EMP\_LOC|) + |TEMP_1|$, which represents the cost of nested-loop execution plus the cost of writing the temporary result to disk. $TEMP_1$ is the result of a conventional equijoin, which means that a cross product on the time domains is carried out for qualifying tuples. Given our uniformity assumption, $C_1 = 40 + (2 \times 2{,}000) + (|EMP\_LOC| \times 50) = 104{,}040$. We assume that steps 2 to 4 are executed in a single scan, i.e. $C_{2\text{-}4} = |TEMP_1| + |RESULT| = 102{,}400$. The total cost of this approach is therefore 206,440 disk I/O's.

4.2. Theta-Join Strategy

In this strategy, we convert the intersection predicate on time into a conjunction of inequality predicates on the time attributes, and treat them as "ordinary" predicates. The query is then processed as a conventional theta-join. Since the creation and concatenation of the new time attributes is unique to temporal data, these operations will still be carried out separately by a temporal processor. The strategy is made up of the following steps:

Step 1. $TEMP_1 \leftarrow EMP\_LOC\,[L\# = L\#]RADIATION\ WHERE$

$EMP\_LOC.T_S \le RADIATION.T_E\ \wedge$

$EMP\_LOC.T_E \ge RADIATION.T_S$

Step 2. $TEMP_2 \leftarrow \prod_{(TEMP_1 - T_{EMP\_LOC} - T_{RADIATION})}(TEMP_1)\ CONCATENATE$

$$\{TEMP_2.T_S = max(EMP\_LOC.T_S, RADIATION.T_S),$$
$$TEMP_2.T_E = min(EMP\_LOC.T_E, RADIATION.T_E)\}$$

Step 3. $RESULT \leftarrow \prod_{(NAME, L\#, T_S, T_E)}(TEMP_2)$

Steps 2 and 3 are identical to steps 3 and 4 of the previous strategy. The total cost is the sum of the cost of reading in the two relations by the nested-loop method, the cost of writing $TEMP_1$ and the cost of reading in $TEMP_1$ and writing $RESULT$. Since $TEMP_1$ and $RESULT$ are of the same size, the total cost comes to $4{,}040 + 3 \times 2{,}400 = 11{,}240$. This is considerably lower than the previous method. In this case we were able to transform a temporal operation to an equivalent conventional one (from the point of view of optimization); we are constrained, however, in this approach to the non-temporal nature of a traditional optimizer. Also, some temporal operators cannot be translated into equivalent relational operators, e.g. the event-join operator.

4.3. Directly Implementing TE-JOIN

The TE-join operator can be implemented independently. There are two primary issues: (1) The manner in which comparison between the tuples is carried out and (2) How concatenation of the new time attributes is achieved. The previous approaches required time-stamp comparisons to be evaluated twice, but we can create the new time stamps for the result tuple, i.e. find $T_S^* = max(EMP\_LOC.T_S, RADIATION.T_S)$ and $T_E^* = min(EMP\_LOC.T_E, RADIATION.T_E)$, then concatenate them iff they are satisfied by the predicate $T_S^* \leq T_E^*$. This test substitutes for the intersection predicate on the two relations' time subschemes.

In algebraic terms, we execute the query as follows:

$$\prod_{(R_1' \cup R_2' \cup (T_S^*, T_E^*))} \left[\sigma_{\bigwedge_j r_1 A_{1j} = r_2 A_{2j} \wedge T_S^* \leq T_E^*} \right] ((r_1 \times r_2)\ CONCATENATE\ (T_S^*, T_T^*))$$

The following procedure executes it.

```
For each $x_1 \in r_1$ {
    for each $x_2 \in r_2$ {
        find $T_S^*$ and $T_E^*$
        for $p \in P_{R'} \wedge P_T$
            if not $p$, do the next $x_2$
        else output tuple on scheme $(R_1' \cup R_2 \cup (T_S^*, T_E^*))$
    }
}
```

The total cost is merely the cost of reading in the relations for the nested-loop method and the cost of writing the output. This comes to 6,440 pages, which is cheaper than the cost of the second strategy. Bear in mind that the sizes of the example relations are relatively small, and the savings would be even more significant for joins involving very large relations.

5. FUTURE RESEARCH

We have introduced and defined four classes of temporal joins: Theta, intersection, union and event joins. We believe that these joins can be used for a large number of join-type queries which have been introduced but not formally defined or identified by others. Moreover, we have developed a framework within which we can evaluate techniques that can optimize the execution of queries involving such joins. We show by example that there are inherent differences between using conventional query processors and developing specialized procedures and algebra to solve these queries. We must remember that the time attributes in a tuple-versioning model must always be treated differently than other attributes, although in many algebraic operations, they may be qualified with the same type of predicates as non-time attributes.

Current and future research address the following issues:

- Developing selectivity estimates based on the model presented, and to expand the scope and sophistication of the model itself. There is also a tradeoff between accuracy of estimates, and the expense of maintaining the necessary statistics and deriving the estimates.

- Investigating the optimization of each class of join. For the temporal equijoin, we are looking at algorithms that exploit data ordering and specialized indexing. Further, the event-join operator is likely to be a commonly used operator, and yet it has no equivalence in the "snapshot" database context. Comprehensive tests of the efficiency of alternatives algorithms are necessary.
- Extending the investigation of temporal operators to those involving temporal ordering and aggregation.
- Continuing our study into the design of efficient data structures, in order to improve the data retrieval capability of a temporal DBMS.

REFERENCES

[Adiba & Quang 86] Adiba, M, Quang, N.B., Historical Multi-Media Databases, *Proceedings of the International Conference on Very Large Databases*, Aug. 1986, pp. 63-70.

[Ariav 86] Ariav, G., A Temporally Oriented Data Model, *ACM Transactions on Database Systems*, 11, 4, Dec. 1986, pp. 499-527.

[Clifford & Croker 87] Clifford, J., Croker, A., The Historical Relational Data Model (HRDM) and Algebra Based on Lifespans, *Proceedings of the International Conference on Data Engineering*, Feb. 1987, pp. 528-537.

[Clifford & Tansel 85] Clifford, J., Tansel, A., On an Algebra for Historical Relational Databases: Two Views, *Proceedings of ACM SIGMOD International Conference on Management of Data*, May 1985, pp. 247-265.

[Gunadhi & Segev 90a] Gunadhi, H., Segev, A., A Framework for Query Optimization in Temporal Databases, *Lecture Notes in Computer Science*, vol. 420. Z. Michalewitz (ed.), Springer-Verlag, pp. 131-147, Apr. 1990.

[Gunadhi & Segev 90b] Gunadhi, H., Segev, A., Query Processing Algorithms for Temporal Intersection Joins, Lawrence Berkeley Lab. Technical Report LBL-28587, Feb. 1990.

[Kolovson & Stonebraker 89] Kolovson, C., Stonebraker, M., Indexing Techniques for Historical Databases, *Proceedings of the International Conference on Data Engineering*, Feb. 1989, pp. 127-139,

[Navathe & Ahmed 86] Navathe, S., Ahmed, R., A Temporal Relational Model and a Query Language, UF-CIS Technical Report TR-85-16, Univ of Florida, April 1986.

[Rotem & Segev 87] Rotem, D., Segev, A., Physical Organization of Temporal Data, *Proceedings of the International Conference on Data Engineering*, Feb. 1987, pp. 547-553.

[Segev & Gunadhi 89] Segev, A., Gunadhi, H., Event-Join Optimization in Temporal Relational Databases, *Proceedings of the International Conference on Very Large Databases*, Aug. 1989. pp. 205-215.

[Segev & Shoshani 87] Segev, A., Shoshani, A., Logical Modeling of Temporal Databases, *Proceedings of ACM SIGMOD International Conference on Management of Data*, May 1987, pp. 454-466.

[Segev & Shoshani 88a] Segev, A., and Shoshani, A., The Representation of a Temporal Data Model in the Relational Environment, *Lecture Notes in Computer Science*, Vol 339, M. Rafanelli, J.C. Klensin, and P. Svensson (eds.), Springer-Verlag, 1988, pp. 39-61.

[Segev & Shoshani 88b] Segev, A., Shoshani, A., Functionality of Temporal Data Models and Physical Design Implementations, *IEEE Data Engineering*, 11, 4, Dec. 1988, pp. 38-45.

[Shoshani & Kawagoe 86] Shoshani, A., Kawagoe, K., Temporal Data Management, *Proceedings of the International Conference on Very Large Databases*, Aug. 1986, pp. 79-88.

[Snodgrass 87] Snodgrass, R., The Temporal Query Language TQuel, *ACM Transactions on Database Systems*, Jun. 1987, pp. 247-298.

[Snodgrass & Ahn 85] Snodgrass, R., Ahn, I., A Taxonomy of Time in Databases, *Proceedings of ACM SIGMOD International Conference on Management of Data*, May 1985, pp. 236-246.

[Snodgrass & Ahn 87] Snodgrass, R., Ahn, I., Performance Analysis of Temporal Queries, TempIS Document No. 17, Department of Computer Science, University of North Carolina, August 1987.

A VISUAL INTERFACE FOR STATISTICAL ENTITIES

Maurizio Rafanelli *, *Fabrizio L. Ricci* °
* Istituto di Analisi dei Sistemi ed Informatica, viale Manzoni 30, 00185 Roma, Italy
° Ist. di Studi per la Ricerca e la Docum. Scient., via De Lollis 12, 00185 Roma, Italy

1. Introduction

In this paper a proposal for a visual interface, able to browse into a graphical representation of a statistical database, to select parts of this data base, to query it and to manipulate the statistical entities selected is discussed. This proposal presents two advantages, with respect to the current solutions: a) to give an integrated environment both of browsing and of querying; b) to have two different windows for the intentional and the extensional level of the metadata (linked dynamically), which assure the possibility to carry out browsing not only of "short-sighted" type. The kind of macro-data which can be represented by a complex data structure is called, in this paper, *Statistical Entity* (SE). It consists of :

a) a single *summary attribute* (quantitative data), representing a property of the statistical phenomenon described by the SE; its instances (summary values) are the numeric values contained in the SE;
b) a set of *category attributes* (qualitative data), that is, the variables which describe the summary attribute unequivocally. In such a set classification hierarchies or cross products among category attributes can appear;
c) a set of values which defines a *domain,* for each category attribute. Such a domain is called "statistical entity category attribute domain" (SECAD). Each SECAD is enclosed in the power set of a primitive definition domain (which is the base for eventual different SECADs);
d) a set of *parameters,* such as "summary type", "unit of measure", "source of data", etc., which characterize every SE.

We note that with the term "summary type" we intend the type of summary attribute (for example "average") which is obtained applying an appropriate aggregate function to the disaggregate (micro) data.

Three levels of abstraction are necessary in order to model the statistical entities:

- the first level (the most abstract) shows the set of category attributes used to describe the phenomenon;
- the second level shows the extensional description of the modalities chosen to aggregate the information on the phenomenon; at this level the SECADs of each category attribute are described;

- the third level shows the logical description of the statistical entities, in terms of the statistical-mathematical function which generated the entity itself and in terms of the universe of reference to which the function was applied.

Therefore, with respect to the traditional data bases, there are a further two levels in this approach, because for statistical data bases it is not sufficient to logically describe the attributes which express the point of view from which the phenomenon is represented; in fact, it is necessary to describe the modality used to aggregate the micro-data (the second level) and the calculation function which generated the summary attribute (the third level).

Manipulating SE generally means changing the reference pattern of data, that is changing the descriptive elements of the SE, changing their characteristics. As a result of this process, it is generally required that the corresponding summary values are recomputed.

With regard to the summary type management, in literature two types of approach were described:

1) complete control exerted by the user (it is the one most frequently considered [Klug 82], [Ghosh 86], [Ozsoyoglu 87]);
2) management transparent to the user; different proposals exist for this second approach [Johnson 81], [Ikeda 81], [Su 83], [Fortunato 86].

Starting from these last papers, the authors proposed a functional model, called Mefisto [Rafanelli 90-a], which represents a statistical entity described by an ordered pair *(r, g),* where:

- *r* is a relation, whose attributes are category attributes describing an SE to which the pair refers;
- *g* is a function which maps from the category attributes, describing the macro-data, to the macrodata themselves.

This model has the capacity to manage transparently the summary type of the statistical entities by means of all the operators necessary for the SE management, without having to define the calculation procedures for each SE.

Let us now take a look at the Mefisto operators:

- *Summarization.* One of the most typical operations which can be carried out in the manipulation of SEs is the summarization of the summary data with respect to a category attribute; in practice this operator eliminates a category attribute. The summary attribute values are calculated in a way which depends on the summary type of the SE.

- *Disaggregation.* If we imagine that the SE values are distributed along a further category attribute which was not contemplated amongst the SE category attributes according to a distribution law represented by another table, the disaggregation operator generates a table whose category attributes are the union of the category attributes of the two input SEs.

- *Classification.* This operator carries out a grouping (or a partition) of the values of the category attributes of the SE according to a correspondence relation, by aggregating the relative values of the summary attribute according to the summary type of the SE.

- *Extension.* This operator enlarges the set of category attributes with another whose modality is a singleton.

- *Restriction.* This operator gives as a result a sub-table of the initial SE. In the case of percent summary type in the initial SE, the data contained in the resulting SE are calculated in function of the values of the initial SE.

- *Enlargement.* This operator can be considered as the inverse operator of the previous one. Given two SEs with the same category attributes, which differ in at least one domain, this operator gives a unique SE in output, as the 'fusion' of the input SEs. In the case of percent summary type in the initial SE.

- *Comparison.* This operator carries out a binary operation that compares each value of the summary attribute of an SE with all the values of the summary attribute of another SE and provides in output a relation obtained by the concatenation of the category attribute values which satisfy the comparison.

2. The graphical representation model of the visual interface

The visual model proposed in this paper uses both an extension of the Grass model [Rafanelli 83], called Grass*, and a set of operators for macro-data defined in the Mefisto functional model. In Grass* the following nodes were therefore defined:

S-node. This node semantically groups a number of SEs or of other S-nodes regarding the same phenomenon or the same part of reality described.

T-node. This node represents the quantitative part (summary value) of an SE. The relative summary type is associated with this T-node, like the parameters, defining further information (some of which are always present) which are linked to its name (which defines the universe of observation of the phenomenon described in the SE). These parameters are:

a) the *summary type,* which depends on the aggregation function which has been applied to the disaggregate data to obtain the summary data;

b) the *unit of measure,* which obviously refers to the summary data;

c) the *information source,* which determines the 'quality' (reliability, etc.) of the information;

d) other eventual *parameters,* that is all the "virtual" category attributes, that is the category attributes which appear only in the name of the SE (for example, in the SE whose name is "Population_in_the_USA_by_year_and_sex ", the virtual category attribute is "country = USA").

X-node. This node has been added to represent complex SEs, that is SEs where there is more than one unit of measure and/or more than one summary type and/or which has been obtained by means of a "macro-union" operation [Fortunato 87] between two or more T nodes which describe different universes or phenomena, etc.

R-node. This node has however been added to represent both the relation models stored in the statistical database definition phase, and the relation models obtained during the manipulation of statistical entities (for example with the previously mentioned 'comparison' operator).

A-node. This node represents the aggregation of all the category attributes which describe the statistical entity. It represents the cross product between all the instances of definition domain of the above-mentioned category attributes and is organized at various levels of aggregation.

C-node. This node represents a single category attribute which describes an SE. Two or more of them can be organized in different levels into a hierarchical classification (for example, "State-County-City"). A category attribute domain is implicitly connected to each C node.

3. The rules of connection of the graphical model

The rules of connection between the previously defined nodes are the following:

Rule 1 - All the S nodes are joined together forming a direct acyclic graph (DAG).

Rule 2 - The T nodes, as well as the X nodes, are all joined to at least one S node, which is a leaf of the DAG; moreover they are always the root of a tree, which consists of a T node, an A node and a set of C nodes (and sometimes other A nodes). This tree represents graphically an SE.

Rule 3 - Under a T node or an X node there is always an A node, unless the T node (or X node) represents a *scalar* or a *vector* (see Fig.1); (for a statistician the term 'scalar' means a statistical entity whose summary attribute is made up of only one number which summarizes the phenomenon under observation).

Rule 4 - An R node is always linked to one or two A nodes with incoming edges while, as it does not have outcoming edges, it always results to be suspended upwards. The edge which joins an A node to an R node is clearly a mapping; in fact the relation represented by an R node is a subset of (at the limit all) the Cartesian product which the A node.

Rule 5 - If an R node is used for subsequent operations of generation of statistical entities, the following two important observations must be made:

- the T node generated by an R node automatically leads to the generation of another A node, linked to the A nodes which generated the R node; the latter is kept (but remains suspended, according to its semantic meaning).
- the statistical entity generated will, in all probability, have null values amongst its summary values; this means that the user must specify, for each one of these, the semantic meaning, which in this visual model, as in those normally used by statistical users, is reduced to just two types: 'not available' (symbol -) and 'impossible' (symbol 0, often called "structural zero").

Rule 6 - A C node can be linked:

a) to another C node, thus forming a classification hierarchy;

b) to an A node, contributing to the Cartesian product which the latter realizes;

c) to a T (or X node) directly, in the previously mentioned case of vector representation.

An example of graphical representation of a SE is shown in Figure 1.

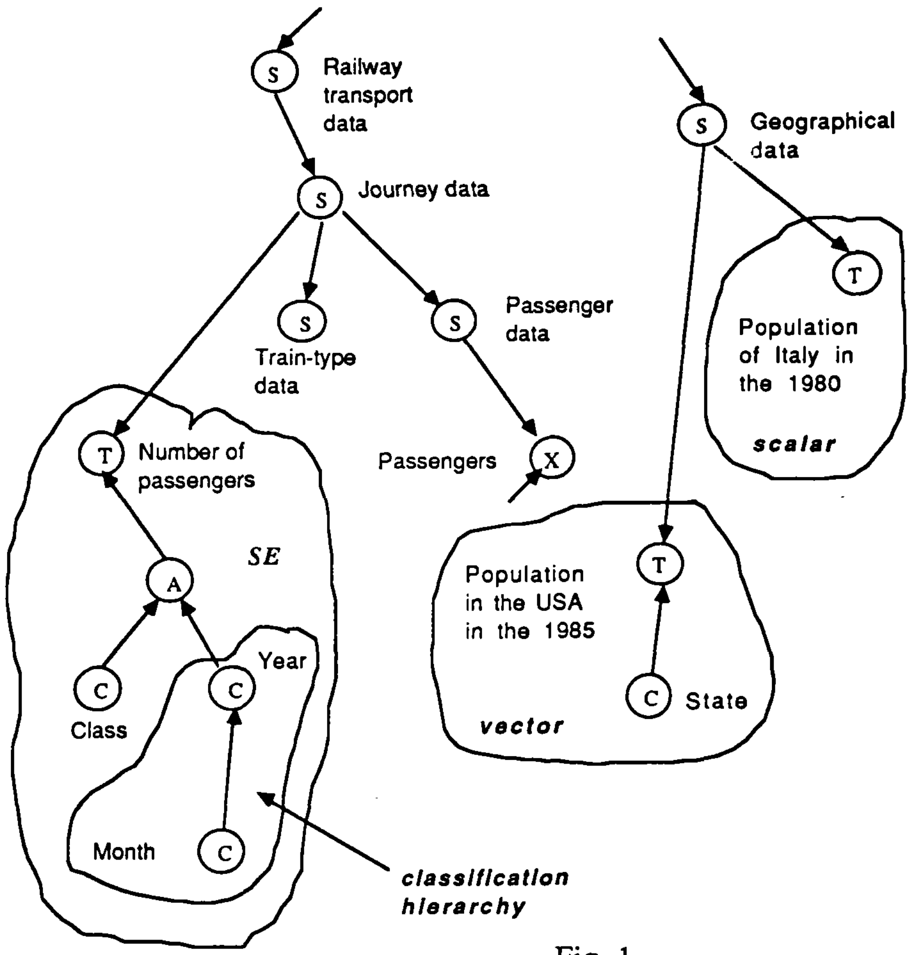


Fig. 1

4. The browser of the visual interface

The aim of the browser is to help the statistical users both to know the three metadata levels, by navigating in the Grass* graph, and to build the statistical entities starting from the SEs stored and using the algebra operators defined in Mefisto.

The primitives to browse are of two types:

a) to establish a starting point;

b) to change the point of view in the adjacency structure.

The former is characterized (by the user) by means of queries regarding the type of node, the information associated to it, the existence of links between them and/or carrying out a search on the words enclosed in the text of the remarks which refer to that node. The latter is characterized by different ways to browse:

- to pass from a node to another node (adjacent to the former), which verifies, if specified in the query, defined conditions;
- to jump from a node to another node, specifying the conditions to individualize it

- to select entire sub-graphs (which satisfy given conditions).

These commands are realized both by using an iconic language, and an equivalent query language. The query language defined for querying the SE database is based on the operators of Mefisto. In browsing in a GRASS* graph, we can highlight and therefore isolate part of the graph. The way in which this happens enables new SEs to be deduced according to the previously defined operators. There is therefore a correspondence between the operators defined in Mefisto and the rules of manipulation of SE.

The main rules for the browsing into a Grass* graph and for the deduction of new graphs and, therefore, new SEs, are the following:

Rule 1 - selecting an S node means reproducing the entire sub-graph of which it is the parent.

Rule 2 - selecting a T or an X node means reproducing the entire SE which the node represents.

Rule 3 - selecting (or cutting) a number of incoming edges in an A node means that a summarization operation is to be carried out.

Rule 4 - adding a C node, with only one modality, to an A node means that an extension operation is to be carried out.

Rule 5 - adding a C node, with more than one modality, to the ones entering an A node means that a disaggregation operation is to be carried out.

Rule 6 - cutting under an intermediate C node means doing a classification operation.

Rule 7 - joining into one C node, two different C nodes relative to two different T nodes, means that an enlargement operation is to be carried out.

Rule 8 - replacing an A (or C) node with another A (or C) node, specifying the relation along which to carry out the substitution, means that a classification operation is to be carried out.

Rule 9 - selecting a modality means that a restriction operation is to be carried out.

Rule 10 - choosing different classifications relative to the variables of the same T node means generating an X node, by applying the classification operation various times.

Rule 11 - substituting the modality of a C node with other C nodes means that a disaggregation / reclassification operation is to be carried out.

In Figure 2 an example of application of these rules is shown.

The user also has the possibility, by direct manipulation, to enrich the information present (for example, to add remarks or synonyms, etc.), to change the classifications (or the classification hierarchies), to add new SEs (these last activities are admitted only "at local level", that is for that user view), etc. It is also possible to memorize the current section before closing it.

5. The implementation of the visual interface

We give now a brief description of the visual interface [Rafanelli 90-b], showing the prototype version, in which only the windows regarding the *Grass** graphical representation, the *instances*, the *information*, the *print format* and the *Staquel* query language have been implemented.

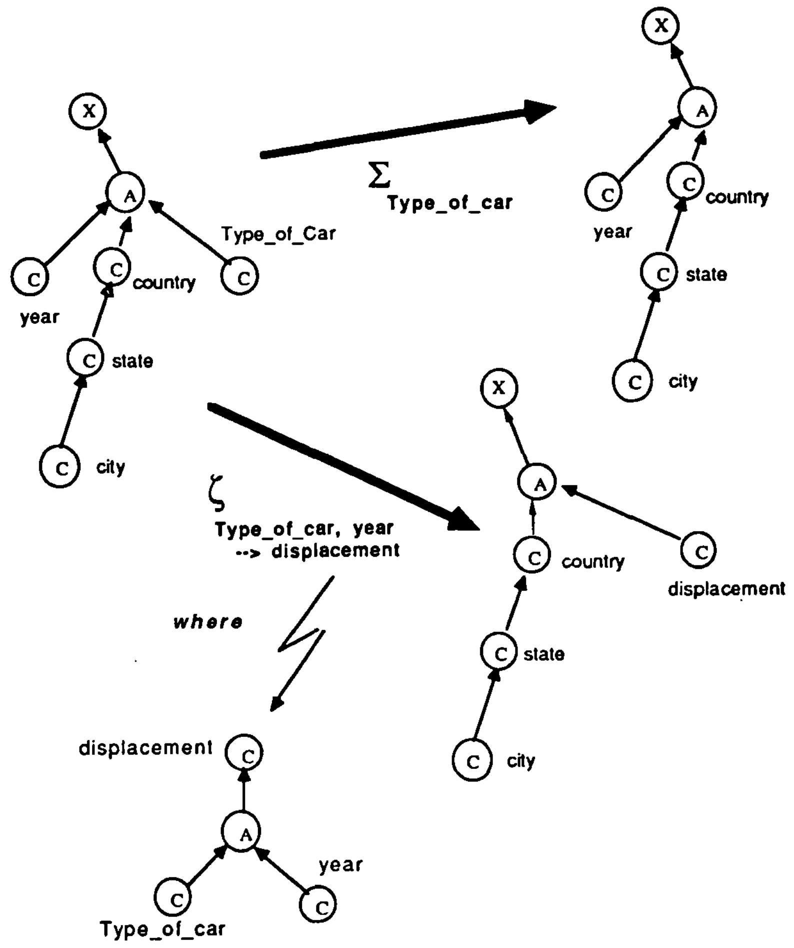


Fig. 2

- *Grass* window.* In this window the intentional form of the Grass*model is graphically represented; in it the various logical representation levels of the statistical database (node levels S, T, X, R, A, and C) are represented. In this window it is possible to browse in (and to manipulate the SEs of) the graph.

- *Information window.* Here it is possible to ask for information regarding the parameters which characterize each statistical entity, such as, for example, the summary type, the unit of measure, the information source etc.

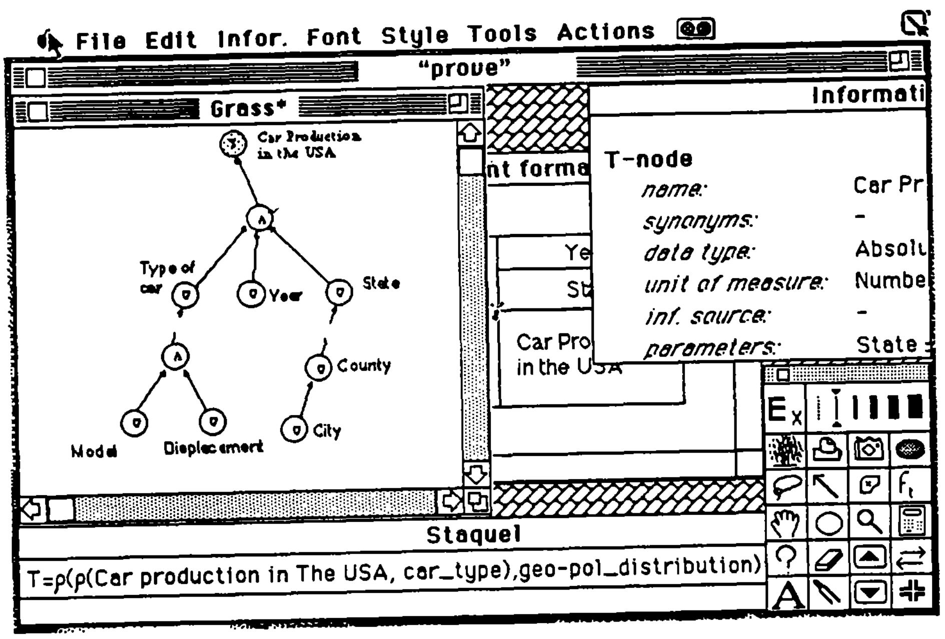
- *Instance window.* In this window all the SECAD values regarding each category attribute (relative to each statistical entity) are shown, as well as the relations which eventually link two or more category attributes, which are written extentionally. This window is, obviously, strongly linked to the Grass* window.

- *Print format window.* In this window it is possible to define how to print the statistical entity (that is, which category attributes to put as rows and which as columns, the order along the two dimensions, etc.).

- *Print window.* Here the system displayed the statistical entity according to the definition of the print format window; the difference from the previous window is that in this window it is possible to see explicitly how both the descriptive (category) attributes and the summary attribute will appear on the paper, or, in general, in output.

- *Staquel window.* In this window the queries, set by means of the Staquel query language, the command of manipulation or the instructions of the data definition language in the database planning phase (planned in this interface), are reported in full.

In this interface other windows are planned. In the screen there is an "icon menu bar" displayed at the top and a palette, whose symbols are icons, which represent an operation for querying, for browsing, for enlarging the schema, for inserting new SEs, etc. In figure 3 en example of how this visual interface presents itself to the user is shown; in it the two cuts (interrupted lines) under the two C nodes "Type of car" and "State" in the Grass* window is the result of a direct manipulation of a user in order to generate a new SE (and expressed by the formula written in the Staquel window); the two cuts are equivalent to two applications of the "classification" operation, derived from the rule 6 above-mentioned.



T-node
name: Car Pr
synonyms: -
data type: Absolu
unit of measure: Numbe
inf. source: -
parameters: State

```
T=ρ(ρ(Car production in The USA, car_type),geo-pol_distribution)
```

Fig. 3

6. Conclusions

The visual interface proposed can be easily used by statistical users. In order to better understand this interface from a graphical point of view, we have briefly presented and discussed the data structure (SE), a functional model with the relative operators and the graphical models for the representation of statistical data.

The commands for the manipulation of the statistical entities which have been realized in this interface are based on these operators.

By means of this interface it is possible to browse in the database, to select topic-classified statistical entities and to manipulate these entities by carrying out queries based on an extension of the Staquel query language.
The interface has been realized by the definition of windows for graphic visualization of the statistical entities, for data analysis options, information and functions (planned so far), for the print format and for the expressions of "data definition language" and "browsing query language".
Our attention has focused on the study of macro-SDBs; in particular we are studying the problems connected with the manipulation of macro-data in order to obtain other data, but without defining new statistical indicators.

Bibliography

[Fortunato 86] E.Fortunato, M.Rafanelli, F.L.Ricci, A.Sebastio, "An algebra for statistical data" Proc. of the 3rd Int. Workshop on Statistical and Scientific Database Management, 1986.

[Ghosh 86] S.P.Ghosh, "Statistical relational tables for statistical database management" IEEE Trans. on Software Engineering, Vol.SE-12, N.12, Dec.1986

[Johnson 81] R.Johnson, "Modelling summary data" Proceed. ACM SIGMOD 1981

[Ikeda 81] H.Ikeda, Y.Kobayashi, "Additional facilities of a conventional DBMS to support interactive statistical analysis" Proceed. 1st InterN Work. on Statistical Database Management, 1981

[Klug 82] A.Klug, "Equivalence of relational algebra and relational calculus query languages having aggregate functions" Journ. ACM N.29-3, 1982

[Ozsoyoglu 87] G.Ozsoyoglu, Z.M.Ozsoyoglu, V. Matos, "Extending relational algebra and relational calculus with set-valued attributes and aggregate functions", ACM trans. Database Systems, 12, 4, 1987.

[Rafanelli 83] M.Rafanelli, F.L.Ricci, "Proposal of a logical model for statistical data base", Proceed. of the 2nd Internat. Workshop on Statistical Database Management, 1983.

[Rafanelli 85] M.Rafanelli, F.L.Ricci "Staquel: a query language for statistical macro-database management systems" Proceed. CIL'85, 1985.

[Rafanelli 90-a] M.Rafanelli, F.L.Ricci "Mefisto: a functional model for statistical entities" Tecn. Rep. IASI, 1990 (in press)

[Rafanelli 90-b] M.Rafanelli, F.L.Ricci "A visual interface for browsing and manipulating statistical entities" Proceed. of the 5th Internat. Conference on Statistical and Scientific Database Management, Lecture Notes in Computer Science, Z.Michalewicz Ed., Vol. 420, Springer Verlag Pub. 1990.

[Su 83] S.Y.W.Su, "SAM*: a semantic association model for corporate and scientific-statistical databases", Information Sciences, Vol.29, N.2 and 3, May and June 1983.

A Scientific DBMS for Programmable Logic Controllers†

Gultekin Ozsoyoglu, Wen-Chi Hou and Adegbemiga Ola

ABSTRACT

We identify database issues associated with programmable logic controllers (PLC), special-purpose computers used in scientific applications and experiments. We propose as a PLC scientific database system a single-user, real-time, scalable main-memory-only relational database system with a two-level architecture having historical data modeling and manipulation capabilities, and query processing techniques incorporating time- and/or error-constrained query evaluation. We revise the ladder logic language, the most common PLC language, to incorporate data manipulation language instructions. We add a separate time component into the PLC processor scan time to handle database updates, backup, integrity enforcement and data archival issues.

1. Introduction

A programmable logic controller (PLC) is a special-purpose computer used within real-time scientific computing systems. In scientific applications and experiments, PLCs are used for signal data gathering and preliminary data processing. Thus, for some scientific applications, a PLC database may also serve as a local/transient part of a larger scientific database.

A PLC automatically controls the operation of a scientific application by running a user-written "application program"#. The input to the application program consists of the status of the application, transmitted to the main memory of the PLC at fixed intervals by autonomous I/O processors and stored into the input buffer#. At its steady state, the application program is continuously scanning its input, solving some boolean equations and setting its output. The functionality of a PLC directly effects the overall flexibility of the scientific application or experiment.

With the rapid advances in computer hardware and falling memory prices, in recent years, the capabilities of new PLCs in the marketplace have been increasing dramatically. We propose the architecture in Figure 1 as the architecture of an environment where real-time data gathering (from multiple sensors) and real-time data manipulation takes place. We now list the advantages of having a database system directly inside a PLC.

(1) **Data Modeling Techniques :** The input and output buffers represent a rather unorganized transient model of the real world, and hence, can be modeled better using the traditional *data modeling techniques* of databases.

(2) **Data Manipulation Languages :** The fields in the input buffer are monitored by the user's application program. The most common language used for application programs in PLCs is called the *ladder logic language*, which is a graphics-oriented, low-level programming language (similar to assembler) equipped with special relay logic, timer and counter instructions. The input and output buffer manipulations can be specified much more precisely and in a compact manner if a *database manipulation language* (DML) is

† This research is supported by the National Science Foundation under Grants DCR-860554, IRI-8811057, IRI-9009897, and IRI-9008632. Address of G. Ozsoyoglu : Department of Computer Engineering and Science, Case Western Reserve University, OH 44106. Address of W-C. Hou : Department of Computer Science, Southern Illinois University at Carbondale, Carbondale, IL, 62901. Address of A. Ola : Department of Computer Science, North Carolina State University, Raleigh, NC, 27695.

Our terminology. The PLC community term for the application program, input and output buffers are control program, and input and output image tables, respectively.

used.

(3) **Historical Databases** : PLCs routinely deal with different versions of data over time. Therefore, *historical data modeling techniques* as well as *historical data manipulation techniques* can replace the ad hoc ways of manipulating historical data in PLCs.

(4) **User-Friendly Interfaces** : The current PLC software allows a limited display of messages and variable-data information in memory. For example, the *contact histogram function* displays the on/off history of a specific main memory bit. A suitable user-friendly visual data manipulation language (such as a revised version of Query-By-Example [Zloo 77]) based on graphically drawn tables may permit more complicated and useful queries specified in a user-friendly manner.

(5) **Handling Large Volumes of Data** : With the added capabilities of a database and a query language, the PLC may analyze much larger volumes of data.

(6) **Data Reduction and Compaction at the PLC Level** : In some scientific experiments and applications, the data gathered is so large that arguments have been raised for "processing the data on-the-fly" during the execution of an experiment/transaction [SSDB 86]. The result is that the host computer gets overburdened with too much raw data. With additional data analysis capabilities at PLC levels, PLCs can aggregate data and send only a fraction of the data produced at the PLC level. This also reduces the overloading of the communication lines between the host computer and the PLCs which is presently a common problem.

We now discuss the properties of a PLC database.

(a) **Real-Time Database** : The data in the input buffer must be scanned within reasonably short "real-time" intervals ranging from microseconds to seconds. Therefore, responses to queries must be guaranteed to be less than a certain "realtime" time bound, almost always less than 5 to 10 seconds.

(b) **Main Memory Database** : Microseconds/seconds query response restrictions necessitate main-memory-only databases.

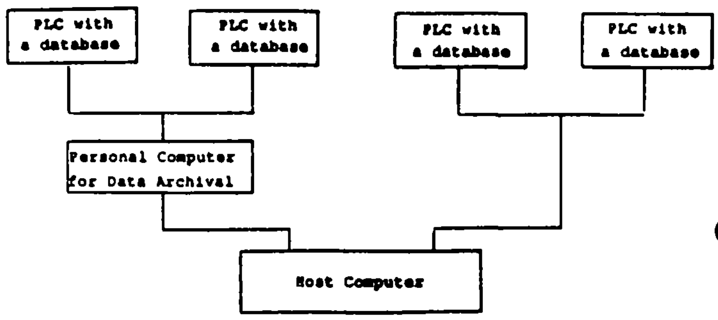


Figure 1. Proposed Architecture for a Scientific Application

(c) **Scalable Database** : Once the environment of a PLC and the requirements of the associated application program are determined, the possible query types to the database stay fixed for a reasonably long period of time. Since the response time of queries is of utmost importance, the DBMS should be scaled so that only the needed routines/functions (e.g., access methods, data structures, etc.) are incorporated.

(d) **Time-Constrained Queries** : The database system must be able to process queries with strict time constraints, i.e., queries of the type "get the output of query q in no more than t time units". For aggregation queries, our approach, discussed in detail in [HoOT 88, HoOT 89, HouO 90a, HouO 90b], is as follows. When a response to a query within the given time units is not possible, we use data samples and construct an *approximate response* to the query. In principle, the actual query response may be a single value (e.g., aggregation queries such as COUNT, SUM, MAX, etc) or it may simply be a set of values. For the latter case, we are currently researching graceful time control and query revision algorithms.

(e) **Error-Constrained Queries** : In some applications, users may specify an acceptable error range for the query response, in addition to (or in place of) a time constraint. An example may be "get the output of a query *aggregate*(E) such that the variance of the relative error in the response is less than *3%*". Clearly, in such an environment, the user is also very much interested in the degree of the error committed due to the approximation. We are currently investigating the techniques for processing error-constrained queries.

In section 2, we discuss the general characteristics PLCs, and briefly present the ladder logic language. Section 3 discusses the features of our design for a database system for PLCs.

2. General Characteristics of PLCs and Ladder Logic Language

In general, the PLC hardware is mostly custom-built with occasional off-the-shelf hardware, and consists of a CPU (or multiple CPUs), main memory, an "industrial" terminal, and high- and medium-speed data communications hardware. Although the CPU has an instruction set similar to those found in CPUs of 16-bit and 32-bit machines, it is especially equipped with fast bit manipulation instructions. The industrial terminal comes with a special keyboard to make the programming of the PLC easier and/or to intervene with the application program.

The PLC software consists of an operating system, high-speed communications software for communicating with I/O processors, medium-speed communications software to the industrial terminal and to other "intelligent" devices.

General-purpose computers and PLCs differ in the programming languages that they use, environmental specifications, and their user types. PLCs are rugged, and work in hostile environments with no special climate controls, tolerating extremes of temperature (60^o C), humidity (95%) and air contamination.

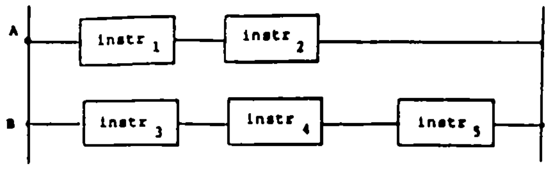
The primary programming language for PLCs is the ladder logic language, which is a visual language that attempts to capture *relay logic schematic*, the industry standard for specifying control functions in scientific applications. In general, the following description of the ladder logic language is common to most implementations.

A *rung* is an ordered set of PLC instructions drawn on a single line. Instructions on a rung are classified as input instructions (those that monitor the input buffer) and output instructions (those that set the output buffer), and are executed from left to right, sequentially (Please see figure 2). A *PLC application program* consists of a main program and a set of subroutines, each of which containing an ordered set of rungs. Rungs in a main program or a subroutine are executed sequentially from top to bottom using the following rules :

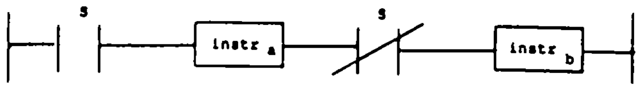
(i) When the "Jump to rung with label A" is encountered on a rung, execution jumps to the rung with label A,

(ii) A *logical switch*# S is a boolean variable with values "*on*" or "*off*" (representing, perhaps, a physical switch or a relay contact). A *logical switch condition* is an atomic formula of the form "$S_1 = on$" or "$S_1 = off$". Each input instruction has the so-called logical switch condition that is evaluated to true or false when the instruction is executed. For example, two instructions of the ladder logic language are "examine logical switch condition $S = on$ ", described graphically by $-|^S|-$, and "examine logical switch condition S = *off*, described by $-\cancel{|^S|}-$ (please see figure 4). During a left-to-right instruction execution on a rung, the output instruction is executed if and only if the logical switch conditions of all the input instruction of the rung are true.

# Logical switch and logical switch condition are our terminology, an attempt to simplify the PLC terminology.



(a) Two Rungs (with Labels A and B) in Ladder Logic Language



(b) A Rung with Four Instructions

Figure 2. Rungs in Ladder Logic Language

Ladder logic instructions include relay-type instructions, timers and counters, data transfer/comparison and arithmetic instructions, bit manipulation, and branching instructions.

The application program has *begin* and *end* statements. *Application program scan* refers to the execution of the application program starting with the begin statement and ending with the end statement. At the end of the application program scan, control is transferred to the PLC operating system which performs an *I/O scan* (please see figure 5). During the I/O scan, the data in the output buffer is copied into the main memory locations for autonomous I/O processors to set I/O devices, and the data in the input buffer is entered from the main memory locations already refreshed by autonomous I/O processors#. There are provisions for sending output to slow (e.g., electromechanical) devices as follows. The programmer of the application program, knowing the time needed to, say, set the slow device, sets directly in his application program the main memory area inspected by the autonomous processor so that the rest of the application scan time and the I/O scan time are sufficient to set the device. At the other extreme, the application program scan time may be long enough for an I/O processor to enter multiple data from a device@. Too much data from a device in a single application scan time, however, indicates a delay on the part of PLC to recognize and act on emergencies, which is not desirable.

To summarize, the application programmer deals with actual (realtime) clock times, and needs to have precise estimates for program scan times and I/O scan times. For time estimations, the PLC manufacturers supply information such as 4 mseconds for 1000 ladder logic instructions, and 1 mseconds for copying 256 words into an input buffer during the I/O scan. In most applications, the processor scan time is kept below 10 seconds. Thus, database manipulation instructions also need to have precise time limits available to (or set by) users.

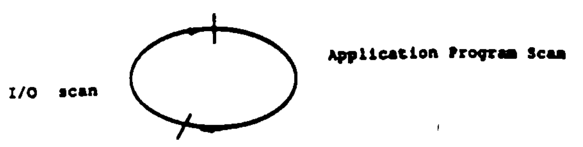


Figure 3. A PLC Processor Scan

After the completion of an I/O scan, a new scan of the application program starts. Thus, a PLC is a nonstop computer, and its processor repetitively executes the application program scan and the I/O scan controlling from hundreds to a few thousand scientific application devices.

3. Features of the Database System

From the discussions above, it is clear that a PLC database is a continuously growing database. At the steady state, to limit the size of the database, historical data that were collected before $(t_{now} - \Delta t)$, where t_{now} is the present time and Δt is a time interval, must be archived and removed from the current database.

When there are no I/O processors (e.g., a simple PLC) the I/O scan simply refers to reading from some input devices and setting other output devices.

@ This case does not exist when there are no I/O processors.

3.1 Architecture

We have designed a two-level, single-user database system architecture. We have omitted from the architecture the external model of the traditional database architecture not because PLCs are not powerful, but because concurrently running application programs using different views create problems in accurately estimating the application program scan times. That is, in a multitasking environment where tasks compete for the resources such as database relations and communication lines, deciding a single top-to bottom execution time of a task in actual time is rather difficult (if at all possible). As far as the hardware computing power is concerned, the present day PLCs are as powerful as personal computers (and, indeed, in some recent products,PLCs are personal computers), and can certainly support concurrent data sharing among the application programs.

3.2 Data Modeling Issues

The traditional data modeling techniques directly apply to PLC databases. There is no reason why, say, the Entity-Relationship Model of the data in the PLC database cannot be designed. All the well-known advantages of data modeling directly carry over to the PLC database environment, and will not be elaborated here.

The PLC environment naturally deals with historical data, e.g., the last reading of a temperature sensor, its value yesterday, etc.. Again, there are various historical data modeling approaches in the literature. For our prototyping effort, we have chosen tuple time stamping with begin-time, end-time values. Perhaps, timestamping tuple components individually may be a better approach in terms of expressiveness and reduced redundancy. However, such an approach produces too many timestamps for individual tuple components, creating a space problem in a main-memory database.

It is important to note that, usually, autonomous I/O processors are not very intelligent devices. Therefore, they simply provide input data from I/O scan to I/O scan. The I/O scan transmits this data into the input buffer. Data in the input buffer must be converted into sets of tuples to be inserted into various relations of the database, and each tuple must be time-stamped by the PLC DBMS software.

3.3. DBMS Issues

There are a number of issues that need to be resolved in a time-constrained, single-user DBMS environment. These are (a) Data Archival, (b) Database Backup, (c) Database Integrity enforcement, and (d) Database Recovery.

Implementing the mechanisms for each of the above issues as independently executing tasks interferes with accurate scan time estimations, and is not an option. Each mechanism must be implemented sequentially during a PLC processor scan. Thus, we create a third component in the PLC processor scan, called the *database-essentials* time. (Please see figure 4).

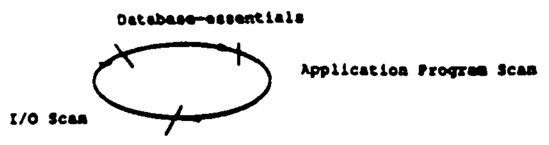


Figure 4. Revised PLC Processor Scan

Archival data is not necessarily the same with the contents of the database. For example, every 1,000th reading of a sensor, or the average, maximum and minimum values in every hour of a sensor device may be archived. Therefore, archived data may first be obtained as a result of some DML manipulations and then can be made part of the current database instance.

Since presently PLCs do not have secondary storage, archived data and database backup copies must be kept in the host computer. There are also arguments in the PLC community for incorporating secondary storage to PLCs for another reason : fast coding, revision and testing of the application program. In either case, the database backup process must be time constrained. It should also be incremental in the sense that only the recent data updates to the database are recorded. Integrity checking and enforcement must also be time-constrained with additional attention to minimize the time needed. Integrity checking involves both the data obtained from the input buffer and the data produced and entered into the database by the application program. Finally for recovery, we do not propose to have any special mechanisms; the most recent backup copy can be used to restart the database.

3.4. Query Language Issues

For our implementation [Liu89], we have chosen the relational model for conceptual modeling. The PLC instructions are augmented with a Data Manipulation Language to allow for the querying of the relations, i.e., the structured data in the main-memory-only database relations. (Please note that the database is different from the input/output buffers). An instruction box in the rung of a ladder logic program can be (a) a relational algebra (RA) query, (b) an SQL query, or (c) a QBE-like query, called Ladder-Logic-by-Example (LLBE) query.

The relational algebra language is used in [HoOT 88, HoOT 89, HouO 90a, HouO 90b] for time-constrained query processing, and , presently, we have the most insight into it in terms of estimators for aggregate queries of RA. SQL is very commonly used in the industry, and may be advantageous because of its common use. LLBE, being a graphics-oriented language, fits in well with the graphical ladder logic language.

An RA expression in an instruction box on a rung is extended by time- and error-control clauses of TIME = *time-limit* and ERROR = (*error type, error-limit*), where *time-limit* specifies a time duration (e.g., 5 microseconds), *error-type* specifies the error type to be measured due to the estimation used (e.g., *relative error* for the variance of the estimator for COUNT(E) query, where E is an RA expression [HouO 88]), and *error-limit* gives an upper bound on the error to be produced for the error-constrained query evaluation (e.g., 10% error in the estimated query output value for the COUNT(E) query). For our prototype development, we have chosen the RA language.

An SQL query can easily be extended by TIME and ERROR clauses, and needs no further explanation. Due to space constraints, we skip the discussion of the LLBE language.

For speed considerations, the compiled query processing approach must be used as opposed to the interpretive query processing approach. Also, the application program is revised rarely, and, therefore, the compiled approach does not create problems.

Since there is one RA expression in a given instruction box on a rung, the time and error clauses directly apply to a single RA query. For the choices of SQL or LLBE as the ladder logic database language, first there is at the compilation time the extra steps of SQL-to-RA or LLBE-to-RA translation, respectively, and minimizations of the produced set of RA expressions. These extra steps are reasonably well investigated, and do not present major problems. An additional issue that needs to be solved is : when an SQL or an LLBE language is used then a given database query in an instruction box may translate into multiple RA expressions. In such a case, policies and mechanisms must be developed for splitting the time quota of the original query given by the user over multiple RA expressions.

The output of a query may be a relation or a single value resulting from applying an aggregate function to a relation. The output relations and values can be referred to by name like other relations or application program variables. Since query results would, in general, be relations, a facility to sequentially scan the tuples of a relation is provided. A pointer or a cursor can be defined to reference the tuples of the relation, in a loop, until a required number of tuples has been read or some logical switch condition is satisfied.

We can associate a pointer PTR with a query output relation, OPEN PTR (to activate the pointer), perform SCAN PTR (to read next tuple) and CLOSE PTR (to deactivate the pointer). PLC arithmetic instructions such as ADD, DIVIDE and COMPUTE can be used to perform aggregation functions during the relation scan. Any pointed tuple component is treated like a "read-only" variable, and may participate in any expression. However, tuples and their components are updated with explicit database primitive update commands (of insert/delete/modify).

Some basic PLC instructions have also been extended to increase their functionality. For example, we have extended the "examine logic switch" instructions, the "examine input closed" and the "examine input open", to test the logical value of a propositional calculus formula, rather than testing a bit value corresponding to the condition of a physical I/O. An "examine F" instruction causes the formula F to be evaluated and the true value is then examined as in the basic examine instructions. In general, the formula F may contain a constant, a variable, a component of a tuple being scanned by the pointer, and *f(E)* where *f* is an aggregate function and E is a relational algebra expression. The functionality of the PLC Timer and Counter instructions have also been enhanced. With the introduction of a time dimension into the database, events and intervals can also be "counted" using database queries.

At times, the user is interested in timing the actual application program scan times and counting events in time scales much larger than the PLC processor scan time. For such cases, there are *retentive* timer and counter instructions that, say, count the PLC processor scans. Therefore, it is important for the user to precisely estimate the evaluation times of database queries. This then necessitates not only an upper bound, but also a lower bound on the time spent for a database query.

4. References

[HoOT88] W-C Hou, G. Ozsoyoglu, B. Taneja, "Statistical Estimators for Relational Algebra Expressions", ACM PODS conference, March 1988.

[HoOT89] W-C Hou, G. Ozsoyoglu, B. Taneja, "Processing Aggregate Queries with Hard Time Constraints", Proc., ACM SIGMOD Conference, May 1989.

[HouO90a] W-C Hou, G. Ozsoyoglu, "Statistical Estimators for Aggregate Relational Algebra Expressions", To appear in ACM TODS, 1990.

[HouO90b] W-C Hou, G. Ozsoyoglu, "Processing Real-Time Aggregate Queries in CASE-DB", May 1990. (submitted for publication).

[Liu89] Y.M. Liu, "A Main-Memory Real-time Database Management System--Implementation and Experiments", M.S. Thesis, CWRU, July 24, 1989.

[SSDB86] Panel on Scientific Databases, Third Int. Workshop on Statistical and Scientific Database Management, 1986.

[Zloo77] M.M. Zloof, "Query-by-Example : A Database Language", IBM Systems Journal, 1977.

Panel: Scientific Data Management for Human Genome Applications

Panelists:
Stan Letovsky (Yale University)
Rob Pecherer (Los Alamos National Laboratory)
Arie Shoshani (Lawrence Berkeley Laboratory)

Genomic data refers to DNA structures in organisms, its stucture and function. Biologists use various techniques to characterize the data, which result in "sequence" structures (DNA, RNA, protein), and "map" structures (called genetic maps, cytogenetic maps, physical maps, etc.) The panelists discussed the specific data management requirements needed to support such data.

The main observation made is that the support for data of type "sequence" is essential. Current commercial relational database technology is based on set theory, and do not support the concept of ordered sequences and operators over such sequences. The only support currently offered is in a form of "blobs" (Binary Large Objects), but the interpretation and manipulation of such objects is left to the application program.

For Human Genome applications there are specific manipulations needed over sequences, such as approximate string matching. This implies that the a scientific data management system that supports the concept of a sequence will need to allow for domain specific operators to be defined and supported. This implies that such a system should be extensible. One of the more promising approaches is the emerging object oriented database technology. The panelists noted that other approaches, such as the extended relational approach, or logic databases are also potentially useful. The belief was that a technology that will combine the benefits of the various approaches will eventually emerge.

The above points were further elaborated by emphasizing the need for integrated tools. Tools development for solving complex problems in DNA structures is difficult because we lack the building blocks which are integrated by a general purpose object paradigm. Specifically, we need user interfaces, programming languages, and persistent databases with basic object capabilities that can be extended (customized) to the particular domain of Genomic data. The most pressing capability is persistent object storage.

Panel Session On Statistical Expert Systems

Panel Members:

- David Hand — Open University, Milton Keynes, UK.
- Gultekin Ozsoyoglu — Case Western Reserve University, Cleveland, Ohio, USA.
- Roger Cubitt (Chairman) — Statistical Office of European Communities (EUROSTAT), Luxembourg (Grand Duchy).

Opening remarks:

- **Roger Cubitt**
  There has been a long standing and active interest in this conference in the development of Expert Systems in the areas of Statistical and Scientific Data Processing and Analysis. A number of papers have been presented over the years, and an excellent summary of the various aspects of this area of interest has been published as part of the report of the third meeting, (then still a workshop), in 1986 in Luxembourg.

  EUROSTAT has launched for 1989 - 1992, a programme of research specifically in the areas of Development of Statistical Expert Systems, "DOSES".

  This programme is organised in a way which is similar to other community research programmes, with a majority of the funding designated for shared cost projects (maximum EUROSTAT contribution, 50%) and a small proportion for fully financed projects. The areas in which proposals have been sought are:

  1. Preparation of a complete system for automated information processing.
  2. Documentation of data and of statistical methods.
  3. Access to statistical information.
  4. Forecasting.

  The proposals received thus far, have resulted in a number of acceptable projects in themes 1 and 4. Themes 2 and 3 have proved much more difficult to find proposals of a nature and content to be accepted for funding.

  Professor Hand who is an active researcher in the field of Statistical Expert Systems and an independent evaluator of the DOSES project, gave his perspective of the European research in this area and some views on his own research.

  Professor Ozsoyoglu is active in the area of statistical and scientific databases and is a regular attendee at these conferences. His remarks hinged on the general orientations of work in this area and its likely results.

- **David Hand**
  Having been active in this field of research for nearly eleven years, I feel it is relevant to give some of the fundamental motivations that were current at the time my research work started. These were principally to help the user of statistics design better experiments and conduct better analysis. The problem of inadequate understanding of the statistical techniques being used has been exacerbated by the increasing availability of software, and hence the case of use—and misuse—of highly sophisticated statistical methods. Interest in the field is still growing; there are many hundreds of research teams worldwide. This is because statistics appears to be well suited to the E.S. Philosophy and researchers in general, need to find statistical expertise which they do not themselves have in the face of a general shortage of statisticians. Progress is now being made, however. Whereas ten years ago, the tendency was to promise "My system will solve all these problems—*when* I have built it", it is now more on the lines of "My system does this—it's not much but it does it successfully". There is now a better understanding of statistical expertise and an appreciation of what might be achieved. A number of themes have become apparent, viz:

  - Statistical strategy.
  - Meta data.
  - Second opinion for "experts".
  - "Consultation" or "knowledge enhancement" systems.

  The last theme, which to some extent arises out of a wish to avoid the term "expert system", is where my own research is currently directed.

  The motivation for this research results principally from the problems inherent in a rule based system. It is ideal for some problem types but by no means for all, as it is fundamentally diagnostic in approach and forces a structure onto the knowledge. This structure is by its nature somewhat inflexible and thus, a whole class of questions a user might wish to answer, are not possible. In addition, the interactions with a rule based system tend to be unsatisfactory in that the client is rarely seeking a highly specific solution. A final problem is a result of the current information explosion and the associated growth in techniques available. Rule based systems tend to have difficulties in locating and accessing information about this growing armoury of techniques. Thus my current orientation is not towards an "expert in a box" type "expert system", but a system to augment the users own knowledge and expertise, i.e. a knowledge enhancement system.

  On an implementation front, I would mention that the "mechanism" or "tool" I am experimenting with in my research, is "hypertext" and a few implementations of this type of system have been tried already (e.g. KENS, NONPARIEL).

- **Gultekin Ozsoyoglu**
  First a disclaimer: I am neither an "Expert Systems" expert nor a Statistics Expert. That said, let us just review the classical paradigm for Expert Systems and see where this leads us for our subject. An "Expert System" is a system which is intended, to some extent or another, to replace an Expert i.e. users who traditionally ask an Expert questions and receive answers, re-direct their questions to an Expert System and get answers. In the

field of statistics, one supposes that the user is some application domain expert (e.g. social scientist, economist, etc.), and the Expert System provided answers some of the questions of a statistical nature. Thus already we appear to have a problem in that on the face of it, we have two Experts, one in a subject matter field, and one (a system) in a statistics field, so the basis for any meaningful question/answer situation becomes less obvious.

There appears to be a requirement for evolutionary learning cycle between the user and the Expert System, as indeed there would probably have been with an Expert. It is not clear that this process can be automated in a Statistical Expert System and certainly not in the general case. The problems of learning, evolving and adapting, appear to be a significant one for Statistical Expert Systems.

There are then other problems that impinge on Statistical Expert Systems, such as interpreting and clarifying the data which is often ambiguous or incomplete. Just the size and complexity of a statistics area can pose a significant challenge for Expert Systems. Think just of sampling or estimation, for example.

Having faced up to all this, we are still a long way from what any implications may be for the Database Management System containing the Statistical or Scientific Data. What the research is that needs to be done to even start to shed light on some of the issues involved, is far from clear.

Points From The Ensuing Discussion:

The role of the Computer Scientist in the general domain of Statistical Expert Systems was questioned. What could he provide and what framework was required in the research environment to ensure that all necessary expertise was available? First questions concerned the nature of the required system; was one seeking a domain specific system with significant capability in a narrow application field, or a more general purpose system? Little evaluation appeared to have been done on this. Behind this was the problem essentially already posed by David Hand; Who was the system intended for? There appeared to be a consensus that the objective was to capture the knowledge of the statistician but to do this, any system required the acceptance by the statistician to succeed. The system needs to be adaptive to facilitate the learning curve of the user and enable the system to be extended to cover new data sources and techniques.

A Summary of the NSF Scientific Database Workshop

James C. French, Anita K. Jones, John L. Pfaltz
Department of Computer Science
University of Virginia
Charlottesville, VA

"... the Earth system science initiative will founder on the rocks of indifference to data access and information management unless an aggressive and supportive new approach is taken — beginning now."[1]

1. Introduction and Background

This quote applies equally well to the space and life sciences. Over the next decade the problems posed by the exponential growth of data in a variety of scientific disciplines will become increasingly pressing. For this reason, an interdisciplinary workshop on scientific database management was organized to look at these problems.

This workshop, conducted at the University of Virginia on March 12-13, 1990, was sponsored by the National Science Foundation. It brought together computer scientists and serious user/proprietors of scientific data collections in several fields of the space, earth, and life sciences. Our objective was to discuss the issues involved in establishing and maintaining large scientific data collections, and to identify opportunities for improving their management and use. More particularly, we sought to assess the current state-of-the-art, assess whether the needs of the sciences are being met, identify the pressing problems in scientific database management, and identify opportunities for improvement. We are still assimilating the results of the workshop and will, in the final report, make recommendations toward improving the usefulness and availability of science data.

Most of the issues arising in connection with scientific databases are similar to those in conventional business environments, but the focus is different. For example, transaction processing and concurrency control issues are more relevant to high volume data processing applications than to DNA sequence analysis, seismic data analysis, or computational astrophysics. Query processing, however, is equally important in each environment.

The relative importance of the issues associated with any data management undertaking is determined by the characteristics of the data and the anticipated operational environment. Much scientific data can be characterized by large volume, low update frequency, and indefinite retention. In fact, it is generally safe to assume that scientific data resulting from experimental observations is never thrown away. The volume of data can be truly staggering. Mapping the three billion nucleotide bases that make up the human genome will result in an enormous volume of data. The *Magellan* planetary probe will generate a trillion bytes of data over its five year life — more image data than all previous planetary probes combined. This suggests that much scientific data will not even be on-line.

A recent article[2] describing the state of NSFNET characterized the problem of access to the net as being hampered by a diversity within the computer world that "verges on anarchy." This same diversity

This workshop was supported by the National Science Foundation under grant number IRI-8917544. A summary of the workshop proceedings was presented at the 5th International Conference on Statistical and Scientific Database Management hosted by the University of North Carolina at Charlotte in April, 1990. The workshop participants have not checked this draft for accuracy and balanced representation.

[1] *Earth System Science: A Closer View*, Report of the Earth System Sciences Committee of the NASA Advisory Council, Jan. 1988.

[2] "Waiting for the National Research Network," *AAAS Observer*, March 3, 1989.

poses an equally substantial barrier to the access of scientific data by those who need it. Indeed, one of the significant problems with scientific databases is largely logistical. According to a recent NASA study of astrophysical data:

> Analyses using multiple data sets from different missions, with support from ground-based observations, are becoming an increasingly important and powerful tool for the modern-day astronomer. However, locating the required observations and accessing and analysing the multiplicity of data sets is not easy at present.[3]

Perhaps the greatest problem facing the scientist is the bewildering array of commercial and custom database interfaces, computer operating systems, and network protocols to be mastered in order to examine potentially relevant data.

From the point of view of the practitioner, there are some relatively simple questions that must be answered in order to enhance the scientific research environment:

What data is available to me?
Where is it located?
How can I get it?

To provide the scientific community with the means to answer these and other questions, database researchers must examine the issues peculiar to scientific database management and the sharing of scientific data.

For these reasons, NSF decided to fund a workshop to examine the issues in more detail with the goal of producing a planning document to guide the foundation as it considered a new research initiative in this area. This two day workshop was held at the University of Virginia in early March. In addition to the computer science representation, the workshop participants were drawn from among the various disciplines of the earth (e.g., oceanography, climatology, geology), life (e.g., microbiology), and space (e.g., astronomy, astrophysics) sciences. The overall representation was approximately 40 percent computer science and 20 percent from each area of the physical sciences. Besides NSF a number of government agencies were represented including Department of Energy (DOE), National Oceanic and Atmospheric Administration (NOAA), National Aeronautics and Space Administration (NASA), National Radio Astronomy Observatory (NRAO), and National Center for Atmospheric Research (NCAR).

The workshop began with invited talks from each represented area with the objective of exposing both common and distinctly different data management problems. The participants were then assigned to one of four panels to examine the relevant issues more closely. Panel representation was proportional across all disciplines. The panel topics were: (1) Multidisciplinary interfaces: standards, metadata, multimedia, etc.; (2) Extensibility; (3) Core Tools: access methods, operators, analysis tools, etc.; and (4) Case Study: Ozone Hole. The case study was used as a vehicle for investigating data management needs, successes and failures in a real mission environment.

2. Dimensions of Scientific Database Systems

The workshop observed that there is a spectrum of database types, which can be characterized in terms of a number of dimensions (which need not be independent). The three we clearly identified (we suspect there may be more) are:

level of interpretation,
intended analysis, and
source.

Many fruitless hours of argument and controversy can be avoided if the protagonists will first identify where the data sets of interest to them lie with respect to these three characterizing dimensions. All too

[3] *Astrophysics Data System Study*, Final Report, NASA, March 1988.

often, major disagreements occur because the participants are implicitly assuming different database types.

2.1. Level of interpretation: A scientific database may be a simple collection of raw data, or real world observations, or it may be a collection of highly processed interpretations. At least two of the panels observed that this dimension manifestly affects what one expects of the data set, and how one employs it. One proposed subdivision of this dimensional axis consists of

raw/sensor data: (seldom saved) raw values obtained directly from the measurement device;

calibrated data: (normally preserved) raw physical values, corrected with calibration operators;

validated data: calibrated data that has been filtered through quality assurance procedures, (most commonly used data for scientific purposes);

derived data: frequently aggregated data, such as gridded or averaged data;

interpreted data: derived data that is related to other data sets, or to the literature of the field.

This sequence of successively greater interpretation need not be precisely correct. But it does indicate that the type of data in a data set can be highly dependent on its level of processing.

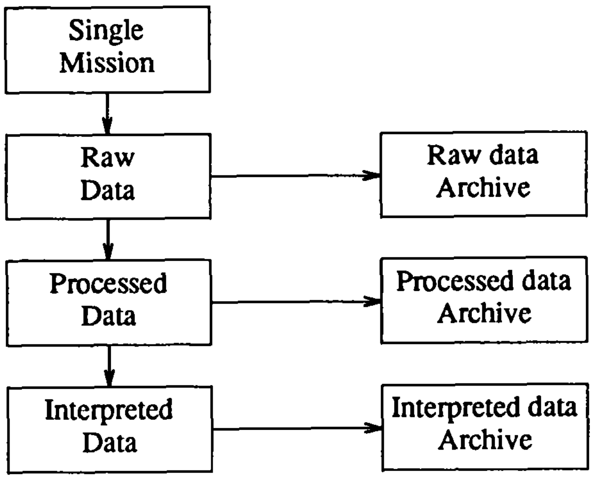
2.2. Intended Scientific Analysis: Our assumption is that all scientific data sets are subject to further analysis, otherwise there is little reason to retain them. The nature of such subsequent analysis frequently determines what data should be retained and whether a particular representational format is adequate. Much earth science data is analyzed statistically, time sequenced, multidimensional tables are common. A predominant activity in biological genome databases is elaborate pattern matching over linear, character data. Multi-spectral analysis in the space sciences apply transformations (e.g. Fourier) to very large two and three dimensional arrays. For each type of analytic processing, a database with different characteristics is most appropriate.

The criticism of the relational data model and its attendant technology is largely because this model is designed for commercial applications, and seems unsuited to any of the analysis domains above.

2.3. Source: This dimension, which is not generally mentioned in the database literature, may be the most fundamental. In Figure 1, we illustrate a familiar *single-source* database environment. Here we envision a single mission, such as the Hubble Space Telescope generating the data that is collected. Either raw or physical data may be retained in its original state in a raw data archive. Commonly, the raw data will be processed, by instrument calibration or by noise filtering, to generate a collection of more usable calibrated or validated data. Finally, this processed data will be interpreted in light of the original goals of the generating mission.

Both the syntactic complexity and the semantic complexity of the interpreted data will be much greater than either of its antecedent data collections. It will have different search and retrieval requirements. Possibly, only it alone will be published.

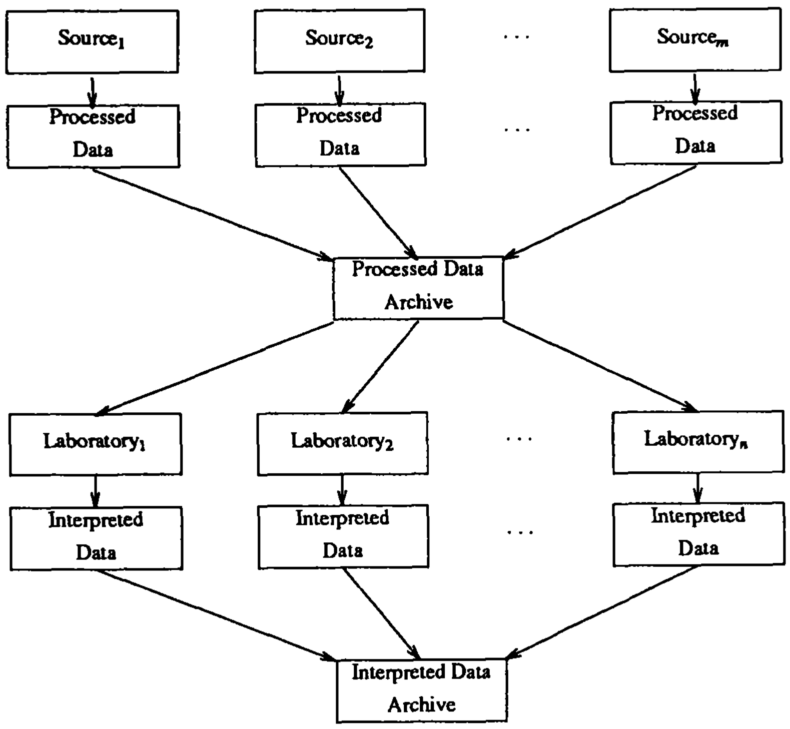
In contrast to such a single-mission/single-source data archive one has data archives that are derived from multiple sources employing multiple data generation protocols. Figure 2 illustrates a typical *multisource* data collection. This structure would characterize the Human Genome project in which several different agencies, with independent funding, missions, and methodologies, generate processed data employing different computing systems and database management techniques. All eventually contribute their data to a common data archive, such as GENBANK, which subsequently becomes the data source for later interpretation by multiple research laboratories that also manage their local data collections independently. In each of the local, multiple, and probably very dynamic, database collections one would



Single-source Data Collections
Figure 1

expect different retrieval and processing needs, as well as different documentation requirements.

This classification of data collection types, however imperfect, helped clarify discussions at the workshop. Readily, the data collections associated with most scientific enquiries will occupy middle



Multi-source Data Collections
Figure 2

positions along these various dimensions. In the following sections summarizing the perceived issues in scientific database management, the urgency and importance of a particular problem is frequently dependent on one's position in the multidimensional space of all scientific database applications.

3. Problems

All sciences have major data management problems, for example: handling volume increases; metadata management; integration of database facilities with applications; finding data; access policy; ease of use; and consistent long term funding. With the exception of long term funding, different sciences seem to have different problems. Technical data management techniques are often domain specific. In the following sections we have subdivided the problems raised at the workshop into two large categories, main issues and lesser issues, and described them more fully within each category. This subdivision has been imposed to indicate a sense of relative importance to the reader without attempting a fruitless exercise of exactly ranking the problems.

3.1. Main Issues in Scientific Database Management

The issues and problems discussed in this section received most of the attention of the participants. For this reason they were deemed to be most worthy of immediate attention.

3.1.1. Metadata: The data within scientific databases covers a wide spectrum of classes: *raw data* — values measured by a sensors or other instruments; *calibrated data* — normalized raw data correcting for instrument, environment, or other experimental differences; *validated data* — errors removed; *derived products* — computed values, graphs, and models; and *interpreted data*. In addition, the *metadata* associated with the data must be preserved and accessible. This is the information required to identify data of interest based on content, validity, sources, or other selected properties. This data encapsulates information such as:

Who did what and when
Device characteristics
Transform definition
Documentation and citations
Structure

It is imperative that the metadata remain attached to the data or the data become meaningless and unusable.

3.1.2. Locating Data: Early in any scientific inquiry, the need to find data becomes critical to the successful outcome of the investigation. Hypotheses need to be corroborated, or perhaps, archived data is being mined for possible undiscovered properties. It becomes necessary to address questions such as:

What data collections exist?
Is the data relevant to my interests?
Do useful data items exist?

This implies the need for a rather general data browse capability providing facilities for locating data sets, and further scanning them for indications of probable interest.

3.1.3. User Interfaces: Interdisciplinary investigations are becoming more widespread. Interfaces to database management facilities will have to support the domain specific jargon of the various sciences in order to foster this kind of investigation. Practitioners can not be expected to master all domain jargon to engage in casual collaboration.

In addition to managing domain specific lexicons, interfaces must provide facilities for better integration with applications. Much of scientific database management is driven by stringent computational requirements. To facilitate flexible investigations and high performance analyses, there must be better means for: hooks to special application programs; audit trail provision; communication with outside users; user-transparent storage media hierarchy. Above all, data management systems must be extensible allowing the user to define new types, structures, manipulation operators and displays.

3.1.4. More Flexible Representational Structures: Perhaps the single unifying cry is that existing data models are inadequate for science data needs. The relational model has some advantages. Chief among them is that it is well-defined and has solid theoretical underpinnings. And, more pragmatically, it exists within successful commercial products. However, the semantic gap between the relational model and what scientists need must be addressed. We must seek alternatives such as extending the relational paradigm, object-oriented database technology, extensible tool kits, and logic databases. We must also consider alternatives to the relational model for efficiently supporting temporal, spatial, image, sequences, graph, and other more richly structured data.

3.1.5. Appropriate Analysis Operators: One area of concern noted by most of the participants was the lack of appropriate operators within existing DBMS for manipulating the kinds of data encountered in scientific applications. More flexible comparators are necessary when attempting to match DNA sequences or retrieve image data. There was not universal agreement as to where these operators belong — within the DBMS as intrinsic operators or external to the DBMS as utilities or part of an analysis package. The approach used now is to have a commercial DBMS export data for use by external utilities. Often the data can not be exported in a format compatible with the utility program so ASCII files are created and subsequently massaged into an appropriate form. If results of the analysis are to be saved, the process must be reversed and the updated data imported back into the DBMS. Since there are no standards this is a tedious and time consuming process.

Extensible database technologies provide the mechanism for embedding custom operators into the DBMS. A philosophical question arises as to how much custom functionality is desirable within the DBMS. It may be more appropriate to create an integrated analysis environment in which a DBMS can interact in a standard way with a variety of useful tools.

3.1.6. Standards: Heterogeneity in data and operational environments is a fact of life. We must find ways to promote consistency within scientific disciplines. It is clearly unreasonable to expect all disciplines to converge on some unifying standard, so heterogeneity will continue to be a force to be reckoned with. However, there are already instances of standardization within disciplines (e.g., the astrophysics community has endorsed FITS as its data interchange standard) and this trend should continue.

It was noted that the most successful standardization efforts arise when an organization creates a useful data format and associated analysis tools and then distributes them widely at no charge.

3.1.7. Standards for Data Citation: There was strong sentiment that data used in the conduct of an investigation should be cited prominently. A standard citation mechanism would allow other researchers to locate and examine precisely the data used in the investigation.

In a related problem, it was noted that much of the interesting metadata is actually citations into the scientific literature. These citations should be handled in a standard way so that where possible their content may be examined as part of a search for important data or to help access the quality of data which is being browsed.

3.2. Other Issues in Scientific Database Management

The following issues and problems associated with scientific database management arose in the discussions of the workshop. We have rated them as less important issues because, either (1) there exist partial, although imperfect, solutions to the problem, (2) they seemed to be less frequently encountered, or (3) the problems are not readily amenable to a technological solution. While these may be less important from our perspective, there exist views in the scientific database management world (using the general dimensions described in section 2) in which they can be very important.

3.2.1. Data Set Transmission: Data sets residing at one site (usually the collecting set or a designated repository) may have to be transmitted to the site where subsequent analysis will take place. Participants observed that there exist a number of wide area networks of sufficient bandwidth and reliability to handle most reasonably sized data sets. However, transmission of very large data sets may be slow. The delay in response time may associated with the time to access and transmit the data set at the host site, as much as network delays.

3.2.2. Conversion of Data Sets to Local Site Format: A data set received from a foreign site may be in a format that is incompatible to the analysis system at the local site. Issues of subsequent data set conversion tend to involve those of adequate metadata to interpret the format and structure of the data discussed above, and the more general issue of standards for scientific data. Some discipline specific models for data exchange already exist, such as FITS in the astronomical community.

3.2.3. Making Multiple Data Sets Comparable: Analysis involving multiple data sets from disparate sources can be difficult. Relations obtained from Oracle, Ingres, or other relational DBMS need not be immediately comparable. With data coming from even less rigid data models, such as OODBMS, the problem is magnified. A straightforward technical approach involves converting all data sets to a local standard as described above. At a much deeper level, this problem involves the general issue of data fusion, which must take into account the semantics (or intended meaning) of the data items in order to make meaningful comparisons.

3.2.4. Need for Interoperability of Multivendor DBMS: In some ways this is a subset of the preceding issue, in some ways it is a superset. The goal would be to allow analysis programs running under the aegis of one DBMS to directly query/access data stored in a different DBMS.

3.2.5. Quality Assessment of a Data Set: Participants repeatedly noted the difficulty in assessing the quality of a received data set. While quality assessment has always been a fundamental scientific problem, many of the technical barriers revolve around the problem of insufficient metadata to interpret the data.

3.2.6. Volume of Scientific Data, Need for Permanent Archiving: The expected volume of sensor generated scientific data is awesome. In the coming decade it will far out strip the resources available to analyze it. The issue is: should (can) all of it be archived for possible later interpretation, or should (can) it be passed through some preliminary filter to determine what should be saved.

A directly related issue — that data collection is often well-funded, while data management is poorly funded, if funded at all, was a recurrent theme.

3.2.7. Proprietary Behavior of PI's with Respect to Data Sets: Data collecting PI's and their funding agencies have little incentive to release verified, but uninterpreted data sets, in a timely fashion. In fact there are a number of sociological and monetary disincentives to do so.

3.2.8. Data Management is not Respected in Scientific Communities: It was repeatedly noted that data management is not an attractive career path within any of the scientific disciplines, whose primary goal is one of discovery.

This summary should convey to the reader the variety of problems faced by scientists in the management of their data. Unless these problems are addressed now, scientists in the 90's will find data management an increasing barrier to continued progress in their fields.

6 SSDBM

Preliminary Call for Papers

Sixth International Conference on Scientific and Statistical Database Management
Zurich, Switzerland, June 16-18, 1992

The Conference:

This international conference provides a forum for the presentation and exchange of current work in the field of scientific and statistical database management. We are particularly soliciting papers on new concepts, novel ideas, and state-of-the-art research results relevant to database and knowledge base design from a theoretical as well as applicative point of view. To encourage the dialog between practitioners and researchers, we invite contributions also from domain-scientists, reporting experiences in data management from their field. Topics of interest include but are not limited to: modelling and semantics, query languages and user interfaces, physical organization, security, scientific databases, data analysis and visualization, management of temporal and spatial data, evolution of scientific, engineering, or statistical applications.

Submission of Papers:

Authors are requested to submit five copies of the complete paper, not exceeding 20 pages, as follows.

- **Contributions from the American continent to (co-chairman):**
  James C. French
  Institute for Parallel Computation
  School of Engineering and Applied Science
  Thornton Hall
  University of Virginia
  Charlottesville, VA 22901 USA

- **All other contributions to (general chairman):**
  Hans Hinterberger
  Institute for Scientific Computing
  ETH-Zentrum
  CH-8092 Zurich
  Switzerland

Time Schedule:

December 15, 1991	Deadline for Submission of Papers
April 10, 1992	Notification of Acceptance
May 29, 1992	Final Paper to be Included in Proceedings

Program Committee : (not complete)

R. Cubitt (Luxembourg), J.C. French (USA) co-chairman, P. Golder (UK), H. Hinterberger (Switzerland) General Chairman, J. Klensin (USA), M. McLeish (Canada), Z. Michalewicz (USA), G. Ozsoyoglu (USA), M. Rafanelli (Italy), R. Pecherer (USA), D. Rotem (USA), A. Shoshani (USA), A. Westlake (UK), M. Zemankova (USA).

Organizing Committee: (not complete)

R. Cubitt (Luxembourg), H. Hinterberger (Switzerland), J.L.A. Van Rijckevorsel (Netherlands).

Sponsors : (not complete)

Swiss Federal Institute of Technology, Zurich.

IEEE COMPUTER SOCIETY
Technical Committee Membership Application

COMPLETE AND RETURN THIS FORM TO:
IEEE COMPUTER SOCIETY
1730 Massachusetts Ave., NW
Washington, DC 20036-1903

INSTRUCTIONS:
Please print in ink or type, one character per box. INFORMATION OUTSIDE BOXES WILL NOT BE RECORDED. Street addresses are preferable to P.O. boxes for mail delivery. International members are requested to make best use of available space for long addresses.

Last Name | *First Name* | *Initial* | *Dr./Mr./Mrs./Ms./Miss/Prof.*

Company/University/Agency Name | *Dept. Mail Stop/Bldg./P.O.Box/Apartment*

Check One: ☐ *New Application* ☐ *Information Update*

Street Address/P.O. Box | *Date: Month* | *Day* | *Year*

City | *State* | *Postal Code*

Country | *Office Phone* | *Home Phone (optional)*

E-mail Network | *E-mail Address (Mailbox)*

Telex Number | *IEEE Member/Affiliate Number*

I am a member of the Computer Society ☐ *Yes* ☐ *No*

The Computer Society shares its mailing lists with other organizations which have information of interest to computer professionals. If you do not wish to be included on those lists, please check here: ☐

If you are presently a member of the IEEE Computer Society, Place an X in the left-side box below corresponding to a TC of which you would like to be a member. If you are not a member of the Computer Society and would like to be a member of a TC, place an X in the right-side box below.

For each committee, please indicate activities of the Technical Committee in which you would like to become involved.

I would like to become involved in the Technical Committee and:

	MEMBER	NON-MEMBER		Conferences	Newsletters	Standards Development	Education
01	☐	☐	**Computational Medicine**	☐	☐	☐	☐
02	☐	☐	**Computer Architecture**	☐	☐	☐	☐
03	☐	☐	**Computer Communications**	☐	☐	☐	☐
04	☐	☐	**Computer Elements**	☐	☐	☐	☐
05	☐	☐	**Computer Graphics**	☐	☐	☐	☐
06	☐	☐	**Computer Languages**	☐	☐	☐	☐
07	☐	☐	**Computer Packaging**	☐	☐	☐	☐
08	☐	☐	**Computers in Education**	☐	☐	☐	☐
09	☐	☐	**Computing and the Handicapped**	☐	☐	☐	☐
10	☐	☐	**Data Engineering***	☐	☐	☐	☐
11	☐	☐	**Design Automation**	☐	☐	☐	☐
12	☐	☐	**Distributed Processing**	☐	☐	☐	☐
13	☐	☐	**Fault-Tolerant Computing**	☐	☐	☐	☐
14	☐	☐	**Mass Storage Systems & Technology**	☐	☐	☐	☐
15	☐	☐	**Mathematical Foundations of Computing**	☐	☐	☐	☐
16	☐	☐	**Microprocessors and Microcomputers**	☐	☐	☐	☐
17	☐	☐	**Microprogramming**	☐	☐	☐	☐
18	☐	☐	**Multiple-Valued Logic**	☐	☐	☐	☐
19	☐	☐	**Oceanic Engineering and Technology**	☐	☐	☐	☐
20	☐	☐	**Office Automation**	☐	☐	☐	☐
21	☐	☐	**Operating Systems**	☐	☐	☐	☐
22	☐	☐	**Optical Processing**	☐	☐	☐	☐
23	☐	☐	**Pattern Analysis and Machine Intelligence**	☐	☐	☐	☐
24	☐	☐	**Personal Computing**	☐	☐	☐	☐
25	☐	☐	**Real-Time Systems**	☐	☐	☐	☐
26	☐	☐	**Robotics**	☐	☐	☐	☐
27	☐	☐	**Security and Privacy**	☐	☐	☐	☐
28	☐	☐	**Simulation ***	☐	☐	☐	☐
29	☐	☐	**Software Engineering**	☐	☐	☐	☐
30	☐	☐	**Test Technology**	☐	☐	☐	☐
31	☐	☐	**VLSI**	☐	☐	☐	☐
32	☐	☐	**Computer and Display Ergonomics**	☐	☐	☐	☐
33	☐	☐	**Supercomputing**	☐	☐	☐	☐

****Please note that the Technical Committees on Data Engineering and on Simulation now charge an annual membership fee. The rates are $15 for Computer Society members, and $25 for non-members. Checks should be made out to IEEE Computer Society and sent to the address above. Credit cards are accepted ONLY FROM NON-U.S. MEMBERS!***

THE IEEE COMPUTER SOCIETY
1730 MASSACHUSETTS AVENUE, N.W.
WASHINGTON, DC 20036-1903

NON-PROFIT ORG.
U.S. POSTAGE
PAID
SILVER SPRING, MD
PERMIT 1398

Mr. Timoleon K. Sellis
Dept. of Computer Science
Univ. of Md.
University of Maryland
College Park, MD 20742
USA