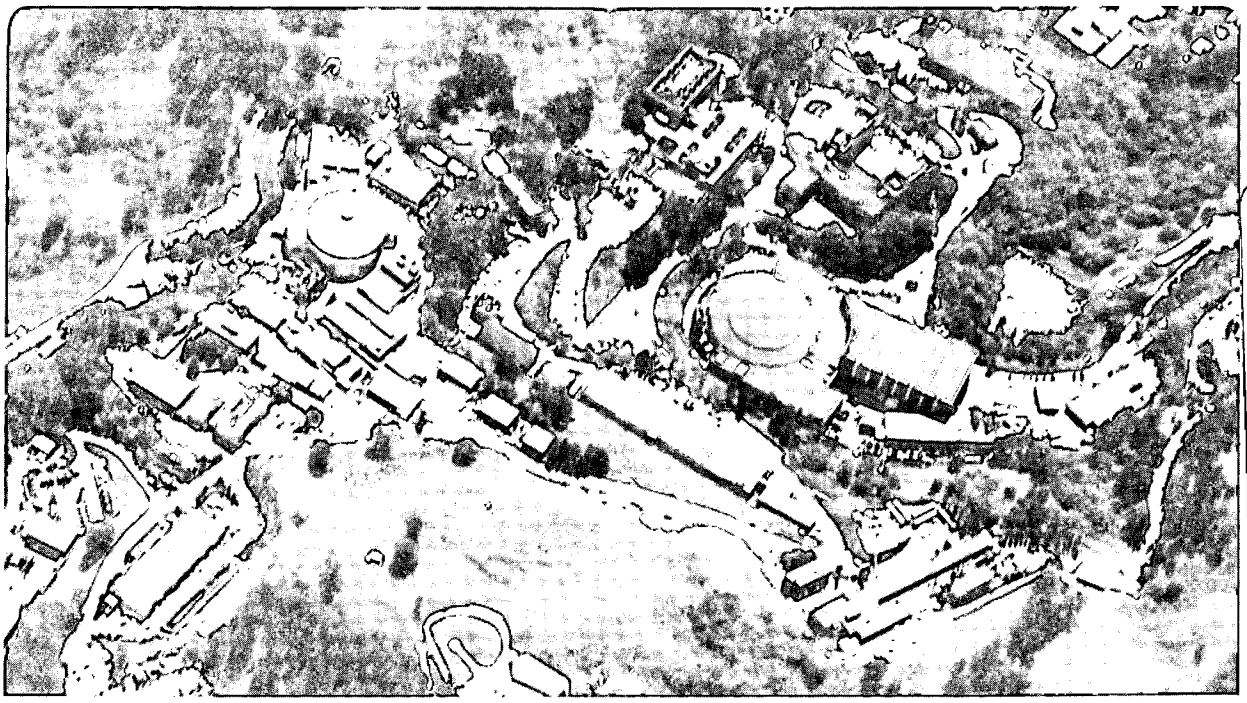
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Information and Computing Sciences Division

Database Management Software at LBL: An Introduction and Comparative Assessment

J.L. McCarthy, R. Firestone, F. Gey,
C. Madison, J. Olivares, and A. Spurlock

February 1990



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

DISCLAIMER

LBID-1585

Database Management Software at LBL: An Introduction and Comparative Assessment

Report of the *ad hoc* Data Management Resource Group

John L. McCarthy (Chair), *Computer Science Research & Development*
Richard Firestone, *Table of Isotopes Project*
Fred Gey, *Computer Science Research & Development*
Claudia Madison, *Workstation Group*
Jose Olivares, *Library*
Arlene Spurlock, *Applications Group*

Information and Computing Sciences Division

February, 1990

Prepared for the U.S. Department of Energy under Contract DE-AC03-76F00098

1. Introduction

The purpose of this report is to assist people who want to organize their data and are considering various database management system (DBMS) options. It is an introductory document which grew out of discussions among people working with data management systems at Lawrence Berkeley Laboratory.

Data management is a rapidly changing field. During the past decade, the relational data model has become predominant over prior network and hierarchical models, but it may in turn give way to object-oriented models in the future. Old products are continually being upgraded and new products come on the market every year. Many specific details about individual DBMS products in this report soon will be obsolete, but we hope that its general points will be of continuing use.

1.1. Organization of This Report

The remainder of this chapter outlines the rest of the report, describes the committee that produced it, and suggests where to go for more help. Chapter 2 introduces basic data management concepts and briefly discusses some of the issues of comparing data management software. Chapter 3 reviews why different data management problems require different database software capabilities. Chapter 4 outlines different data management system features that address diverse user requirements. Chapter 5 gives a brief overview of database software currently running on LBL computers (from mainframes to microcomputers) and compares their major strengths and weaknesses. A selected bibliography suggests further reading on different aspects of data management, and a short glossary of DBMS terminology gives cross-references for variant terminology used by major LBL database systems that our group studied.

Each chapter is relatively self-contained. They need not be read sequentially, though someone unfamiliar with data management will find it easier to proceed in order. Those who simply want a brief overview of current data management systems at LBL, for example, can proceed immediately to chapter 5.

Two unpublished documents (available separately) provide further information for those interested in more details. *1988 LBL DBMS Comparisons* evaluates selected systems in several different ways, including tables comparing detailed features and command languages, how each DBMS was used to carry a specified set of tasks for a small test dataset, and detailed listings of actual code used to do example problems. *DBMS Vendors* contains contact names, addresses, etc., for information on individual systems.

1.2. Background

The Data Management Resource Group (DMRG) began meeting in July, 1987, to facilitate exchange of information among people working with different database management systems at LBL. Those who were able to participate on a regular basis above and beyond their normal responsibilities are listed as co-authors

Other individuals who participated on the committee at earlier stages included Allan Konrad from the Computing and Communication Resources Group, Gary Wagman from the Particle Data Group, and Bob Leedy from the SSC Central Design Group.

Drafting of this report began in 1988, but was not completed for over a year due to demands of other projects. In retrospect, that may have been a blessing in disguise, because it gave time for the contents to "season" Given the longer time perspective, we have tried to filter and reorganize our results in a way that focuses on their less transient aspects.

1.3. Where to Go for More Help

This report is intended to provide a starting point for people with database needs at LBL. It certainly does not provide enough information in itself to select a suitable DBMS, but we hope that it will help facilitate that process for others. For further information, there is a selected, annotated bibliography at the end of this report that spans the range from brief introductions to computer science texts and specialized technical product comparisons. Readers should also consult the growing cadre of people at LBL familiar with various aspects of database management and call DBMS vendors (list available separately) to get up-to-date, system-specific information.

2. Basic Data Management Concepts

People use a wide variety of software tools to help organize and manage computer-readable information, from simple name and address lists and bibliographic references to laboratory equipment inventories and experimental results. Such tools range from simple text editors and spreadsheet programs on personal computers to sophisticated database management programs that run on large, multi-user mainframes. Some are specialized for particular types of information, such as bibliographic citations or mechanical design drawings, while others are more general in nature.

In order to choose the right tool(s) for a particular set of applications, it is helpful to be acquainted with basic terminology and concepts that are common to most types of data management software. Unfortunately, database terminology is still far from standardized. Various authors and database vendors may use several different terms for the same thing, and sometimes the same term for different things. The glossary at the end of this report attempts to clarify some of the most common terms, along with particular terminology used by selected database management packages currently in use at LBL. The next section briefly reviews some of the fundamentals in terms of a simple example.

2.1. Records and Fields, Types and Instances

Consider a simple address file. A typical address file consists of individual *records* , each of which contains several different *fields* (sometimes called *attributes*) for different types of information such as name, title, organization, street address, city, state, country, postal code, and telephone number(s). "Name" is used here to indicate a generic *type* of field, while John Jones would be a particular *instance* or *value* of that field. *Domains* are sets of allowable values which may be associated with one or more attributes (e.g., positive integers, integers 1 through 12 for months of the year, "M" and "F" for sex codes, and so on).

Records can also have types and instances, such as a bibliographic record type and a specific instance for *The Handbook of Chemistry.*

Database records usually pertain to real world *entities*, and fields to particular *attributes* of those entities. Some databases pertain to a single type of entity (e.g., addresses), while others contain different attributes for different types of entities.

Data management tools become even more important when one needs to maintain related information on several different types of entities. If "organization" were a field in an address record, for example, the organization name could be used to automatically *link* the address information to an "organization" record type, which might include further information (fields) on particular organizations. Information about the organization could thus be maintained *independently*, rather than stored and maintained redundantly with the address record for each individual.

2.2. Relational and Other Data Models

During the past decade, the relational data model has evolved from a mathematical, research concept to the new conventional wisdom. Nearly every data management vendor now claims that its system is "relational." In the trade press, the term "relational" is often used to characterize any system that represents data in logically related "tables," and that links related information across multiple tables. Computer scientists tend to apply a more rigorous and restrictive standard to use of the term. [Gey and Wong, 1986] summarizes a series of criteria proposed by E.F. Codd, who originally developed the relational data model [Codd 1970].

The standard relational data model requires that all information be organized as single values (cells) within simple *relations* -- tables whose columns represent *attributes* (fields) and whose *tuples* (rows) represent individual record instances. Relational systems can deal with two or more tables simultaneously in retrieving and updating data.

While relational tables are a natural way to represent many types of information, some systems support direct representation of more complex data structures, such as hierarchies and networks. CODASYL [O'Connell, 1975] is a network database standard that preceded the relational model. Sometimes database

designers find it useful to begin with a richer logical data model -- e.g., entity-relationship diagrams -- even though implementation may be carried out on a relational system (for further discussion, see [Estvanik, 1987], [Gillenson, 1988], and [Martin, 1977]). Object-oriented data models and data management systems are receiving increased attention from both researchers and commercial vendors because they promise a richer variety of semantic representation, including inheritance, aggregation, and incorporation of active operations (e.g., "raise" salary) as part of the database environment. Some systems may permit more than one logical view of the data -- e.g., a relational interface as well as a CODSYL interface.

2.3. Query Languages and User Interfaces

Non-procedural languages (sometimes called 4th Generation Languages or 4GL's) specify what information is to be drawn from a database, rather than the specific procedures used to do so. The DBMS interprets 4GL commands and turns them into a series of procedures that will effect the desired result. Other user interfaces include menus and forms that prompt users to fill in the type of information they want (e.g., Query By Example, or QBE).

SQL, the Structured Query Language developed at IBM Research, is a non-procedural language for creating, accessing, modifying, and managing data stored in a relational system [Date, 1987; Finkelstein, 1987]. SQL may be used directly by users or invoked by calling sequences from within an application program. SQL has become an international standard; a standing committee of the International Standards Organization (ISO) is responsible for its continued development and future modifications. As with standard compiler languages, such as FORTRAN, many vendors have developed extensions to the base-level SQL standard. Some have argued that SQL's importance as a standard for data management has been exaggerated because the bulk of most implementations usually involves non-standard extensions such as forms specification, report definitions, and so on [Stonebraker, 1988].

2.4. Comparing Data Management Software

Comparison of data management software is not easy. There are many aspects to consider which have differing importance depending on particulars of applications and data. A specific DBMS may be optimal for one project but terrible for another. Some systems are fast and highly versatile, but may be difficult to learn or use. Obtaining a general purpose system for a general user community usually requires compromises on a variety of issues.

Ideally, it is best to choose database management software to fit application needs, and then determine hardware necessary to support the software and applications, rather than vice-versa. In many cases, however, the hardware and operating system environments may already be "fixed." One or more database systems may already be available.

The selection process can begin by ennumerating which characteristics are mandatory (e.g., hardware environment) and the relative importance of other characteristics -- in light of specific applications requirements. Each organization and user will have somewhat different "weights" ranking the relative importance of different characteristics. Most software selection methodologies recommend explicit numeric representation of such weights and rankings prior to review and analysis of specific systems.

Comparative evaluations require considerable time, effort, and resources. Comparative analyses available in the trade press (e.g., [Finkelstein, 1988]) or from specialized publications (e.g., [DataPro Research, Software Digest]), can provide a useful starting point. Talking to users who are already running similar applications using a DBMS can be very enlightening. But it is always necessary to tailor and weigh such information in light of local requirements. The next chapter outlines some of the major kinds of local requirements that deserve careful consideration.

3. User Application Requirements

Distinct kinds of data management problems have varying requirements and different solutions. Software that will match your needs depends on the kinds of data you are dealing with, and how you wish to organize, retrieve, and manipulate it. Some collections of data may not require a full-blown database management system -- sometimes a simple text or spreadsheet file may suffice. In other cases, the type or amount of data may exceed current data management system capabilities. Potential DBMS users need to assess their application needs and match those against available software options.

What kinds of situations can benefit from a database management system? Some of the more common ones are as follows:

You have an amount of data that is unwieldy to use without something to manage it -- in terms of the number of objects (rows), attributes (fields), tables (record-types), or relationships;

Your data is changed regularly, perhaps potentially by many people at once;

You need software to help collect information and insure its validity;

You need access to objects of interest based on different attributes (e.g., author, title, and subject term indexes for a bibliographic database);

You need to display a set of information in a variety of different ways;

You need to enforce security constraints on access to information by different types of users;

You need to keep track of how your data is being used or changed;

In the future, you may want to change the ways in which data are collected and stored without having to change application programs that use the data.

Once you have decided that a data base management system may help, you can narrow the choices further by explicitly specifying your requirements. The major types of requirements discussed below are as follows:

- Common Environments and Costs
- Data Sharing and Portability
- Users and General Types of Use
- Data Characteristics
- Data Volumes and Operations
- Data Security and Privacy
- Information Integrity and Quality Control
- Special Considerations

3.1. Common Environments and Costs

Often a DBMS decision is constrained by existing hardware and software. *Do you have to run on existing hardware and operating systems?* In many cases a computer hardware vendor will supply a DBMS which is closely integrated with its own hardware at significant savings over products from independent software vendors. Yet relying on a hardware vendor's DBMS ties users to the vendor for the indefinite future, whereas independent software vendors often support versions which run on different types of computers. *Are there any DBMS's already running on machines to which you have access?* Using an existing DBMS can save significantly in terms of acquisition, development of expertise, shared costs, and so on.

DBMS software varies dramatically in cost, from $100 on a microcomputer to $30,000 on a shared mini-computer to $250,000 for a large mainframe system. DBMS's also consume significant computer resources (cpu cycles, storage space, I/O channels, etc.). *What are your budgetary constraints? Can you share costs (and control) with other projects?*

3.2. Data Sharing and Portability

Sharability of data files is often more important than other considerations of functional capabilities. *Do you have a single application, or will you need to integrate several different applications? Will the application need to import files from other applications or software? Will there be a requirement for "bulk load" of large data files which are already being used locally or elsewhere? What kind of data export facilities are needed? Standard data base formats (e.g., DBase)? spreadsheet formats? word processor formats?* DBMS tools to simplify import and export of data vary widely in efficiency and ease of use.

Networked environments with heterogeneous hardware pose other challenges and opportunities for database sharing. Databases

may reside on a common file-server, accessed by client processes on remote machines. Alternatively, a local client process may run in conjunction with X-Windows servers on remote machines -- in order to provide a common interface as well as common data. *Can the candidate DBMS's run as remote servers? Will their interface(s) run under X-windows?*

Will you need to move applications from one machine to another? If the application has an expected lifetime of more than a few years, the answer is probably yes. Applications written entirely in standard or vendor-specific languages (sometimes known as Fourth Generation Languages or 4GLs) such as SQL or FOCUS may migrate more easily to equivalent software on other machines than applications written in third generation compiled languages (e.g. Cobol, Fortran) with direct calls to DBMS library subroutines. Note however, that except for SQL, DBMS users will remain locked into a particular DBMS vendor in using a 4GL. Since many vendors have extended SQL in idiosyncratic ways, since there are as yet no standards for embedded SQL, and since there are no standards for other important aspects such as form and report definition languages, portability across DBMS's is still difficult at best.

3.3. Users and General Types of Use

Some applications access database software via intermediate programs (e.g., data analysis programs), while others directly utilize DBMS user interfaces and tools such as query menus, report generators, and so on. *Will the DBMS be used directly by people, by computer programs (written in C, Fortran, Cobol, Pascal, PL/1, etc.), or both?* Database systems differ in terms of which programming languages they support, whether DBMS calls from the programming language are compiled or turned into intermediate code by a preprocessor, and compatibility with programming tools such as debuggers.

To what extent does data need to be shared? Data sharing may be accomplished in a variety of ways. Multiple users can share a single central database via networks or telephone dial-up (distributed use). Some database systems have "front-ends" which run on different types of hardware. An increasing number of data management systems also are beginning to support distributed databases -- in which different parts of the data reside on different machines, sometimes even on different types of hardware (distributed data).

Will more than one user need simultaneous access to the data? Can the multi-access be read-only or is there a requirement for two or more people modifying or updating the database at the same time? Single user and read-only access can be satisfied by microcomputer systems or relatively unsophisticated multi-file management systems (such as Datatrieve on VAX computers). True multi-user concurrent update capability calls for sophisticated system security, data integrity and recovery facilities.

Will users be inexperienced or sophisticated? Some users are willing to put up with a longer learning curve to get more power and flexibility.

3.4. Data Characteristics

Most DBMS's support physical storage of different data types, including fixed length characters, integers, and floating point numbers. Some support special data types such as dates, variable length text, and bitmaps (e.g., digitized pictures). A few are beginning to permit "user-defined" data types. Some store data in variable length "tagged" fields, while others store data in fixed length fields whose types and lengths must be pre-specified separately in a system dictionary or schema. The former can save storage space if one has highly irregular data -- e.g., variable length text entries or sparse data values -- and they usually permit addition of new data fields without reloading data. The latter can save space and access time if data values are more uniform, but they may prohibit addition of new data fields without reloading data.

What kind of data is being stored? Is it scientific data (double precision reals, vectors, matrices)? Is it variable length text (documents and bibliographic records)? Is it multiply occurring? (e.g., bibliographic citations frequently include multiple authors) Do you need to handle dates, personal names, or currency in special formats? Do you need to store, retrieve, and manipulate graphic images (e.g. digitized drawings or photographs) ?

What about sounds, speech segments, or other 'Multi-media' objects? Some systems are well-suited to handle text and personal names, while others are not. Some are stronger with respect to numeric data (e.g., built-in trigonometric functions, scientific notation, etc.). A few have special facilities for digitized images or sound.

Although a clever database designer can usually figure out ways to represent a particular set of data within the constraints of a given logical data model (relational, hierarchical, network, or object-oriented), mapping between a DBMS based on a particular logical data model and a specific application that does not conform easily to that model may be difficult or inefficient at best. *Do you need to represent information about many different types of entities and the relationships between them? Is the data quite regular (e.g., many entities with the same attributes) or sparse and irregular? Do the entity types, attributes, and relationships themselves change frequently over time? Do you need to edit, manipulate, and display complex objects?*

Most DBMSs have length restrictions on the size and number of data elements that can be used within a single file. Some have a maximum size for text fields (e.g. 256 characters). If an application calls for data structures which exceed these limits, artificial restructuring of the database may be necessary to make the application work with the chosen DBMS product. *How many data base files (or relations) will be needed? How many data elements or fields do you expect to have in your data base? What size are they? Is there a need for unlimited length text descriptions?*

3.5. Data Volumes and Operations

Some large applications may exceed DBMS limits in terms of data volumes and operational demands.*How many files will the application require to be open and simultaneously accessible for linkage of data? How many records (tuples)? How many values in all files and records?* Many DBMS products place a strict limit on the number of files that can be manipulated simultaneously by an application. Others limit the total number of records in a file. Still others require that you pre-allocate disk storage space for the total expected amount of data to be ultimately stored in the database. Obtaining estimates of the volume of storage necessary can also yield some idea of the disk storage costs which the DBMS application will incur.

Data volatility impacts both storage and DBMS update facilities. Archival data can be placed upon lower cost storage media, while very frequently updated data may require extra storage for maintenance of backups of application critical data to assure continuous operation with minimal disruption in the event of computer hardware failure.*What is the volume and frequency of changes to the data? Are these changes updates or modifications to the database (such as updating a person's phone number), or are they merely additions to the database (e.g. adding a new citation to a bibliography)?*

Some DBMSs are optimized for rapid access to single records by multiple simultaneous users (airline reservation systems are an extreme example). Other DBMS's bog down if even a few users issue simultaneous multi-record-type queries. Some include query optimizers.*How often will the database be queried? by how many simultaneous users? How rapidly should the DBMS respond to a query? Will the DBMS be used for real time data acquisition or user interface control?* If so, speed and efficiency are prime considerations.

3.6. High Level Tools

Mature database systems usually include a variety of high level tools for non-programmer users, such as query and report generation facilities, which can differ substantially in terms of power and ease of use.

Some DBMS's include query interfaces other than SQL or other command languages. A Query-By-Example (QBE)-style interface, for example, lets users edit templates containing example attributes for objects of interest. Some systems (particularly microcomputer DBMSs) include business graphics as a reporting and display option. Others facilitate linkages to other analysis and display software (e.g. spreadsheets, drawing programs, statistical packages). *Do you expect to make ad-hoc queries on the data? Will they involve arithmetic or higher math functions? Do potential users want a command, menu, or*

graphical interface? Do you need to make standard formatted reports of the data (possibly including counts and subtotals)? Do you want graphic displays (e.g. bar charts/pie graphs)?

Some DBMSs have extensive and elaborate tools to ease the job of building a complex application, while others are very limited in their applications development repertoire. *How many data entry and inquiry screens does the application expect to need? How are these screens ordered hierarchically? Will the screens require simultaneous access to multiple data files (e.g. personnel file information combined with project scheduling information)?*

3.7. Data Security and Privacy

Multi-user database systems provide facilities to specify different types of privileges (e.g., read, write, delete) for different users with regard to different databases. Some support such facilities at various levels of granularity, from entire databases down to individual data element values. *Is the data sensitive? Does it need some privacy protection from unauthorized usage? Do you need encryption, password, account security? Is the security required at the file, record or element/field level?*

3.8. Data Integrity and Quality Control

One of the principle reasons for use of database management systems is to improve data quality and integrity. Four major types of integrity which some DBMS's can help enforce are illustrated by the following four examples:

1. Primary-key Integrity. A user creates a new employee record but fails to put in the employee's social security number, which is the primary identifier for locating that employee's record.

2. , Referential Integrity. A user assigns that employee to a non-existent department number (either through error or omission).

3. Domain integrity. The user types in the number 22 for the month-of-hire for that employee.

4. User-defined or Business Integrity. The department to which the employee is hired into has a personnel ceiling which will be exceeded if the employee is assigned to that department.

In addition there are other areas of data quality which an application may require which go beyond the above examples of fundamental DBMS integrity. For example, one may wish to attach footnotes to particular data values which give some description of their derivation and the experimental conditions under which they were collected. Different versions of the value may be stored as the experimental apparatus becomes more accurate. *Does the DBMS application need extensive input validation facilities for quality control? Will an audit trail of data value changes be required? Is there a need for authority tables of permissible data values for certain domains?*

3.9. Special Considerations

Other factors may also need to be considered in describing your own particular requirements. For example, *do other organizations with which you collaborate already use a particular DBMS?* Using a common DBMS makes it much easier to share data and applications. *Is it difficult to find and retain programmers?* The more popular the database system, the more likely you will be able to find programmers who already are familiar with it.

3.10 Requirements vs. System Characteristics

Once you have prepared a list of your own requirements, answering questions such as those outlined above, you can evaluate specific functionality of different data management systems in terms of your own particular needs. The next chapter outlines a general set of functional features that can provide a starting point for such evaluations.

If you are planning a major application or expensive DBMS acquisition, you may want to use a formal methodology, such as outlined in [Quinn, 1980]. Such methodologies typically involve assigning points or mandatory status to each requirement, and then evaluating each DBMS in detail, dropping those that fail to meet all mandatory requirements and assigning some fraction of the points in each category for corresponding features of each individual system being considered.

4. System Characteristics

The outline below lists major types of system characteristics which may be important in matching application requirements to specific database systems. Groupings of characteristics into more general categories are necessarily somewhat arbitrary, since many characteristics pertain to more than one category (e.g., data representation and input/output).

The categories shown here are not exhaustive; rather they are intended to give the reader a starting point for further consideration. More comprehensive and up-to-date check lists of this kind can be found in special reports on DBMS features, such as [Datapro], [Auerbach], and [Software Digest]. An unpublished *1988 LBL System Comparisons* document (available separately) includes a detailed features table based on this outline for selected database software.

Note that many database management system vendors offer separately priced modules for specialized functions such as individual program language interfaces, report generation, full screen forms input, and so on.

4.1. DATA ACCESS

Integration and network capabilities
- client-server facilities
- linkage to external software
- network compatibility (which ones?)
- distributed capabilities
  - partitioned schemas
  - distributed tables
  - distributed query optimization

Indexing and access methods
- hashing/randomization/key transformation
- B-tree indexes
- linked lists
- automatic pointer creation & updating
- create new index without reloading
- pass different elements, same record
- pass elements from different records
- pass pointer from record to record
- duplicate index keys
- automatic re-indexing on update
- composite key indexes

Interactive I/O
- Display commands
  - Standard (native) I/O format
  - Setable alternative formats
  - Default element list selection
- Relationships to local editor(s)
- Screen input forms definition
- Menu structure and flow control specs

Batch I/O and import/export
- Load initial database
- Unload/reload for restructuring
- Backup to tape
- Specific import/export formats

Custom format definition language
- Procedural or non-procedural
- Structured standard record in DBMS
- Definable elements (components)
- Control structures
  - if, then, else; while; exit/return
- Nesting of formats (frames, windows)

4.2. DATA DEFINITION

Data types and sizes
- Integer
- Floating point
- Character string
- Date
- Dollar
- Packed decimal
- Logical (True/False)
- Binary (bitstring, bit map)
- Executable (stored procedures)
- Computed/Redefined/Virtual
- User-defined types (e.g., graphic)

Data structures within records
- Variable length elements
- Multiple occurrences of element
- Simple hierarchy repeating groups
- Multiple (branching) repeating groups

Data structures between records
- Automatic validation and update
- Simple hierarchy (no branching)
- Multiple (branching) hierarchies
- Network relationships
- Records of same record type (plex)
- Many to many relationships
- Relational join of different record types

Schema definition
- Active, integrated role in DBMS
- Controls data definition, validation
- Is itself a standard DBMS record
- Updatable on-line
- Compiled by utility to produce system tables
- Separate record/dictionary for each database
- Changes do not require reload

Data dictionary components
- All levels (database, record, element, etc.)
  - Name
  - Aliases/synonyms
  - Comments, notes, etc.
  - Security and access control
  - Creation/update date(s)
- Data element (field or item) level
  - Internal structure (nested elements)
  - Occurrence (optional, fixed ,variable)
  - Length
  - Input validation checks
  - Table look-up validation
  - Input conversion functions, coding
  - Output conversion functions
  - Output format specification(s)
  - Default row/column headers
  - Missing data specifications
- Index (pointer linkage) level, if any
  - Elements being indexed
  - Target record to receive pointer
  - Record pointer refers to
  - Other element values to carry

Data Validation and Integrity
- Specified in schema or only in entry screens?
- Element level checks
- Inter-element checks--same record
- Inter-record checks
- User-specified subroutines
  - In database language
  - Exit to compiler language(s)

Support for Views
- Multi-record
- Selected, redefined elements
- User definable

4.3. LANGUAGES AND SUBLANGUAGES

Programming (host) language interface(s)
- Languages supported
- Embedded query language precompiler
- Via regular call statements
- Locking control
- Exits to programming language

High level command language(s)
- SQL support
  - ISO standard SQL
  - Embedded SQL
  - SQL extensions
- General and informational features
  - setable message modes
  - setable defaults
  - show command for parameters
  - help/explain facility
  - search history display
  - browse records, indexes sequentially
  - set global search modifier(s)
- Query/search commands
  - Selection features
    - multiple selection
    - recursive selection
    - case sensitive/insensitive searching
    - non-indexed fields
    - global search & actions
      - *using expressions*
      - *delete*
      - *merge to other tables*
  - Data types covered
    - numeric
    - string
    - date
    - time
  - String content operators
    - prefix
    - suffix
    - word
    - string
    - having
    - mask
    - with
    - partial match
    - wild card
    - phonetic
  - Relational operators
    - =, ~=, >, >=, <, <=
  - Range operators
    - before
    - after
    - between..and
    - from..to
    - boolean logic on multiple qualifiers
    - parentheses for explicit precedence
    - sequential series of search operations on successive lines
- Saving and manipulation of search result
  - Save result of search commands
  - On-line sort of search result
    - maximum sort levels
    - maximum concurrent sort levels
    - simultaneous ascending/descending
    - sort on virtual/calculated fields
  - Set operations (boolean save, combine)
  - Create arbitrary sets of records
  - Manipulations
    - comparisons (numeric, date, time, string)
    - arithmetic (numeric, date, time)
- Data modification commands
  - Batch and on-line
  - Add, remove, update, merge

Stored procedural or macro control language
- Sequences of query language commands
- If, go to, etc. control structures
- Ability to capture and parse user input
- Batch processes

Report writing language (see also I/O)
- Formatting
  - headers & footers, etc.
  - left, right, center alignment
  - automatic page numbering
  - current date, day, time
  - automatic field labels from schema
  - hierarchical sort key suppression
  - page breaks
  - user-specified field size and placement
  - flexible field formatting
    - money
    - embedded commas
    - common date formats
  - Reports to preprinted custom forms
- Aggregation
  - counts, subtotals
  - average, standard deviation
  - minimum and maximum values
  - control break post-computation
- Cross-tabulation
  - subtotals, counts in table cells
  - aggregate operations in tabulations
  - range collapsing (e.g., Quarter 1 = Months 1-3)
  - missing data types & handling
- Operations
  - pause between records
  - save & reuse named report formats
  - send reports to screen/disk/printer
  - merge output with external software
    - (e.g., graphics display or word processor)

4.4. OPERATIONAL ISSUES

Resource Requirements
- Hardware & operating systems
  - Hardware supported
  - Operating systems supported
- Output devices supported
  - Alphanumeric terminals
  - Graphic terminals
  - Personal computers
  - Hardcopy devices
- Compatibility across systems
  - Primary source code available
  - Primary developmental system
  - Different limits on different systems
  - Some features not on some systems
- System resource requirements
  - Main memory
    - minimum configuration
    - dbms
    - common data buffers
    - indices (if any)
  - Disk
    - minimum configuration
    - approximate % overhead
    - intensity of use
    - system files
  - Operation overhead
    - DBMS internal overhead--disk/CPU/RAM
    - access method/retrieval path(s)
    - more memory yields better performance
    - service functions/tuning
- Multi-user considerations
  - Re-entrant code
  - Multi-task
  - Multi-thread
  - Multiple copies for multiple users?

Limits and Maximum Values
- System level
  - Separate databases
  - Concurrent users
- Database level
  - Elements (fields)
  - Record types
  - Indexes
  - Concurrent users
  - Record types for simultaneous traverse
- Record level
  - Records
  - Elements (fields)
  - Bytes/record
  - Indexes
  - Repeating groups
- Repeating group level
  - Occurrences
  - Elements
  - Bytes
  - Levels of nesting
- Element (field) level
  - Total bytes
  - Occurrences
  - Name length
  - Number of aliases
- Index level
  - Indexes per file/database
  - Elements passed to single database
  - Record types associated with single index
  - Characters per index
  - Value length

Ease And Efficiency of Use
- Operations done without reload/lockout
  - Add new elements
  - Add new record types
  - Add new indexes
  - Rename and/or add new aliases
  - Change security specifications
- Over-all modularity for ease of learning
- Interface consistency across modules

4.5. DATABASE ADMINISTRATION

System management
- Initial installation procedures
- Maintenance and Updates
- Database Displays
  - Record structure
  - Structured dictionary
  - Data element table
- Operations and Scheduling
- Other Utilities

Use and performance monitoring
- Billing data
- Space utilization report(s)
- Access time report(s)
- I/O usage report(s)

Physical Storage & Optimization
- Compaction routines
- Page size, segments
- Buffer size(s)
- Dynamic automatic garbage collection

Security and Recovery
- Access control to what levels?
  - Database
  - Record-type
  - Element-type
  - Individual record instance
  - Specified element values
- Security mechanisms
  - Account number
  - Separate password(s)
  - Terminal ID
  - Program/module identifier
  - Security level number
- Different types of protection
  - Read, add, update, delete information
  - Specify permitted operations/functions
  - Encryption
  - Concurrent access
  - Setable global search modifier(s)
- Restart, recovery and logging
  - Journaling/transaction logging
  - Checkpointing
  - Warm restart/before images/roll back
  - Cold restart/after images/ roll forward
  - Valid(but incorrect) data recovery

4.6. COST AND SUPPORT ISSUES

Acquisition and annual maintenance costs
- Minimum configuration price
- Optional components
  - Unbundled subsystems
  - Bundled combinations
- Discounts
  - Educational
  - Multiple copies
  - Site license

Documentation and training
- Documentation
  - On-line documentation
    - Explain <term, command, etc.>
    - Search history and advice
    - Tutorials
    - Example <term>
    - Syntax <term>
  - Printed manual(s)
    - General query language user
    - Reference card(s)
    - Dba, database designer
    - Applications designer
    - Technical, systems programmer
- Training
  - Amount required
  - Courses offered, frequency, timing
  - Instruction manuals

Vendor support
- Assistance
  - Consulting, hotline hours
  - Installation assistance
  - Users groups
  - Contract support availability
  - Support personnel
  - Local expertise
- Reputation and position
  - Financial and institutional stability
  - Staff size
  - Reputation
  - Responsiveness to users
  - Commitment to DBMS
  - User base

5. Data Management Systems at LBL

LBL has developed, acquired and used a variety of software systems to help manage both scientific and administrative data. The Berkeley Data Management System (BDMS) was originally developed on LBL's CDC 7600 to support bibliographic and publication databases for the Particle Data Group. Somewhat later, a lab-wide committee recommended acquisition of System 2000, a commercial hierarchical system that also ran (after a fashion) on the CDC machines. As LBL shifted from CDC to DEC hardware for scientific computing and IBM hardware for administrative applications, it replaced System 2000 with DEC's Datatrieve and two systems that ran on IBM mainframes -- SPIRES from Stanford University for bibliographic and general purpose applications and FOCUS from Information Builders, Inc. for administration and business activities. With the advent of departmental mini-computers and personal workstations, the number of different data management software packages at LBL has proliferated still further.

Although a number of LBL's major scientific projects still maintain data on tapes and standard system files, an increasing number have begun to use commercial database software. These include the Table of Isotopes Project (Datatrieve), the Particle Data Group (SPIRES and ORACLE), the Earth Sciences Division (Ingres), and the SSC Central Design Group (Sybase and Informix), and the Human Genome Project (Sybase and Ingres). Most of LBL's administrative data is currently maintained on FOCUS.

Database management systems covered by this report are restricted to those that were familiar to committee participants and used routinely by LBL staff in FY 1988. These included Datatrieve, dBase, Focus, Oracle, SPIRES, and 4th Dimension. Several DBMS's currently used at LBL are not covered in detail because committee members were not sufficiently familiar with them; they are described briefly in the next subsection. Additional software that may be used for certain data management tasks (e.g., personal bibliographic systems such as BIB and REFER on Unix, EndNote on Macintosh, and ProCite on Macintosh and IBM PC's) were not studied in depth because of their more specialized nature.

The subsections below briefly describe individual database management systems at LBL. These descriptions include the types of hardware and operating systems on which each DBMS runs and representative LBL applications for which they are currently being used. More extensive discussion of each system and comparisons between them are included in *1988 LBL DBMS Comparisons* (available separately).

5.1. Datatrieve

DATATRIEVE is a product of the Digital Equipment Corporation (DEC). It was designed to operate on most DEC operating systems, including RSX and VMS. DATATRIEVE utilizes standard DEC utilities and file structures and is very suitable for managing data within a diverse DEC environment. It is available on the LBL-VAX cluster, where it has been used to prepare the *Table of Radioactive Isotopes*, as well as interactive access to a database on thermochemical properties of aqueous solutions.

Strengths. The principal strength of DATATRIEVE is its close compatability with the VAX-VMS operating system. It recognizes standard VMS files and allows straightforward interaction with various programming languages and VMS command procedures. Separate databases can be linked together and complex searches can be performed with minimal effort. There is virtually no limit to the size of databases handled by DATATRIEVE, and most large applications can be handled efficiently. As a mainframe facility, DATATRIEVE has inherent access to mail and networking utilities. For example, DATATRIEVE at LBL can be accessed from many laboratories through the high energy physics network (HEPnet). Data files resident on any node of HEPnet can be accessed through DATATRIEVE without directly copying those files to a local disk. DATATRIEVE can be readily learned by users familiar with VMS, and simple data searches can be performed by nonprogrammers. Documentation is substantial, and considerable on-line HELP is available.

Weaknesses. It is probably fair to say that DATATRIEVE's initial interaction with users

is less than 'user friendly'. DATATRIEVE is fairly bullet proof, but it provides cryptic responses to incorrect commands. Input syntax is slightly cumbersome and commands must be in upper case. Great power is available to the user for file modification but it is up to the user to establish the proper protection. It is completely possible for users to unintentionally destroy or modify files and delete previous versions. The power afforded the user to manipulate files should not be looked upon only as a disadvantage, however, because it also allows DATATRIEVE to interact freely with the outside world (e.g. FORTRAN programs) which is not always possible with other database management systems. A major disadvantage to some users is that DATATRIEVE provides only limited output capability. Screens, graphics, mailing labels, and other special applications are not readily available through DATATRIEVE. Some third party software, including a forms package, are available but have not been reviewed here. Also, as a proprietary system, DATATRIEVE is not directly available on non-DEC mainframes or PC's. The cost of using DATATRIEVE at LBL is complex. Access to this database management system is free on the cluster, but disk space and computer time charges can add up rapidly.

Summary. DATATRIEVE is probably most useful for users who are comfortable with VAX programming, have a large and complex problem, and are not concerned about fancy display capabilities (at least within the DATATRIEVE environment).

5.2. dBase

dBase III+ runs on IBM PC/XT/AT machines and compatibles using MS- or PC-DOS. It is known as the *defacto* standard DBMS for IBM microcomputers. At LBL the program has been used in administrative applications for list management and record keeping tasks. Space allocation and telephone information systems have been done in dBaseIII+. It is used as a prototyping system for applications that will be fully developed in a compiled language, as a part of a more elaborate system (for example, as a data-entry front end), and as a bridge to receive data from some system, parse or re-arrange it, and transfer it to a different system.

Strengths. dBase III+ may not be the best DBMS for any single application, but its strength derives from its versatility. It is possible to do most data management tasks with dBaseIII+. Non-programmers can create many applications in the menu-driven Assistant environment, which has a rich assortment of search, selection, screen-painting, and report-writing tools. For more complex applications, a programming language is available which includes the on-line query language commands so that relatively unsophisticated users can jump from on-line use to program creation. Finally, the dBase III+ user base is large, and there is a spin-off industry which provides a variety of add-on utilities, compilers, and training materials.

Weaknesses. dBase III+ is weakest with respect to repeating and/or complex data elements (as opposed to simple, single-valued fields). Currently its multi-user features are fairly primitive. It is slow in execution, though related products offer compiling of dBaseIII+ code for faster execution.

5.3. FOCUS

FOCUS is a fourth generation language and database management system that runs on a wide range of computer hardware and operating systems. At LBL it is available under MS-DOS or PC-DOS on IBM compatible PC-AT's, VMS on DEC computers, and CMS on IBM mainframes.

EXAMPLES of FOCUS applications running at LBL include the Stores Catalog and Issues Report, General Ledger, Monthly Effort Reports, Property Management, Information Services Recharge (Cost-Recovery) System, Travel Information Management System, and Project Management System

Strengths. FOCUS runs on a variety of hardware, under a variety of operating systems, with comprehensive features for entering, maintaining, retrieving, and analyzing data. It is designed for use both by people with no formal training in data processing and by data processing professionals who need powerful tools for developing complete applications. The non-procedural FOCUS language was designed to replace traditional programming languages. The simplicity of the language stems from the fact

that it uses simple English phrases. It is easy to retrieve data from files using the TABLE environment in FOCUS. The TABLE environment allows the user to produce a basic report with a minimum of commands. At the same time the TABLE environment has a wealth of capabilities which allow users to specify headers, footers, page-breaks, line-skips, groupings, sub-groupings, aggregations for rows, columns, grand-totals, groups and sub-groups. Users can COMPUTE variables on the fly, specify nested sorts, and print fields ACROSS a page or OVER each other. Users may specify conditional clauses which records must meet. FOCUS can read many different file structures. This allows users to issue the same TABLE requests against FOCUS files, RMS files, Lotus files, ORACLE files, dBase files, etc.

Weaknesses. FOCUS has several "environments": TABLE (already mentioned), MODIFY, ANALYSE, GRAPH, REBUILD, SCAN and the home environment, DIALOG-MANAGER. The environments share command names and syntax. However, many commands act differently in different environments. The syntax is not always straight forward. Documentation has traditionally been very poor for FOCUS. IBI has relied on training classes for explaining FOCUS's features. Until recently, local expertise for support of FOCUS in a VMS environment was not readily available. One other drawback is that FOCUS only understands commands that are typed in UPPERCASE. When using FOCUS interactively, it converts all commands into UPPERCASE automatically. FOCUS can store data in upper/lowercase. However, the default is to store data in uppercase.

5.4. Oracle

ORACLE from Oracle Corporation was the first DBMS to fully implement the Structured Query Language (SQL) proposed by IBM for relational data base management systems, which has become the standard query language for such dbms's. At LBL, ORACLE runs on the full range of VAX computers under both the VMS and ULTRIX operating systems, on IBM-PC's (albeit with expanded memory requirements) and SUN systems. The major host language interface is through the C language.

The main Oracle application currently running at LBL is the Particle Data Group's set of special databases which support its various publications.

Strengths. ORACLE's major claim to fame is its portability across many hardware/software environments. Data base files exported from any of these ORACLE implementations may be loaded without change into ORACLE on any other machine. The same is true for ORACLE's forms interface. ORACLE permits users to dynamically modify many aspects of a database (e.g., add a new column) without reloading data. ORACLE Corp is the largest independent vendor of relational dbms software.

Weaknesses. It is only at the host language interface level that ORACLE portability begins to break down, with a FORTRAN call interface not available for the IBM-PC or for many minor hardware implementations (e.g. UNISYS unix minicomputers). ORACLE's forms entry system is a cumbersome question and answer program to create entry screens. Modifying forma is difficult.

5.5. SPIRES

SPIRES is a DBMS originally developed at Stanford University in the 1970's for large bibliographic applications and is currently used for most of Stanford's administrative databases. It currently runs on IBM 370 mainframes under MVS, TSO, and CMS at several dozen academic and government installations in the U.S., Canada, and Great Britain. A C version for VMS and Unix is currently under development.

LBL users currently use SPIRES on the U.C. Berkeley campus 370 system via LBLNet. There are about a dozen SPIRES database applications at LBL currently in production, including the employee database used to produce online and printed telephone directories, LBL Reports, Mechanical Engineering drawings and notes, SSC drawings and addresses, a prototype material properties data system, and an online Account Authorization system.

Strengths. SPIRES has proven to be a very versatile dbms that has been applied to very diverse applications. It has a number of features that make it particularly suited for

text and complex data structures. It does not require an over-all Database Administrator; each database owner defines and maintains his own database. Many forms of security are provided by SPIRES at both physical and logical levels. It also provides a rich variety of mechanisms for automatic indexing and passing information between different files and record-types, connection of related databases, integrity constraint specification, and custom environment development. SPIRES was one of the first systems to provide access via an autonomous server process, so that remote users can access databases automatically via Email without necessarily having an account on the host machine.

Weaknesses. SPIRES currently runs only on IBM mainframes. Because it is an extensive system with many features, the early part of the learning curve for users is steep.

5.6. 4th Dimension

4th Dimension runs on Apple Macintosh Plus, SE, and II machines under System 4.1 or higher and Finder 5.4 or higher. It combines the Macintosh graphical interface with a structured programming language and comes up with an amazingly versatile DBMS. A relatively new product, 4th Dimension is finding uses at LBL in applications as varied as personnel-payroll administration, a hyper-text like front end to a material properties data base, job tracking, a travel-records administration system, and a space utilization system.

Strengths. 4th Dimension is strongest in handling multiply-occurring elements and in integrating multi-table elements on single data entry and/or display screens.

Weaknesses. 4th Dimension lends itself to the development of user friendly, fully Mac-like custom applications, but it is not so friendly to the developer--a fact which may confuse those who expect the development of friendly applications to be friendly too. Developers must master several graphics-based editors and the programming language. Interestingly, Mac aficionados are put off by 4th Dimension's programming language, and more traditional programmers might be put off by its graphical aspects.

5.7. Other Data Management Software

LBL scientists and other staff also use a number of other database management systems that this study did not address in any detail. Some of these are described briefly below. We have undoubtedly overlooked others.

5.7.1. FILEMAKER II

FileMaker Plus is an excellent flat-file data management program for the Macintosh. It has no relational capabilities, no programming language, no customizable menus or buttons, no very complex searching capabilities. It does have excellent layout, file definition, sort, and import-export facilities and is extremely easy to learn, compared with more full-featured DBMS. It has built-in facilities for doing mailing labels and has a text export facility that creates files formatted for Microsoft Word merge printing. FileMaker is the program of choice at the Laboratory for mail list management and similar simple data organization tasks.

5.7.2. Informix

Informix is a full-featured relational system that runs on different mini and mainframe systems, including Unix. It is less expensive than Ingres, Oracle, or Sybase and boasts a large number of third-party applications. It is currently being used by the Advanced Development Group in conjunction with several projects that address data and program sharing.

5.7.3. Ingres

Ingres is a relational system which runs on a variety of hardware and operating systems -- though not quite as wide a variety as Oracle. It is based on fixed length, rather than variable length fields. Users currently cannot add columns to an existing table without unloading and reloading the data already in that table. It has a more fully developed and integrated forms management subsystem than Oracle. Ingres also has the most fully developed set of features for distributed databases of any commercial product. Ingres is currently being used by the Earth Sciences Division for a seismic database and by the Human Genome Center for a database of digital images.

5.7.4. Paradox for DOS, Paradox for OS/2

A full featured data base application system, Paradox packs a lot of power for developers of complex applications. While this product is not widely used at LBL, it is a major DBMS for IBM and compatible microcomputers. The Workstation Group has a copy for user evaluation.

5.7.5. Reflex

Reflex on the IBM PC family is a single file DBMS. It uses pull-down menus or commands for almost everything it does. Some menus have dialog boxes. Reflex can be used with or without a mouse. Screens are nicely laid out. Establishing and maintaining a simple database is very easy. Many tools are available - e.g. nested sorting, filters, global edit capabilities, report formatting, a number of different "views", graphs, cross-tabulation facilities, etc. Reflex is very fast. The drawbacks are that the size of a database is limited to the amount of memory on the machine. The entire database is loaded into memory - whence the speed. Another drawback is that Reflex will only handle one file. PC Reflex is not a relational system. It does not support lookup tables, etc. Reflex on the MAC is a relational DBMS. It will work with more than one file.

5.7.6. Sybase

Sybase, one of the newest entries in the relational marketplace, competes with Ingres and Oracle on midsize computers. It also comes as part of the standard software on NeXT computers. The primary niche for which Sybase was designed is transaction processing -- i.e., ability to process hundreds of simultaneous transactions very quickly. It was the first relational system to support "triggers," a very general mechanism that can be used to execute user-defined processes in conjunction with read or write access to specified data fields (analogous to "rules" in SPIRES). It also was one of the first to support scientific functions (sine, cosine, etc.) and binary large objects (BLOBS). LBL runs Sybase on its NeXT machines and has recently acquired a SUN file-server version for use in the Human Genome Project.

6. Glossary of Generic and System-specific Database Terms

NOTE: ABBREVIATIONS AND SOURCE OF DEFINITIONS:[1]

4D	Fourth Dimension
DTR	Datatrieve
FOC	FOCUS
OR	Oracle
SPI	SPIRES
dB	dBase
Auerbach:	Auerbach Reports Glossary, 1984.
WIS:	WWMCCS Information System Data Standards & Data Standardization Procedures.
ANSI CCLOIIR:	Proposed American National Standard for Information Sciences Common Command Language for Online Interactive Information Retrieval.

ALIAS: A data element having the same definition and structure as a standard data element but with a different name. It is a synonym. (WIS) An alternate label used to refer to a data element. (Auerbach).

- 4D No provision.
- DTR Query name.
- FOC One alias allowed in file description. Additional aliases may be established using DEFINE and LET commands.
- OR Used as defined.
- SPI An alternate name for a data element.
- dB Used as alternate label at file level only.

ATTRIBUTE: Property that can assume values of entities or relationships. Entities can be assigned several attributes (e.g., a tuple in a relation consists of values). Some systems also allow relationships to have attributes. (Auerbach) A definitive characteristic of a data element or data item which quantifies, identifies or describes its representational, administrative, or relational concept . (WIS) Data field. Data element.

- FOC Refers to security attributes, video display, etc.
- OR Column.
- SPI Length, type, occurrences, other processing rules. Data element.
- dB Refers to video display only.

AUDIT: Record of activity during data base system operation. May be record of changes and/or record of actions.

- DTR History List: record of processing.
- FOC Logging: record of processing. Dialog Manager: record of commands.
- OR Audit: audit of system use. After Image Journaling: record of changes.
- SPI Command Logging: record of commands for each user of a specified data base. Many levels of logging available. Subfile logging.
- dB History, Do History: retains commands in memory for recall or printing.

---

[1] See bibliography for citations.

BINARY ELEMENT: A constituent element of data that takes either of two values or states usually either true or false or zero or one. (Auerbach).
- SPI One possible type of element. Type = logical.
- dB Logical field type.

BROWSE: Display of selected group of records and/or selected fields.
- SPI Used to browse an index (i.e., an inverted list with pointers). Display small subset of values from a specified index; may be centered around a specified value or drawn arbitrarily from the index.
- dB Allows full-screen editing, record addition, and deletion during browse.

CLASS (of entities): All entities held by a given proposition (conditional statement).
- 4D Has commands to create, modify, save, and load classes of records, which are called SETS.
- SPI A set of records may be created in a variety of ways. Can manipulate a set of records (add, delete, update).
- dB SET FILTER.

COLUMN: A vertical table cut in which values are selected from the same domain. The column is named in the heads. (Auerbach) Single attribute in a relational table.
- OR Used as defined.
- SPI Data element.
- dB Field.

COMMAND LANGUAGE: User interface to data base for retrieval, update, display, etc.
- 4D Menu driven User Environment.
- DTR Used as defined.
- FOC Dialog manager.
- OR QUERY.
- SPI Searching and updating language. File maintenance language.
- dB Used as defined. Also has menu-driven command interface, "The Assistant".

COMPLEX INDEX: Index keyed on derived data element.
- 4D No provision.
- SPI Compound index.
- dB Allows any derivation except mixed data types. Allows converted data types.

DATA DEFINITION: The statement of the data entities, their attributes, and their relationships in a coherent data base structure to create a schema. (Auerbach).
- 4D Data structure, which is a list of fields and their attributes, plus a schematic of tables and their relationships, plus layouts and layout procedures, which define input/output specifications.
- FOC Master File Description.
- OR Schema. Table table. Column table.
- SPI File Definition.
- dB File structure plus filters, indexes, and data input/output specifications.

DATA ELEMENT: A basic unit of information having a unique meaning and sub-categories (data items) of distinct units or values. (WIS).

- 4D Field is the basic unit, within which there may be sub-files which contain subrecords.
- FOC Field.
- OR Column
- SPI Spires uses term element or data element rather than field. Spires "elements" have no spatial connotation for storage, deliberately.
- dB Field is the basic unit of data. There are no sub-units.

DATA ITEM: Unit within data element.

- 4D Sub-file.
- DTR Elementary Field.
- FOC None.
- OR None
- dB None.

DERIVED DATA ELEMENT; VIRTUAL ELEMENT: A data element whose value is generated as a result of the operation of an application. (WIS) A data element that is not necessarily stored but that can be generated when needed. (Auerbach).

- 4D Non-enterable field or variable.
- DTR Global variables, local variables, new fields.
- FOC Temporary Field. Global/local variables. Defined and computed fields.
- OR Temporary data field. Pseudo-column.
- SPI Static variable, dynamic variable, virtual elements. Phantom elements, phantom structures, redefined elements, etc. Dynamic element.
- dB Memory variable, public and private.

DERIVED RELATION: A relation that can be obtained from previously defined relations by applying some sequence of retrieval and derivation operators (e.g., a table that is the join of others plus some projections.) (Auerbach).

- SPI Phantom structure.

DOMAIN: Set of legal values from which actual values are derived for a given attribute or data element (Auerbach). A set of all data element values from which a data item is drawn. (WIS).

- 4D Not supported directly, but validation can be done with input layout procedures.
- DTR Refers to file.
- FOC Limited support in schema.
- OR Supported with triggers in the forms entry process.
- SPI Not supported directly, but validation for admissible values can be done via inprocs, searchprocs, and userprocs (processing rules).
- dB Not supported directly, but validation can be done with input procedures.

EDIT MASK: Format template for control of data element display on screen or printer; also may be basis for data validation.

- 4D Not defined in data structure, but can be defined in layout; numerous date and number formats, text alignments, and so forth.
- DTR PICTURE clause in data definition.
- FOC Format in schema definition. Edit masks in dialog manager and in TABLE.
- OR Format model.

SPI Format refers to report formats. Uses edit mask in standard way. Standard usage. Available as inproc, outproc, searchproc, or function call in protocols language.
dB Template following PICTURE clause in screen or report format.

ENTITY: Person, place, or thing of interest.

FIELD/ELEMENT: As a subset of a record, a set of alphanumeric characters or other structured data treated as a unit and used to store a particular kind of data. (Proposed ANSI CCLOIIR).
4D Field may contain sub-file(s) of repeating sub-records.
DTR Field may contain subfields. Group field may contain other fields.
FOC Used as defined.
OR Column. Field also means data entry unit of input form.
SPI Uses term element; structure elements may contain other elements. Spires has data elements and structure elements. Only data elements contain data. Structure elements contain other (data and/ or structure) elements.
dB Used as defined.

FILE: Set of records treated as a unit and stored under a single, logical file name. (Auerbach) An organized collection of data, usually comprising related records. (Proposed ANSI CCLOIIR).
4D 4D does not store data tables as separate files; rather a data base of related tables are stored together in a collection of physical files: data, layouts, indices, procedures, and so forth. The files are organized into a folder. 4D takes care of file management automatically, and the prudent user doesn't interfere.
DTR File holds logically related records. Term also applies to physical storage .
FOC Used as defined.
OR Partition .
SPI Uses term subfile. A Spires FILE can contain 1 or more subfiles. Each subfile can have 1 or more record types, etc. Two meanings: (1) the physical files (2) the logical set of all the record types defined by a single file definition. Subfile means access to a particular record type. Record types contain records goal or index records.
dB Used as defined for data file. Separate files hold indices, memos, views, memory variable listings, and so forth.

FORMAT: The arrangement or layout of data in or on a data medium (i.e., buffer) or in a program definition. (Auerbach).
4D Layouts govern display of data input and output. May be governed by procedures attached to the layouts.
SPI Input format: used to load data from a physical file or the terminal into Spires. Output format: used to display or print database records from internal Spires to an eye-readable layout. Format specifications for display of one or more record types on an output device (CRT or printer). OUTPROC: output processing rule for transforming the value of a data element from internal to external representation.

FULL SCREEN EDITING: Presentation of logical unit records from one or more record types (relations) for operator editing of displayed fields.
4D Uses Macintosh user interface, which includes support for full-screen I/O layouts.
DTR Not available.

FOC Uses TED (XEDIT lookalike). See also FORM.
SPI Uses XEDIT. Input format can be defined via SCREEN DEFINER.
dB Managed by system in crude way with no edit masks or range checking. User defined screen form required for polished full screen editing.

HIERARCHICAL MODEL: Tree structured model of data relationship. Spires, Datatrieve, Focus. 4D provides some hierarchical structuring.

I/O FORM.
4D Layouts are used for all I/O; interactive and reporting/display.
FOC CRTFORM used for interactive I/O screen operations.

INDEX: The portion of the storage structure maintained to provide efficient access to a record when its index key item value is known. (Auerbach) A means of determining the location of data in a file; contains words or phrases by which a record in the file may be identified and retrieved. (Proposed ANSI CCLOIIR).

INVERTED INDEX FILE: An index organized by a nonunique key to speed the search for data in a previously unspecified manner. (Auerbach).
SPI Index specified by unique key to allow search of data in manner previously specified.

JOIN: An operation that takes two relations as operands and produces a new relation by concatenating the tuples and matching the corresponding columns when a stated condition holds between the two. (Auerbach) May be physical or logical.
4D No provision for physical join (though could probably write a procedure to achieve such a thing). Link: a logical join. The effect of a logical join can also be achieved in layout procedures.
DTR Cross.
FOC Join: logical join. Match: physical join.
OR Join column is term for the common key. Cluster join: Physical join.
SPI Available only by effort of the file definition. Not interactively by the user.
dB Join: physical join, creates new data base file. Select & set relation: logical join.

KEY: Value used in index to identify row (record, tuple) from which it came.
4D Multiple index fields supported. Indexed fields need not be unique.
FOC Key refers to the key value within RMS/ISAM files. Uses SEGTYPE sequence key for storage and retrieval. Uses CRKEY (cross reference key) for linking files. Uses FIELDTYPE = I to index a field. The entire value of "field" is indexed.
SPI The unique identifier of any goal record or index record.

LOGICAL FIELD TYPE: See entry for BINARY ELEMENT.

MACRO: A sequence or package of commands to perform a multi-step process. (Proposed ANSI CCLOIIR).
4D Global procedure can be called from a menu or by any other procedure and used as a macro.
FOC Files called focexecs or on VMS system FEXes.
OR Function key assignment.

SPI Processing rules (inproc, outproc, passprocs, search procs, user procs) and Protocols language. Protocol: sequence of command language statements.
dB Uses term in reference to substitution of variable into string to create label. Sets of commands can be stored in procedure files for use similar to macros.

MULTIPLY OCCURRING: Data element which can contain more than one value (e.g., a vector). Number of occurrences may be fixed or variable. Related item: repeating group.
4D A field of type "sub-file" contains "sub-records" of one or more fields. Number of occurrence of sub-records is not fixed.
DTR Group field.
SPI Elements can: not occur, occur one or more times. An occurrence of an element need not have a value. Elements can: not occur, occur with a null value, occur with a value Number of occurrences can be specified (1,2,..etc.). Otherwise number of occurrences may vary from record to record.

NAVIGATING: Steering a course through a data base by using such devices as indexes and pointers to arrive at and examine a record and data item values. (Auerbach).
FOC Nexting.
OR Walking the tree.

NETWORK MODEL: A data model that provides data relationships based on records and that groups records into so-called sets in which one record is designated as the set owner and a single member record can belong to one or more owner relationships. (Auerbach).
SPI Can represent networks directly via multiply occurring pointers.

PARAMETER: An elementary data item or array of data items that specifies the data type of its values and assumes or supplies the value(s) of the corresponding argument in the call of a procedure. (Auerbach).
4D Used as defined.
FOC Variables.
OR Used as defined.
SPI Used as defined.
dB Used as defined.

PRIMARY KEY: An item whose value uniquely identifies a record or tuple. (Auerbach).
4D Indexed fields can be specified unique, but need not be.
FOC Keys are available. User defined. Need not be unique. Need not have any key.
SPI Each record type must have a designated key element, whose values must be unique.
dB No provision for unique key.

PROCEDURE: Set of commands stored as named entity which may be invoked directly or called from another procedure.
4D Used as defined. Procedures may be attached to layouts, to files, or to a data base as a whole.
DTR Used as defined.
FOC Focexec (FEX).
OR Collections of SQL statements can be invoked by triggers in SQL-FORMS
SPI Procs Protocol.
dB Used as defined.

PROGRAMMING LANGUAGE: 4GL; command sets stored and recalled by name; usually have branching, looping capabilities in addition to command or query language repertoire.

- 4D Procedures.
- DTR Indirect Command File.
- FOC Dialog manager, modify, table, scan comprise programming language. This programming language can be executed interactively or stored in executable files called Focexecs.
- SPI Protocol. File definition language Formats language.
- dB Command file.

PUBLIC.

- 4D The term refers to variable whose value is available to all procedures of a data base. There are in addition various levels of password protection; non-protected layouts and procedures are public.
- OR Object (table, view, etc.) available or visible to all users.
- SPI Level of security assigned to a database, format, etc., which makes it accessible to all users (but which may be restricted to read-only).
- dB Variable whose value is available to all levels and procedures of a command file.

QUERY LANGUAGE: A language that enables a user to interact directly with a DBMS to retrieve and possibly modify its data. (Auerbach).

RECORD: Aggregation of values of data items or elements. (Auerbach) Row or tuple.

- 4D Prefers term record.
- FOC Prefers term record. Segment is a logical structure/group with record.
- OR Prefers term row.
- SPI A goal record or index record is a collection of data element values with a unique key.
- dB Prefers term record.

RELATIONAL MODEL: A data model allowing the expression of relationships among data elements as mathematical relations. The relation is a table of data representing occurrences of the relationship as rows. (Auerbach) Model which organizes data into tables consisting of one or more units (rows) each containing the same set of data elements (columns). dBase, Oracle, Focus vendors describe these systems as relational. In general, the term is applied to all microcomputer DBMS that are not hierarchical or network; the term originally had a more restricted meaning.

REPEATING GROUP: A collection of data that can occur several times within a given record occurrence. (Auerbach).

- 4D Sub-files can contain multiple sub-records associated with a single parent record.
- DTR List. Repetition of field or group of fields using the OCCURS clause. Must specify number of repeats. GROUP FIELD.
- FOC Segment.
- OR Not supported.
- SPI Same as multiply-occurring (whether a single data element or a multiply-occurring structure).
- dB None.

ROW: A nonempty sequence of values in a table and the smallest unit of data that can be stored into and erased from a table. (Auerbach).

SCHEMA: A complete description of the data base in terms of the data characteristics and the implicit and explicit relationships between data types. (Auerbach) A conceptual description of a data base. (WIS).

- 4D No such entity. A graphical presentation of files, linked files, and sub files, plus print- out of records and field attributes, plus layouts and layout procedures all combine to serve the function of a schema.
- FOC Master file description.
- SPI File Definition (output formats, command procedures, etc. are stored separately).
- dB No such entity. File structure plus any relations and input screen range specifications have to be combined to achieve the function of a schema.

SCREEN DEFINITION: Screen painter: full screen operator interface.

- 4D Layout editor in the design environment.
- FOC FIDEL. Also has a screen painter.
- OR Form.
- SPI Screen definer, formats.
- dB Create screen: .frm files.

SET: A number of distinct objects with a membership criteria. In a network or CODASYL-type of data base, a set is a named logical relationship between record types, consisting of one owner record type, one or more member record types, and a prescribed order among the instances of member records. (Auerbach).

TABLE: A relation that consists of a set of columns with a heading and a set of rows (i.e., tuples). (Auerbach).

- 4D File. One table per file; many files per database. (File here does not mean that tables are stored in separate physical files.)
- FOC Table refers to report generation facilities/ language.
- OR Table. Used as defined.
- SPI Format $REPORT: tables can be defined by the user "on the fly". Subfile, record type.
- dB File. One table per data file.

VARIABLE LENGTH FIELD: Variable length (text) field in otherwise fixed-length data base system.

- 4D Text field.
- OR Long field.
- SPI Any field may be declared variable length or fixed length.
- dB Memo field.

VERSIONING: Retention of previous versions of file(s) or other aspects of a database.

VIEW Derived relation (using operations such as join, project, etc.)

- 4D Used as defined.
- DTR Collection.
- FOC Alternate file view. Screening specifications. Dynamic and static joins available.
- OR Used as defined.
- SPI Does not use the term view, but analogous functionality is available via formats, index linkages, virtual elements, and phantom structures.
- dB Use of the term is not standard. "View" in dBase refers to a materialized view -- i.e., an actual physical file which contains the desired relation between two or more files, filter statements, indexes, and so forth.

VIRTUAL ELEMENT: See derived data element.

VIRTUAL RELATION: See derived relation, view.

WILD CARD: Symbolizing unknown or unspecified characters in a search term by special characters defined to represent any character (also known as truncation). (Proposed ANSI CCLOIIR).

- FOC Uses EDIT mask function.
- OR Pattern.
- SPI Truncation character used in index searching, e.g., Find Jones or Fine Jon#. Partial string matching.
- dB Skeleton: used for file names and memory variables only.

7. Further Reading

The following list contains references to selected publications that participants in the *ad hoc* Data Management Resource Group have found useful. It is far from comprehensive, but it does suggest the range of publications that are available on the subject of data management, from short articles in the micro-computer trade press to full-length academic monographs.

7.1. Brief Introductions

Auerbach, *Data Base Management*, p. 8, Auerbach Publishers Inc., Pennsauken, NJ, 1984.

A small (8 page) dictionary of database management terms.

Finkelstein, Richard, "Lingua Franca for Databases," PC Tech Journal, vol. 5, no. 12, p. 52, December 1987.

With IBM's backing the nonprocedural Structured Query Language or SLQ, is on its way to providing a universal langauage that allows different databases to communicate. SQL-based data managers are migrating from mainframe to PC in a variety of dialects.

Lynch, Clifford A., *Developments in Database Mangement System Technology and Their Impact on Information Retrieval*, Division of Library Automation, University of California, Oakland, CA.

O'Connell, Mike, *Data Base Management: What's It All About*, Digital Equipment Corporation, 1975.

A concise (34 page) though somewhat dated statement of the basic issues from the perspective of file access methods, with emphasis on the CODASYL (network) model.

Sandberg, G. "A Primer on Relational Data Base Concepts," *IBM Systems Journal*, Vol. 20, No. 1, 1981 pp.23-39

Seiter, Charles "Data Basics," *MacWorld* June 1988, pp.136-142

Stanford University. Center for Information Technology, A Guide to Data Base Development: A SPIRES Primer, Center for Information Technology, Stanford, CA, 1983.

Discusses general database issues from the standpoint of user requirements, as well as how SPIRES addresses certain classes of database requirements.

Stonebraker, Michael, *Future Trends in Data Base Systems*, in Proceedings of the Fourth International Conference on Data Engineering, Los Angeles, 1988, pp. 222-231.

A trenchant critique of SQL as a standard. There are already different dialects of SQL, and the majority of database application code will not be written in SQL. Users are still locked into vendor-specific solutions because of "4th Generation" languages, full-screen interface tools, and so on

7.2. Full Length Texts

Banet, Bernard A., Judith R. Davis, and Ronni T. Marshak, *Data Base Management Systems: the Desk-top Generation*, The Seybold series on professional computing. A Byte book., p. viii, 199, McGraw-Hill, New York, 1985.

Includes index. Microcomputers.

Cardenas, Alfonso F., *Data Base Management Systems*, p. xix, 745, Allyn and Bacon, Boston, 1985.

Includes bibliographical references and index.

LBL Bldg 50 QA76.9.D3 C37 1979

Date, C. J., *A Guide to the SQL Standard: A User's Guide to the Standard Relational Language SQL*, p. xiv, 205, AddisonWesley Pub. Co., Reading, Mass., 1987.

Includes index. Bibliography: p. 199-201. SQL (Computer program language)

UCB Engin QA76.9D3 D36951 1987

Date, C.J., *An Introduction to Database Systems*, Addision Wesley systems programming series, Addison-Wesley Pub. Co., Reading, Mass., 1986-.

Classic survey of the field, with special emphasis on the relational (System R), hierarchical (IMS), and network (CODASYL) approaches. Includes bibliographies and index

UCB Bus/SS QA76.9 D3 D371 1986 v.1

UCB Engin Reserve QA76.9 D3 D37 1986

UCB LibSchLib QA76.9 D3 D37 1986 v.1

Date, C. J., *Relational Database: Selected Writings*, p. xiv,497, Addison-Wesley, Reading, Mass., 1986.

Includes bibliographical references and index.

UCB Engin QA76.9.D3 D37241 1986 Reserve

IEEE Standard Glossary of Computer Applications Terminology, Institute of Electrical and Electronics Engineers, New York, 1987.

This glossary defines terms in the field of Computer Applications. Topics covered include automated language processing, business data processing, computer-aided design and manufacturing, control systems, medical appications, office automation, personal computing, and telecommunications applications.

LBL Library

Loomis, Mary E. S., *The Database Book,* p. xxiv, 465, Macmillan London : Collier Macmillan, New York, 1987.

Includes bibliographies and index. System design.

Maier, David, *The Theory of Relational Databases,* Computer Science Press, Rockville, Maryland, 1983.

Not for the novice. The author discusses relations, relation schemes, relational operators, joins, set theory, relational algebra, dependencies, normal forms, etc. This is an excellent, advanced and technical introduction to relational database systems.

LBL Bldg 50 *QA76.9.D3 M33 1983*

UCB Engin *QA76.9.D3 M33 1983*

UCB Math/Stat *QA76.9.D3 M33 1983*

Martin, James, *Computer Data-Base Organization,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1977.

An extremely readable classic, with lots of excellent diagrams and schematic representation.

UCB Engin *QA76.9.D3.M361 1977*

UCB LibSchLib *QA76.9.D3.M361 1977*

UCB Math/Stat *QA76.9.D3.M361 1977*

Weiderholdt, Gio, *Database Design,* McGraw-Hill, 1983.

Another classic text.

7.3. Journals and Series

Database Programming and Design, Miller Freeman Publications, 500 Howard St., San Francisco, (415) 397-1881.

Application oriented, short articles of somewhat uneven quality cover the mainframe to mini spectrum, from DBII to DBase. "Database Library" feature in each issue reviews relevant books.

DataPro Reports on Software (continuing series) DataPro Corp. Delran, NJ.

In depth reports on software products for DBMS and Applications Development, with management overviews, detailed features comparisons, and prices.

Software Digest Ratings Newsletter, One Winding Drive, Philadelphia, PA 19131-2903, (1-800-223-7093).

Rates Personal Computer Software, including database management systems. Also prepares special reports for clients.

7.4. Data Characterization & Logical Design

Blaha, Michael R., Premerlani, William J., Rumbaugh, James E. "Relational Database Design Using An Object-Oriented Methodology," *Communications of the ACM ,* Volume 31, Number 4, pp.414-427, Computing Practices, April 1988

Explains how object-oriented methods can be used to design relational systems.

Estvanik, Steve, "Logical Approach to Data Base Design," *Computer Language,* vol. 5, no. 1, pp. 49-58, Miller Freeman Publications, San Francisco, CA, January 1988.

Design guidelines for non-programmers and for microcomputer database implementors. As data bases become more widespread and more complex, the need to plan and design them carefully before implementation is increasingly important. This article looks at how one company used structured analysis to accomplish this task: by evaluating needs, organizing data, recognizing departmental crossovers, reorganizing data and creating the application.

Gillenson, Mark L., "The Duality of Database Structures and Design Techniques," *Communications of the ACM,* vol. 30, no. 12, pp. 1056-1065, Association for Computing Machinery, New York, NY, December 1987.

Attempting to pair database structures with database design techniques that seem incompatible yields some fascinating concepts about the world of database, including foreign keys and multiple relationships.

LBL Library

Ross, Steve, *How to Define Data to a Computer or Whaddayamean Specifications,* University

of Ottawa, Systems Development Services, 1983.

An irreverant primer that discusses what aspects of a client's data need to be examined in order to design an appropriate database.

Shlaer, Sally, Mellor, Stephen J. *Understanding Object-Oriented Analysis*, Project Technology, Inc. ©Hewlett Packard, 1989

An excellent text on object-oriented design.

Shoshani, A., Wong, H.K.T., and Olken, F., "Characteristics of Scientific Databases," in *Proceedings of the 10th International Conference on Large Databases*, Singapore, August 1984.

Distinguishes between "Experimental" and "Associated" data and identifies common characteristics shared by different types of scientific data, including identifiers, access patterns, database size, and useage. Includes Time Projection Chamber example and excellent table of examples by characteristics.

7.5. Product Comparisons

Department of Commerce, *A Guide to Performance Evaluation of Database Systems*, National Bureau of Standards Special Publication 500-118 (December, 1984)

One of a series of publications oriented to technical issues that arise with respect to procurements.

Quinn, John-Charles "Two Basic Tools Pave the Way to Choosing..." *Computerworld*, Special report pp.22-23, June 30, 1980

[a brief outline of formal methodologies for developing requirements and evaluating software in terms of those requirements]

Rothenstein, Philip I., Manger of Time-Sharing First Boston Corp. *Nine Questions to Ask When Selecting a Non-Procedural Database Mangement System*, Focus In Action, Information Builders, Inc. 9pages

7.5.1. Micro Computer System Evaluations

---, "Data Basics", *Macworld*, vol. 5 no. 6, pp. 136-143, PCW Communications Inc., San Francisco, June 1988.

Review of 7 mostly flat-file DBMS programs for the Mac.

---, "Databases for OS/2: The First Wave", *PC Magazine*, vol. 8 no. 11, pp. 94-131, Ziff Communications Co., New York, June 13, 1989.

Review of Paradox OS/2, Q&A OS/2 and R:BASE for OS/2 and discussion of OS/2 applications requirements in general.

---, "In Depth: Database Trends, *Byte*, vol. 14, no. 9, pp. 244-292, McGraw-Hill, Inc., September 1989.

Discussion of trends in distributed data management, server technology, and data models.

---, "Project Database 3: Programmable databases: dBASE and its challengers," "SQL: An emerging database standard for PC's", *PC Magazine*, vol. 7, no. 9, pp. 93-306, Ziff Communications Co., New York, May 17, 1988.

Review of 43 programmable systems and 5 SQL systems. Sidebars discuss most relevant PC DBMS issues.

---, "The Data Chase", *MacUser*, vol. 4 no. 12, pp. 159-185, Ziff Communications Co., New York, December 1988.

Review of 7 relational DBMS programs for the Mac.

Finkelstein, Richard and Fabian Pascal, "SQL Database Management Systems. *Byte*,"vol. 13, no. 1, pp. 111-118, McGraw-Hill, Peterborough, NH, January 1988.

Comparison of Informix-SQL, Ingress for PC, Oracle, SQLBase, XDBII, and XQL, all SQL systems for IBMPC/AT and compatibles.

LBL Library

Information Builders, Inc., *A Performance Benchmark: PC/FOCUS, dBase III, R:base 4000, R:base 5000*, Software Digest Inc., Philadelphia. 55 pp.

FOCUS reprint of tests conducted by National Software Testing Laboratories of Philadelphia. Shows code as well as times for each problem. (Same folks as Software Digest Ratings Newsletter).

Software Digest Inc., "Relational Database Management Programs," *Software Digest Ratings Newsletter*, vol. 3, no. 3, Software Digest Inc., Philadelphia, March 1986. 67 pp.

Very well done comparison of features and performance - but does not show code for each system or each test.

[Stanford University] PC Development Task Force, *Product Comparisons, update,* originally written in August 1987, update- June 1988

An in-depth comparison of selected products.

Urschel, William, "Tomorrow's Data Bases Today," *PC World,* vol. 6, no. 2, pp. 146-156, Ziff Communications Co., New York, February 1988.

Comparative evaluation of three database management systems for the PC which are built around SQL: Informix, Ingres, and Xdb.

LBL Library

7.5.2. Mainframe System Evaluations

A Datapro Feature Report *A Buyer's Guide to Data Base Management Systems,* DATAPRO RESEARCH CORPORTATION ©1977

An outdated example of the type of information that can be purchased from organizations such as Datapro that specialize in product comparisons.

A Comparative Analysis of Data Base Management Systems for Use in Medlars III. MITRE Corporation 81W00379 (July 1981).

A decade old, but still interesting in-depth comparison of mainframe systems for major bibliographic applications.

Request for Information on a Generalized Data Base Management System. Center for Information Technology, Stanford University (June, 1980).

One of the documents from a database management system selection process at Stanford some years ago, this study contains an extensive list of system attributes for evaluation, as well as vendor-specific issues.

7.6. System-specific References

Benne, Bart, *FOCUS: Developer's Handbook,* Computer Professional Series, Wordware Publishing, Inc., Dallas, TX, 1987.

This book is geared to the more advanced audience - computer professionals and advanced end-users. The topics it addresses are describing, building and structuring FOCUS files, various types of calculations, FOCUS executable programs to build and maintain files, select and sort records, produce reports from files, interact with the user, use the FOCUS text editor, produce graphs, use statistical tools, etc.

Date, C. J., *A guide to INGRES: a user's guide to the INGRES product (a relational database management system with built-in application development facilities) from Relational Technology Inc,* p. xiii, 385, Addison-Wesley Pub. Co., Reading, Mass., 1987.

Includes index. Bibliography: p. 377-378. INGRES (Computer system)

Jones, Edward, *dBASE III Plus Programmers' Reference Guide,* p. 430, Howard W. Sams & Co., Indianapolis, 1987.

A well-written, concise manual for fairly sophisticated dBASE users who want to extend their skills into writing command language files. Well-indexed, no bibliography.

7.7. Previous LBL Reports on Database Management Systems

Berkowitz, D., Bernzott, P., Borges, J., Fink, R., Franz, J., Fry, D., Johnston, J., Osterer, L. Lawrence Berkeley Laboratory Computer Center *Database Planning Group Position Paper* , September 29, 1980.

Summarizes procedures the LBL Computer Center used to identify and obtain a DBMS for LBL in 1980.

Gey, Fred and Wong, Harry T.K., "Codd's Twelve Rules for Relational DBMS Functionality" LBL Technical Report LBL-22202 (September, 1986) 18 pp.

A reorganization and explanation of features for data integrity, data manipulation, and data independence which should be supported by DBMS products in the 1990's.

Olken, Frank, "A DBMS Selection Checklist for the Army Corporate Database Project" LBL Technical Report LBL-22181 (October, 1986) 26pp.

A brief summary of selection criteria for a large administrative database application.

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
INFORMATION RESOURCES DEPARTMENT
1 CYCLOTRON ROAD
BERKELEY, CALIFORNIA 94720