Requirements for a generalized
data base management system

by A. C. PATTERSON

Bankers Trust Company
New York, New York

Several approaches to generalized data base management systems have been documented for the benefit of the data processing community. Perhaps the two most significant reports are Codasyl's Data Base Task Group Report issued in October, 1969, and revised in April, 1971, and the joint GUIDE/SHARE Data Base Management System Requirements. Each report brings with it its own distinct jargon to our already over-developed Tower of Babel. Although the reports often advocate similar solutions to the data management problem, there is a clear difference in emphasis and dominant philosophy. Perhaps the most immediately obvious difference between the two reports is that Codasyl has specified an actual COBOL syntax for the implementation of a data base management system (DBMS) and the GUIDE/SHARE report has simply defined the requirements the data processing community would impose on implementors of DBMS's.

The GUIDE/SHARE report is the result of work carried out over a twenty month period beginning in 1969 and culminating in the November, 1970, document. The committee was comprised of representatives from more than forty companies engaged in diverse activities from banking to armed forces, life insurance to machine manufacture, and government agency to university. It is reasonable to assume, then, that the requirements set forth by the committee are indeed representative of the requirements of the data processing community at large.

The GUIDE/SHARE committee was charged to:

Define the users functional requirements for a data base oriented system. The group does not intend to concern itself with implementation methodology at this time.

The document which summarizes the committee's thinking neither pretends nor intends to describe an existing system or one currently in development. The report is in no way intended as system specification. Some of the requirements may not even be implementable under the current hardware and software technologies. The report is rather intended to state a group of requirements the universe has placed on data base management systems of the future, both near and long range. The committee felt that to limit its thinking to concepts effectively realizable on currently available hardware and within the reasonably primitive software technology of today would be to limit the value of the work by assuring its obsolescence upon publication. Furthermore, it has been the intention to advance the state of the data base art by providing a forum where data base related concepts could germinate and burgeon.

Examine first the functions which are basic to a data base management system. First, there is a Data Base Descriptive Language (DBDL) which fulfills among others, the data definition function. The Data Base Manager (DBM) is the supervisory function carried out by hardware and software which coordinates all components of the system. The Data Base Command Language allows a user to make requests within the Data Base Management System. The Data Base Administrator is a function responsible for the definition, organization, protection, and efficiency of the data base(s) of an enterprise. It is a function performed by humans using, among other things, the facilities provided by the Data Base Management System. Hence, we have the components of the system: DBDL, DBM, DBCL and DBA. And, of course, there must be physical data to be manipulated, and that named collection of units of physical data which are related to each other in a specified manner is a data base.

The Data Base Management System has data independence as a primary design criterion. Data will be structured in physical storage in a logical and non-redundant manner, insofar as practical. As new processing requirements imply new structures, dynamic restructuring of the physical stores will be possible.
without impacting the current processing. Not only will 
“access” to data be simple, versatile, and secure, but its 
rules will be easy to learn and present few problems 
during its integration with a current system. It will 
achieve data independence for the user, data rela­
tibility, data non-redundancy, data integrity, security, 
and performance.

Data independence removes the burden of access 
strategy from the application programmer which frees 
him to do the job he does best—programming. The 
programmer should not concern himself with data 
formats or structures but rather with procedures and 
data manipulation. How and where the data are 
stored is to him immaterial. What relationships the 
data he is using have with other data which he is not 
using should not be his concern. The DBMS allows the 
programmer to concentrate on doing his procedural 
description well.

Data non-redundancy has several virtues which 
recommend it. Costly storage facilities should not be 
taxed with the responsibility for keeping duplicate data 
elements nor should a system be taxed with the main­
tenance responsibility for redundant data items. Where 
several different logical record occurrences have an 
identical data item value, a relationship should be 
maintained, where feasible, to assure that the mate­
rialization of the logical data item always is made from 
the same physical representation in the physical store. 
Duplicate data values in the physical store must be 
minimized or eliminated.

Through control facilities incorporated in the DBMS, 
data integrity will be provided. There must be a 
reasonable assurance that data and relationships are 
maintained accurately. For instance, this requirement 
includes the necessity for a facility to propagate changes 
to all derived data elements when a component of the 
derived data element is changed. If \( X, Y, \) and \( Z \) are 
data elements derived according to the following 
equations:

\[
X = A + B \\
Y = B \cdot C \\
Z = Y + D
\]

a change to \( B \) (which is a “real” data element as are 
\( A, C, \) and \( D \) also) effects a change to \( X, Y, \) and \( Z. \)
These changes must be accurately reflected not only in 
derived data element values, but also in any asso­
ciations the derived data elements enjoyed based on 
their values. A derived data element is a physically 
stored element derived through some algorithm from 
one or more data elements.

The DBMS will provide security for data based on 
installation specification. Security in the system is 
especially bidimensional: sensitivity of data, and 
authority of user or requestor. The given installation 
may place security restrictions on individual data items, 
specific combinations of data items (for instance, one 
user may be allowed to see salary averages, but only 
when more than five salaries are part of the computation 
of the average or he may be allowed to see age without 
name and vice versa), and of course restrictions may 
be placed at any higher, more general level of the DB 
such as the logical file level.

Because many of the ways in which the system 
performs may be specified by the installation, and 
because user exits to the DBMS will be provided at 
appropriate points, the DBA will be able to optimize 
the system’s performance. Consistent, measurable, and 
tunable performance of a system of this magnitude 
is a critical design level consideration. It is primarily 
the performance criteria which obviate the immediate 
wholesale implementation of the GUIDE/SHARE 
DBMS.

At this point, it is appropriate to examine in some 
detail the major components of the system, that is the 
DBA, DBDL, DBM, and DBCL. The requestor’s view 
of the DBMS is critical. He interfaces directly with two 
components—one is human (the DBA) and one is that 
hardware/software combination, the DBM. The re­
questor first communicates with the DBA to have data 
definitions entered into the Data Base Directory. The 
Data Base Directory (or DBD) is a part of the DB 
and contains all information pertaining to ac­
ceptibility of data. That information includes:

- physical to logical mappings (or physical record 
descriptions);
- logical to physical mappings (or logical record 
descriptions);
- relationships and associations among data elements 
and items;
- security and authorization required at the data 
level;
- miscellaneous information pertaining to the main­
tenance of and access to the data.

The DBA uses DBDL to enter data definitions and 
their associations and relationships. The language is 
non-procedural. It defines, states, or describes to the 
DBM how the DBM may operate on data. Once the 
definitions are entered in the DBD, the Requestor may 
proceed with his access to the DB.

Then, the Requestor communicates with the DBM 
using the DBCL. The Data Base Command Language, 
itslef non-procedural, states requests in a form that 
DBM can interpret. Included as primitive functions in 
the language are the approximate equivalents in current
data management systems of: (1) Open; (2) Retrieve; (3) Replace; (4) Add; (5) Delete; and (6) Close. A primitive command, once invoked, requires by definition no further intervention by the Requestor until its function has completed. Primitives operate only at the logical data level. Any physical ramifications resulting from the invocation of a DBCL primitive are created by DBM and remain unknown to the Requestor.

Available with each primitive command is a parameter list where the Requestor may supply further qualifications for his request. Among the items he may specify are: (1) a logical file name; (2) logical record name or names; (3) data area name; (4) a communications area where DBM and Requestor may transfer information to each other, such as status of request, exit routine addresses, and the like; (5) self-imposed data access restrictions to be used when DBCL primitives are executed. Where parameters are not specified in a given primitive, reasonable defaults will prevail. For instance, if no file name is provided in the “open”, SYSIN may be assumed for an input file and SYSOUT for an output file. System-provided default values may be replaced by particular installations at a time approximately equivalent to system generation (SYSGEN). Depending on the Requestor’s own authority or security clearance to access the DB, the sensitivity or security level of the data he wishes to access, and the defined relationships and associations of this data supported in the DBD, he may access the DB in any number of ways. Before the Requestor may begin accessing the DB, he must alert the DBM that he intends to use the system. This operation in DBMS is roughly equivalent to OPEN in current data management systems. The more information the Requestor provides DBM with at OPEN, the less information he will have to provide with each subsequent command. Also, the more DBM knows at OPEN about the kinds of access the Requestor intends, the greater the efficiency with which DBM can satisfy the requests in subsequent commands. In short, the longer specifications are deferred, the more it costs. The following are two of the ways of accessing data: (1) retrieve a record occurrence of a file based on record identifiers (something like keys in currently implemented data management systems); (2) record qualifiers. Clearly, the more description the Requestor provides to DBM through DBCL the greater the likelihood of a unique “hit”, or at least a reduced population of records which satisfy the retrieval criteria.

A few examples of the kinds of qualifications the Requestor might provide are: (a) record identifiers such as account number or part number; (b) qualifiers such as NEXT (nextness for a given logical file being described in the DBD). Other qualifiers available to the Requestor are previous, first, last, nth, a relationship name, a search procedure name where the procedure has been developed by the given installation, and Boolean expressions using AND, OR, NOT, BETWEEN, EQUAL TO, NOT EQUAL TO, GREATER THAN, and LESS THAN. Qualifications used for record searches may be predefined, named, and stored in the DBD or may be explicit with each command. Deletion, Replacement (that is, update), or Addition may be done only when the record being operated upon can be uniquely identified. That is, a record was retrieved based on certain qualifications, but more than one record in the file satisfies all the selection criteria, the record could not be updated (that is, replaced with modifications) or deleted. Only unique occurrences of logical records can be added to a file. The three categories of data used in almost any data base system, namely, the logical data, the entity, and the physical data, are also present in the GUIDE/SHARE DBMS. The largest unit of physical data is the Data Base. The most general reference to the concept of entity is the word itself. The DBM is that portion of the system (along with the DBA) which understands the entity concept. That is, DBM knows that both physical and logical data are organized around an entity or entity type. The entity is any person, place, thing, or event of interest to the enterprise. An entity type is a so-called classified entity, where an entity has been specifically identified as a particular type, such as employees, departments, vendors, tractors, sales, and so on. The entity construct is any association of entity record types and the entity relationships which connect them. When an entity construct has been named, it loses its general connotation to become an entity record type.

There are three types of or levels where relationships can exist—logical, entity, and physical. The physical and logical relationships are self-explanatory. The entity relationship is defined as a named relationship that exists between two entities of the same or different entity types, such as that between employees and departments. The physical correspondences for the logical file are data set, form extent, and space extent. The data set referred to here is roughly equivalent to the data set in current data management systems, or, more precisely, it is a named collection of stored records of one or more stored record types. A data set, in GUIDE/SHARE terminology, is a collection of one or more space and form extents. A space extent is a unit of contiguous space on some medium. A form extent is that portion of a data set in which all stored record occurrences are of the same stored record type—a physical subset of the physical file.

The logical record is the only record a program ever “sees.” It is made up of one or more data items which
are materialized from data elements found in one or more stored records, and are moved from physical to logical arrangements and vice versa, according to descriptive information and attributes defined in one entity record. This information is stored in the DBD along with other information the DBMS needs to access data, and there is therefore no tangible entity record or entity record file. The entity record type refers to a named record, that is, a collection of entity field names that represents the attributes of a particular entity type.

A logical record type is a specific collection of one or more data items. An occurrence of a logical record type is called a logical record.

The stored record type bears the same relationship to its physical counterparts as the logical record type bears to its logical counterparts.

A logical record occurrence is a particular instance of a specific group (one or more) of data items which make up a unique logical record. The stored record occurrence is similarly defined on the physical side.

The data item is a unit of logical data which may be either an elementary or group data item. Effectively, it is the smallest named unit of logical data. A similar definition applies to data element. The entity field is a generic term referring to either an elementary or group entity field. An elementary entity field is a named field referring to an attribute of an entity. The group entity field is simply a named association of entity fields within an entity record type.

An entity field type is a named entity field that represents some particular attribute of an entity type.

The data item type is a specific data item; such as an account balance data item, as the stored data element type is a specific data element. Data item occurrence and data element occurrence are specific representations of the values of units of data. Group data item and elementary data item fall under the generic term, data item, and are self-explanatory.

The last concept to be addressed in some detail is the Data Base Descriptive Language. The entity concept has been explained and the interfaces between and among the DBM, DBA, DBCL, User or Requestor, and physical and logical data. The DBDL discussion should clarify the DBA function and how the DBM knows what to do with data.

DBDL is primarily the instrument of the DBA function. This function uses the language to define to the system the necessary physical, entity, and logical descriptors associated with a Requestor's needs. Although the Requestor may define certain logical data locally and temporarily if those logical data are a subset of existing entity descriptors, only the DBA may describe physical data.

The DBA must describe data, define relationships, define mappings, define security and specify performance measurements. These functions may be fulfilled using the DBDL. DBDL statements are interpreted by the DBM and the derived descriptors are stored in a directory (DBD) which is maintained as part of the Data Base. This, then, is the major supporting element of data independence.

The DBDL provides the DBA with the means to:

- Describe the physical and logical characteristics of the data base.
- Describe the relationships which exist between the various components of the data base.
- Describe the rules by which the DBM resolves the differences between the physical data and the logical data (processed by the application program).
- Describe the rules by which the DBM performs security and integrity checks to prevent unauthorized and destructive access to the data base.
- Subset the DBMS and organize the data base to control costs while optimizing performance.
- Monitor and control the day-to-day operation of the DBMS.

There are some constraints placed on the DBDL. First, it must be an independent language and not an extension of current host languages. It must be an extensible language, or one which will provide the primitive functions which can be combined and expanded to fulfill new requirements as they evolve. DBDL is a descriptive and not a procedural language. The DBDL is, however, capable of identifying procedure written in a procedural language by the user which is to be invoked in particular instances, but DBDL cannot describe the procedure itself.

Appropriate defaults are provided in DBDL but with the facility to restate defaults or to override them at a number of different times in the course of processing the DBMS. The facility to name units of data on all levels (including field, group, file, data set) and define relationships allows the DBD to contain more than one version of each descriptor and define the conditions under which each is to be used. Alias names are supported as well as names with degrees of qualification, or indexed names. This latter facility is provided to reduce ambiguity.

DBDL will define identifiers for records or so-called key fields. Alternate identifiers are provided for. Data attributes and alternative data attributes with rules for their uses are specified in DBDL.
Among its most important functions, the description and definition of logical data relationships in DBDL ranks high. These relationships may be among any number of levels of data (elementary, group, etc.), any number of levels of entity definition, and any number of units of data. The only relationships which the application programmer must be aware of are the logical ones between a logical record and its data items and between and among logical records.

The DBDL allows the DBA to define membership and association rules for files. It also allows the DBA to specify rules for additions and deletions, security requirements, propagation requirements, and the sequence in which logical records will be added to the file. Propagation requirements may include something like this: a man was in the personnel department and was on the salary committee. To be on the salary committee, he must also be in the personnel department. The man transfers to data processing and his department code is updated. He will automatically be removed from the salary committee if the DBA has specified the general rule via DBDL.

Repeating group and elementary data items (including the zero case) may be described in DBDL and interrelationships between and among the files may be defined. Mappings may be defined which are consistent with the host language.

It is through DBDL that the DBA defines entity fields, record types, constructs, and relationships along with the data and function attributes for each entity level. Real, virtual, and derived fields are described in DBDL.

Entity relationships may be defined in terms of entity fields, record types, constructs and any other appropriate entity relationships. There are a number of other facilities which the DBDL provides, especially regarding mappings, space use, indexes, planned data redundancy, security, integrity, performance and control, and so on. These other facilities are more fully described in the GUIDE/SHARE November, 1970, report.

The time frame envisioned for the implementation of a DBMS which satisfies the requirements in the GUIDE/SHARE report is about five years. However, there are in existence today some systems which satisfy some of the requirements. It is not anticipated that any implementor will fulfill all the requirements in a first release. Rather, a system reflecting all the requirements will begin by satisfying a subset of them and then evolve into the sophisticated system proposed by GUIDE/SHARE. The continually advancing hardware technology will, to a degree, determine how soon some of the proposed features will be made available.

Perhaps the most important idea that the data processing community must digest and accept is that there should be one standard philosophy applied to any DBMS development by any implementor. This philosophy must intelligently address the concepts of data independence, data integrity, data relatability, data non-redundance, as well as others. It remains to bring together the various groups which have worked on DBMS definition and to then develop a single, standard approach to data base—one which can be accepted by the data processing community at large. It will not be an easy task, but it must be done.

BIBLIOGRAPHY

1 Data description and access
   ACM SIGFIDET Workshop Proceedings November 1970
   Available through ACM

2 R M BALZER
   Dataless programming
   Conference Proceedings FJCC 1967

3 R G CANNING
   EDP Analyzer Vol 8 Nos 2 3 4 5 1970 Vol 9 Nos 5 6 1971

4 Report to the programming language committee
   CODASYL Data Base Task Group April 1971
   Available through ACM

5 A survey of generalized data base management systems
   CODASYL Systems Committee May 1969
   Available through ACM

6 Feature analysis of generalized data base management systems
   CODASYL Systems Committee May 1971
   Available through ACM

7 M E D'IMPERIO
   Information structures: Tools in problem solving
   Journal of ACM SIGFIDET Vol 1 No 2 December 1969

8 R W ENGLES
   A tutorial on data base organization
   IBM Ref TR00.2004 March 1970

9 Data base management system requirements
   Guide/Share
   Guide Secretary Distribution (GSD–23) January 1971
   A J Burris Secretary The Northern Trust Co Box N
   Chicago Illinois 60690
   Share Secretary Distribution (SSD–208) December 31
   1970 Share Secretary
   Share Inc Suite 750 25 Broadway New York New York 10004

APPENDIX

GLOSSARY

Access Method. A routine external to the application program that performs storage and retrieval of physical data.
Access Strategy. An algorithm by which stored records are identified and located.

AP. Application Program(s)

Association. A non-directed relationship that defines a collection of zero or more units of data based on some specified criteria. The occurrences of units of data within the collection may or may not be ordered.

Binding. The firm association of an attribute of data with the application program.

Checkpoint. The act of capturing the state of units of data and those application programs operating on them for the purpose of reconstructing the data and restarting the programs.

Command. A generic term referring to either a primitive command or a compound command. The DBCL facility through which the DBM functions are invoked by an application program.

Communication Area. That area embodied within an application program that permits communication between the application program and the DBM.

Compound Command. A DBCL command which is a combination of primitive commands and procedural logic.

Data Area. That area in the application program which contains the logical data.

Data Base. A named collection of units of physical data which are related to each other in a specified manner.

Data Base Administrator (DBA). A person or persons given the responsibility for the definition, organization, protection and efficiency of the data bases for an enterprise.

Data Base Command Language (DBCL). A language whose statements are used to invoke the DBM facilities.

Data Base Descriptive Language (DBDL). A language whose statements are used to describe all units of data to the DBMS.

Data Base Directory (DBD). A collection of descriptors of all units of data that are available to the DBMS. These descriptors are derived from DBDL statements.

Data Base Management System (DBMS). The data processing system consisting of the tri-partite interaction between the Requestor, the DBA, and the DBM.

Data Base Manager (DBM). A combination of hardware and software which controls and processes all requests for data in the data bases.

Data Element. The smallest named unit of physical data stored on some medium.

Data Independence. The concept of separating the definitions of logical and physical data such that application programs need not be dependent on where or how physical units of data are stored.

Data Integrity. The concept that all units of data must be protected against accidental or deliberate invalidation.

Data Item. A unit of logical data which can be either an elementary data item or a group data item.

Data Name. A name given to units of data for the purpose of uniquely identifying that unit of data.

Data Set. A named collection of stored records of one or more stored record types. More precisely, a collection of one or more space and form extents.

DB. Data Base(s).

DBA. Data Base Administrator.

DBCL. Data Base Command Language.

DBD. Data Base Directory.

DBDL. Data Base Descriptive Language.

DBM. Data Base Manager.

DBMS. Data Base Management System.

Derived Data Element. A data element whose value is derived from the values of other data elements by a specified algorithm.

Descriptors. The detailed definition of all units of data as represented in the DBD.

Elementary Data Item. The smallest named unit of logical data available to a program.

Elementary Entity Field. A named field which refers to an attribute of an entity.

Entity. A person, place, thing or event of interest to the enterprise.

Entity Construct. An association of entity record types and the entity relationships which connect them. An entity construct when named becomes another entity record type.

Entity Field. A generic term referring to either an elementary entity field or a group entity field.

Entity Record Type. A named collection of entity field names that represents the attributes unique to a particular entity type.

Entity Relationships. A named relationship that exists between two entities of the same or different entity types.

Entity Type. A particular kind of entity. For example: employees, departments, vendors, sales, etc.

Exclusive Control. A facility to prevent multiple concurrent interactions with a specific unit of data in the data base such that the integrity of the data is preserved.

File. Named collection of occurrences of logical records which may be of more than one logical record type.

Form Extent. That portion of a data set wherein all stored record occurrences are of the same stored record type.

Format. A formal description of a unit of data that
contains information about its length, base, scale, precision, representation, etc.

**Group Data Item.** A named association of data items.

**Group Entity Field.** A named association of entity fields within an entity record type.

**Hierarchy.** A set of directed relationships between two or more units of data; such that some units of data are considered owners while others are members. This is distinguished from a network in that in a hierarchy, each member can have one and only one owner.

**Host Language.** The programming language used to write the application program. A program written using this language contains DBCL statements that invoke DBM functions.

**Identifier.** A unit of data whose value uniquely identifies an occurrence of that unit of data or a different unit of data.

**Journal.** A record of all environmental conditions and changes relative to the data bank. It may include time and date stamps, user identification, attempted security breaches, changes to a data base, etc.

**Linkage.** A mechanism for connecting one unit of data to another.

**Logical Data.** That data which the application program presents to or receives from the DBM.

**Logical Record.** A collection of one or more data item values. More specifically, an occurrence of a logical record type.

**Logical Record Type.** A specific collection of one or more data items.

**Logical Relationship.** The relationship that exists between two units of logical data.

**Materialize.** The act of making logical data available to the application program.

**Member.** See membership

**Membership.** A directed relationship connecting one unit of data called the owner to another unit of data called the member.

**Network.** A set of directed relationships between two or more units of data such that some units of data are considered owners while others are members. Unlike a hierarchy, each member may have more than one owner.

**Occurrence.** A specific representation of the value of a unit of data that is usually associated with a value called its identifier.

**Own Code.** Specialized routines developed by an installation to perform functions not provided for by the Data Base Manager.

**Physical Data.** That data which the Data Base Manager stores on, or retrieves from, some medium.

**Physical Record.** That data accessed by the hardware from some medium with a single access.

**Physical Relationship.** The relationship that exists between two or more units of physical data.

**Primitive Command.** A DBCL command which can be executed without further intervention by the application program.

**Profile.** A subset of descriptors in the DBD that relates to something of interest to the DBA such as a program or a user.

**Qualifier.** Criteria used to select a logical record.

**Real Data.** A logical unit of data that has a physical unit of data as a counterpart.

**Redundancy.** A situation where there are multiple occurrences of a particular unit of data in a data base.

**Relationships.** A generic term referring in this document to either one of two kinds of relationships: membership and association.

**Requestor.** An individual desiring to use the data in the DBMS.

**Security.** The concept that units of data can be altered, viewed or processed only by users who have the proper authority and the “need to know”.

**Self Defining Data.** A unit of data whose description appears with its occurrence.

**Shared Control.** A facility that allows multiple concurrent interactions with a specific unit of data in a data base.

**Space Extent.** A unit of allocated space on some medium which is contiguous.

**Span of Control.** That period of time during which the application program has exclusive or shared control of a specified unit of data.

**Stored Record.** A collection of one or more data element values. More specifically an occurrence of a stored record type.

**Stored Record Type.** A specific collection of one or more data elements.

**Structure.** A generic term which refers to the aggregation of units of data, their formats and their relationships.

**Unit of Data.** A generic term denoting a named conceptualization of logical data and physical data. The units of logical data are:

- data item
- logical record
- logical relationship
- file

The units of physical data are:

- data element
- stored record
- physical record
- physical relationship
- data base
- form extent
- space extent
- data set

*User.* A generic term referring to the Requestor or the DBA.

*User Exit.* A DBM facility that permits an installation to execute its own code.

*Virtual Data.* A logical unit of data that is materialized and does not exist as a physical unit of data.