Lecture Notes in Computer Science

Edited by G. Goos and J. Hartmanis

132

Data Base Design Techniques I:

Requirements and Logical Structures
NYU Symposium, New York, May 1978

Edited by S.B. Yao, S.B. Navathe,
J.L. Weldon, and T.L. Kunii

Springer-Verlag
Berlin Heidelberg New York 1982

Editors

S.B. Yao
University of Maryland
College of Business & Management
and Dept. of Computer Science
College Park, MD 20742, USA

S.B. Navathe
Computer & Information Sciences Dept.
University of Florida
Gainsville, FL 32611, USA

J.L. Weldon
New York University
Graduate School of Business Administration
90 Trinity Place, New York, NY 10006, USA

T.L. Kunii
Dept. of Information Science
University of Tokyo
Hongo, Tokyo 113, Japan

CR Subject Classifications (1979): 3.72, 3.73, 3.74, 4.33

ISBN 3-540-11214-6 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-11214-6 Springer-Verlag New York Heidelberg Berlin

Printing and binding: Beltz Offsetdruck, Hemsbach/Bergstr.
2145/3140-543210

PREFACE

The design of data base organizations is one of the most important steps in the development of a computerized information system. Size and complexity combine to make this task disproportionately time consuming and expensive. In the past, database design activities consisted of trial and error approaches using ad hoc techniques: systematic method was lacking. In the past several years, practitioners have been working on database design methodologies. Independently, researchers have begun to develop theories and models for data base design. It is our objective to bring together these different approaches into a book to facilitate the exchange of ideas.

Two symposia were held to compare and summarize various newly developed approaches to data base design.
The NYU Symposium on Data Base Design, organized by S. B. Yao (Chairman), S. Navathe, and J.L. Weldon; New York, May 1978.
The Symposium on Data Base Engineering, organized by T.L. Kunii; Tokyo, November, 1979.

The organizers of both symposia have edited proceedings that contained many excellent papers covering a wide range of data base design models and methods. In order to further disseminate these, the present book (which contains certain revised papers from the proceedings and several new papers) was compiled. The book is divided into two volumes. Volume 1 contains two parts. The first presents a general framework for considering the problem of data base design and classifies data base design techniques into two major categories: logical design and physical design. Logical design techniques are introduced in the second part. Volume 2 contains the third and fourth parts of the book. Physical design techniques are introduced in the third part. The last part gives examples of data base applications in several important fields. Both practitioners and researchers in the field of data base design should find this book useful. Although not a textbook, it might also be used in an advanced seminar on data base systems.

The symposia and this book were made possible through the efforts of many people. We would like to acknowledge NYU and IBM Japan for sponsoring the symposia.

S. B. Yao
University of Maryland

T. L. Kunii
The University of Tokyo

April 3, 1981

CONTENTS

† Presented in NYU Symposium on Data Base Design, New York, 1978.

†† Presented in IBM Symposium on Database Engineering, Tokyo, 1979.

An Integrated Approach to Database Design

S. B. Yao[+]
Purdue University

Shamkant B. Navathe[++] and Jay-Louise Weldon
New York University

This paper provides an integrated approach for research related to the problem of database design. The process of database design is classified into five phases: requirements analysis, view modeling, view integration, view restructuring, and schema analysis and mapping. The input, processing steps, and output for each phase are described. The problems associated with each phase are pointed out. Existing approaches to database design are reviewed and related to these five phases. The significance of this integrated approach for the development of computer-aided methodologies for database design is discussed.

1. INTRODUCTION

At the current state-of-the-art, the methods used in the design of database applications are essentially trail-and-error, supported by neither a scientific foundation nor an engineering discipline. The ad hoc approach to design frequently leads to inflexible solutions that do not meet the prescribed requirements. Costly remedial measures often produce more delay in operation without a tangible improvement. Much of the existing information on system design is presented in the form of individual analyses. These analyses do provide valuable insight, but they can hardly be adequate substitutes for a systematic design discipline.

It is generally accepted that there are two levels in the design of a database a) the logical design, defining and combining the views of many applications into a centrally controlled and maintained logical databases, with provisions for data sharing and security; and b)

+ Presently at University of Maryland.
++Presently at University of Florida.

the physical design, including all the implementational details and considerations of a particular database system.

In this paper, we will address mainly the design issues which apply at the logical design level. Since realistic databases involve thousands of data elements, and the evaluation of an enormous amount of structured information is implied (e.g. see Raver and Hubbard [A5]), it is desirable to develop computer aided tools to aid the design. In what follows, a conceptual framework is presented within which current research in logical database design is reviewed. It is also suggested as to how these seemingly unrelated approaches may be integrated into a computer aided design system.

2. THE DATABASE DESIGN PROCESS

The problem of database design is rich; it ranges from system-independent analysis to system-dependent optimization. Existing research tends to concentrate on only a few aspects of the design process. Consequently, each approach has its own view of the design process. It is desirable to define a general design process in order to compare and integrate existing approaches.

The process of database design can be divided into five general steps:

1) Requirement Analysis. The problem or environment in the real world must be analyzed to make the necessary components of the database explicit and to elicit both the data and processing needs of all potential database users.

2) View Modeling. Using the results of step 1 as input, abstract representations must be developed that correspond with each user's view of the real world. This step both verifies the previous step and lays a basis for the next.

3) View Integration. The several (and perhaps conflicting) user views must be integrated into one global or community view of the database. This global view must continue to support all user views.

4) View Restructuring. If the target system is known, a given community view can be mapped into alternate logical structures in the particular system. This step takes as input the canonical representation of a community view and restructures it into multiple structures in the target system.

5) Schema Analysis and Mapping. This step arrives at the storage level representation of data in the given target system. An analysis of physical implementation alternatives is performed and optimal storage structures are chosen.

If a target database management system has not been selected, steps 4 and 5 must be repeated for each candidate target system.

It is important to note the following:

1) It is observed that the requirement analysis provides input to all other design steps, since the information extracted from applications is relevant to all design stages.

2) Each design step produces not a unique solution but a set of solutions associated with measures which represent various properties of the particular solution.

3) The designer should interact with the design process to select an appropriate solution as the input to the next design step. A selection criterion must be stated.

4) Since the design requirements collected by the requirement analysis may be incomplete and inconsistent, the designer must be consulted to resolve ambiguities.

5) The design is usually not a single-pass process. Various conditions discovered by the designer may force a re-design and iterate to an earlier design step.

Although there exists a large amount of work in the literature which relates to database design, there is no existing approach which is comprehensive enough to address all the steps of the design process described above. In the following sections we will specify the inputs, processes and outputs of each step. Existing design methods will be discussed under the above framework. Problems yet to be addressed will be pointed out.

3. REQUIREMENT ANALYSIS

Requirement analysis provides initial input to the database design process. This step seeks to identify each user (application) of the database and, for each, analyzes the user's requirements regarding the content and the use of the database. Ideally this step should provide input which is both an accurate representation of the users' views and also a complete specification of the required database (i.e. all succeeding steps will be able to draw the data they require from the requirement analysis output). Further, the output of this stage should be in a form which is both usable by the succeeding steps and also amenable to review and verification by the users of the database. Thus, to be effective, a methodology for requirement analysis should include:

- a specification of the data to reside in the database
- suggested techniques for data collection (e.g., analysis of oral or narrative descriptions, document review, accumulation of performance statistics).
-a language or format for data collection and review
-an output specification which is compatible with the input to one or more view modeling techniques.

A specification of necessary data has been proposed by Kahn [R4] which classifies the information structure and the process structure. Some typical design inputs following this classification are shown in Figure 1. A major shortcoming of existing approaches to database design has been that they consider either the information structure oriented input or the process structure oriented input, but fail to incorporate both.

Requirement analysis for database design may be considered a special case of the general problem of gathering and specifying requirements for information systems. Little work has been reported in the literature which focuses on the requirement analysis for database design per se. However, there is a large body of literature devoted to the more general problem as evidenced by a special issue of IEEE Transactions in Software Engineering [R3]. The work in this area consists of development of high level languages to specify, store and retrieve the description of an information system being developed. Facilities for analysis, report generation, cross referencing, and

A. The information-structure-oriented design input:

-for each entity: entity name cardinality;
-for each attribute: attribute name, repeating factor, length of data value, value set size, probability of existence;
-for each intra-entity relationship: relationship name, attributes related, cardinality of multi-valued dependency;
-for each inter-entity relationship: relationship name, relationship type (aggregation/generalization), entities related, cardinality ratio, probability of existence, whether used for identification of instances.

B. The process-structure-oriented design input:

-for each process: frequency of occurrence, priority and weight, precedence, volume of data processed, data items required;
-for each access operation: type of operation, mode of access (random/sequential), hit ratio, frequency of occurrence, frequency of data items accessed, frequency of intra-entity relationship path accessed, frequency of inter-entity relationship path accessed.

Figure 1. Database Design Input parameters.

selective display are present in each system.

The PSL/PSA technique developed by the University of Michigan ISDOS project [R6] uses the Problem Statement Language to describe the following attributes of a system: input/output flows, structure between system components, size and volume, system dynamics. The data structure aspect of system description includes all the relationships which exist among data used and/or manipulated by the system as seen by the users of the system. The ELEMENT, ENTITY, ATTRIBUTE, GROUP, SET are some of the PSL statement-types used to describe a data structure.

The data derivation aspect of the system description specifies which data object are involved in particular PROCESSES in the system. It is concerned with what information is used, updated and/or derived, how this is done and by which processes. The PSL/PSA system has a command language facility which a designer uses to retrieve and display parts of the database containing the system description. A number of reference, summary and analysis reports can be produced. If

PSL/PSA were to be used for view modeling/integration, each view would either have to be described in PSL syntax or extracted from the central PSL/PSA database.

The CASCADE system at University of Trondheim, Norway is another system which has the objective of developing formal tools of information system analysis and design. The system specification and system presentation modules in CASCADE store a description of the information system. The information objects are related to other objects, variables, and elements. Processing is represented in the form of a tree structure. The coded representations of information and process structures would constitute an input to logical database design [R5].

Hammer et al have developed a very high level language called BDL which a designer could use to develop transaction processing programs for business applications [R2]. The basic data structure of this language is a document, or form, composed of fields of data items or data groups. Homogeneous collections of documents may be defined as files. BDL addresses the implementation of individual applications. As such the only provision for data sharing is redundancy, i.e. one document flowing through more than one application.

The design methodology of Bubenko et al [R1] starts with a knowledge of the queries which will be used against the database (all the queries which will ever be used, not just a minimal set which must be supported), and derives two schemas: a response-oriented schema and a storage-oriented schema. The storage-oriented schema corresponds to what might be called the normalized database design, as it stores each item of information once and does not have redundant set-types or links. The response-oriented schema is designed so that each query will have to access as few record types as possible when ordering and other constraints of the design procedure are taken into account. The design procedure then derives the final design as a compromise between the two schema alternatives.

The design system implemented by Yao et al [A8,A9,A10] uses a Functional Data Model (FDM) to capture the information requirements and a Transactions Specification Language to capture the processing requirements. The system provides interactive facilities to edit, verify, and analyze the design requirements. Tools are also provided to perform many analyses in the view modeling and integration stages.

The existing approaches to view modeling or to logical database design in general have not addressed the collection and analysis of information and processing requirements in detail. For example, IBM's DBDA [A5] is initiated by a list of data items and inter-item relationships obtained from input/output documents while Smith & Smith assume that the designer can name and assign attributes to the objects in the abstraction structure [M6]. CINCOM Systems, Inc. has a requirements analysis plan that utilizes tabular checklists to collect information on both entities and processes [D4].

An integrated approach to database design requires that the requirement analysis step provide data which is both accurate and complete. In other words, the user views should not be misrepresented by inadequate document sampling or incorrect formulation of required processes. However, redundant relationships and even inconsistent data groupings may not be eliminated at this step. Implicit decisions on such issues may freeze the design too early and result in artificial design constraints. The view integration and schema analysis steps will address these questions explicitly and the designer can then make a better decision. The level and nature of this tolerable uncertainty in requirement analysis has yet to be determined.

As mentioned in the overview of the design process, requirement analysis provides input to every step of the process. The designer and the users interact to develop the specification and refine it as the design progresses. Thus the full extent of requirement analysis will be determined by the input requirements of the succeeding steps.

4. VIEW MODELING

The objective of the view modeling step is to represent each user's view of the database using a common modeling technique. Once the views are represented, the designer can compare and integrate them. Figure 2 shows a schematic for the View Modeling step. Inputs to this step consist of information and processing requirements as determined by the requirement analysis step, as well as semantic information about the application which is known by the user or designer of the data base. The View Modeling process includes a modeling process and a verification process. In the modeling process inputs are analyzed and formal representations of user views are

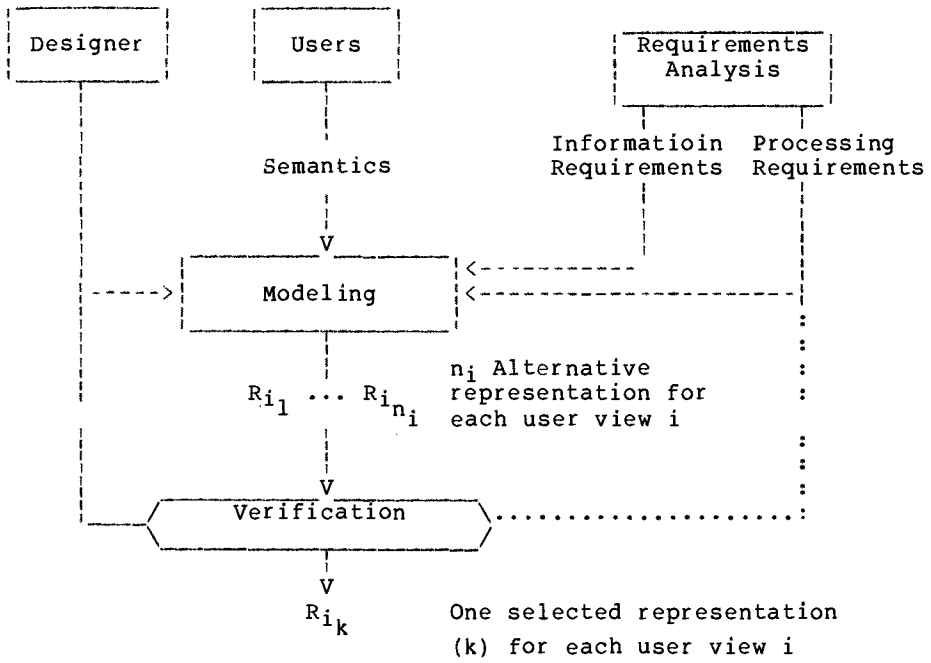

Figure 2. The View Modeling Step.

developed. In the verification process these representations are evaluated by the designer and the users in light of their requirements and a representation is selected for each view. The output of View Modeling is therefore a formal representation of each user's view.

The view modeling step is most clearly addressed by database design methods which are explicitly concerned with data modeling. Data models of Bachman [M1], Codd [M3], Senko et al. [M5] and Chen [M2] all accept as input, descriptions and attributes of entities and the relationships among these entities. Using the primitives of the target data model, a representation is developed which embodies the semantics of these entities and relationships. Bachman represents entities as record types and relationships as owner-member set types. Loops and m:n relationships are then eliminated to produce a network data model. Codd represents both entities and relationships as relations. Using the data semantics in the form of functional dependencies relations are normalized.

The DIAM [M5] incorporates four successive models of data, starting at a highly abstract level and going down to the level of encoding on physical devices. The Entity Set Model (ESM) and the string model are of interest in view modeling since they deal with logical data structures. In ESM the basic item of information is an entity which corresponds to some real-world-object or concept, and appears as a collection of attribute values. The string model is used to represent attribute grouping and relationships among entities.

Chen's entity-relationship model is a simple extension to the data structure diagram. It is simple and easy to conceptualize, and claimed to be semantically richer than the relational and the entity set model. The entity-relationship model represents a view by means of entities, relationships, and the attributes of each of them. However, it does not allow the representation of relationships between two relationships, or between an entity and a relationship.

The database abstraction methodology [M6] represents both entities and relationships as objects. This method distinguishes between two different types of relationships: association of similar entities (generalization) and association of related entities (aggregation). The semantic processes of aggregation and generalization are used to build a multi-dimensional hierarchy of objects which represents the user's view.

Navathe and Schkolnick [M4] draw from the data abstraction model of Smith and Smith and propose a technique for view representation to achieve a better modeling of the usage perspective and to incorporate the relationships among data instances, especially those which are used for identification purposes. The objective is to obtain a vehicle which represents a user view as explicitly as possible.

In all of the methods described above the "objects" of the view representation produced are data groups. The contents of these groups are either unspecified (Bachman), or specified directly by the designer (Chen, Codd, Navathe & Schkolnick, Senko et al, Smith & Smith). Codd, Senko et al and Smith & Smith allow the designer to determine identifying data elements (keys) for each object whereas Navathe and Schkolnick explicitly model the identification of instances.

In database design methods oriented toward producing implementation level schemas, the view modeling step is less distinct: it is

more tightly bound to the requirements analysis or view integration steps. In general, these methods are process-oriented and define a view as data used in a process. Most take descriptions of data items, item relationships and usage of data elements as input to the view modeling step.

The CINCOM method [D2, D4] relies on descriptions of data elements, processing specifications and policy constraints on events and data as input. Each process is then documented as a series of events and data elements are associated with the events in which they are used. Each view, or function, is represented by a flow diagram of events, annotated with associated data elements. These diagrams become the input to an analysis of frequency-of-use which groups elements, determines keys, and synthesizes a community view. This is described further in Section 5.

In Gerritsen's method [A1], each view is described by the queries that must be supported. These queries in the HI-IQ language, names of entities, and data item descriptions are the input to view modeling. Gerritsen's design system derives relationships between items and between items and entities from the queries. The information structure needed to support the view is then represented by a series of assertions governing the location of items and entities in the final DBTG schema. The cardinality ratios of the relationships described are implicit in these assertions, since in a DBTG schema a record type placed "above" another one implies a 1:m relationship

In DBDA [A2, A5] views are represented by directed graphs of data items and their relationships on which the cardinality of each relationship is recorded. These diagrams are produced by recording data items from existing input/output documents and analyzing their interrelationships. Nodes which are the source of relationships, but not the target of any are implicitly regarded as keys.

Mitoma also uses data item descriptions and item interrelationships as input to view modeling [A3]. He defines data relations (or binary relationships) for each pair of items to be used together. The items and data relations are recorded using a non-directed graph, in which each node is an item and each edge, a data relation. Unlike the other methods, Mitoma's view representation contains explicit and detailed instance level information. The cardinality of each data relation (the number of instances of item pairs) is recorded. In addition, the cardinality ratio for each relation is recorded specifi-

cally in terms of its average and maximum values, e.g. 2:5 or 1:250 instead of m:n or 1:n.

The process-oriented methods described above address the problem of data grouping during view integration rather than during view modeling. Key determination is similarly postponed until view integration, except in the case of DBDA.

Table 1 summarizes the characteristics of the view modeling step for the several design methodologies discussed in this Section.

Table 1. Characteristics of View Modeling in Several Database Design Methods.

CHARACTERISTICS	Chen	CINCOM	Codd	DBDA (Hubbard)	DBTG (Bachman)	Gerritsen	Mitoma	Navathe & Schkolnick	Senko	Smith
Input:										
info. requirements										
objects	x		x		x	x		x	x	x
items		x		x		x	x		x	
processing requirements		x		x		x	x	x		
Processing:										
semantic analysis	x		x		x			x	x	x
item analysis		x		x		x	x		x	
data grouping	x		x						x	x
key determination/ identification analysis			x	x				x	x	x
Output:										
representation										
objects	x		x		x	x		x	x	x
items		x		x		x	x		x	
explicit cardinality ratios of instances				x			x			
explicit cardinality of instances				x			x			

5. VIEW INTEGRATION

The objective of the view integration step is to merge the several view representations produced in view modeling into an integrated canonical structure. This output structure represents the community view and must satisfy the following:

a) it must be internally consistent

b) it must reflect accurately each of the original views

c) it must support the processing requirements specified in the requirements analysis step.

As with view modeling, the integration step may actually produce several candidate structures, all of which would then be verified against processing or specifications and those satisfying the process requirements will be regarded as community structures (see Figure 3).

View integration initially involves some editing to remove inconsistencies. Inconsistencies and redundancies may arise at the data-element, data-group or data-relationship level, in the form of one name referring to different components (homonyms), or different names referring to the same component (synonyms). Data-relationship redundancy also includes the relationships that could be implied by other relationships in the system. These could be reported and subsequently corrected after designer intervention.

When no inconsistencies or redundancies remain to be resolved the local views may be integrated by consolidating data and relationships into a community view. Again, several alternatives may exist. Process verification is required to determine which of the alternate structures satisfy processing requirements. The process specification may be either in terms of queries or procedures. Verification will consist of insuring i) that keys used in processing may be found, ii) that data items required are available and iii) that the relationships needed to associate data items or groups are represented. Efficiency related characteristics such as access path length and access frequencies, will be considered in the Schema Analysis and mapping steps.

View integration approaches can be generally classified into three categories:

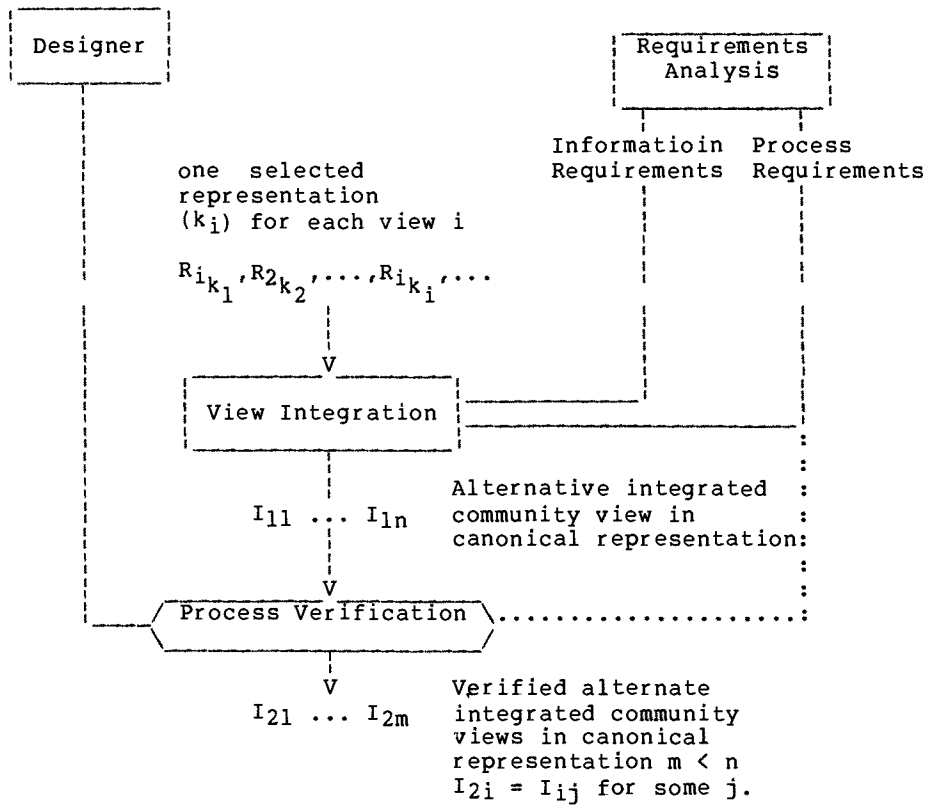

Figure 3. The view Integration Step.

i) Item level synthesis using frequency information,

ii) Item level synthesis using items and functional dependencies,

iii) Merging of object-level structures.

5.1. SYNTHESIS USING ITEMS AND FREQUENCIES

In this approach a user view is modeled in terms of items, associations between items and frequency information about the use of these associations by processes. Integration follows the rationale that items most frequently accessed must get priority in the structure by being placed nearest to the entry points in the structure.

This approach is followed in a method developed by CINCOM Systems, Inc. [D2, D4] and in another method described by Mitoma et al [A3,A4,A7].

In the CINCOM method, user views are represented by flow diagrams of processes. The data items which are required for each processing step, or event, are noted on the diagrams. Using this information three statistics are computed for each data item:

1) the number of events in which it is used

2) the number of other data items with which it is used

3) the percentage of data items with which it is used often (i.e. for which the frequency of use exceeds a specified threshold).

For example, suppose a data item A is used by 10 events, is used with 6 other data elements, and in 4 of these 6 cases, the frequency of use exceeds a threshold of 70%. The three statistics for A would then be: 10,6 and 67%, respectively.

A classification scheme (see Table 2) is used to classify data elements into (a) keys, (b) attributes with one key, and (c) attributes with 2 or more keys. (These categories correspond to TOTAL control keys, master file elements, and variable file elements,

Table 2. Item Classification in CINCOM Method.

	No. of Events (#1)	No. of Other Data Elements (#2)	% of Data Elements with a High Data Usage (#3)
Entity Key	Used by HIGH No.	Used with a HIGH No.	Used a LOW % of Time
Attribute to 1 Key	Used by LOW No.	Used with a LOW No.	Used a HIGH % of Time
Attribute to 2 or more Keys	Used by AVERAGE No.	Used with an AVERAGE No.	Used an AVERAGE % of Time