OBSERVATION ABOUT EXISTING PROBLEM

The problems in this paper are those of data independence - independence of application programs and terminal activities from growth in data types and changes in data representation and certain kinds of data inconsistency which are expected to become troublesome even in non deductive systems. Users of large data banks must be protected from having to know how the data is organised in the machine. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed.

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence.

The variety of data representation characteristics which can be changed without logically impairing some application programs is still quite limited. The current model of data is still cluttered with representational properties.

The principal kinds of data dependencies that need to be removed are:

- Ordering dependence: Elements of data in a data bank may be stored in a variety of ways some involving no concern for ordering some permitting each element to participate in one ordering only, others permitting each element to participate in several. Those application programs which take advantage of the stored ordering of a file are likely to fail to operate correctly if for some reason it becomes necessary to replace that ordering by a different one.
- Indexing dependence:- An index is a purely performance oriented structure. From an informational standpoint, an index is a redundant component of the data representation. So an ability to create and destroy indices from time to time will probably be necessary.
- Access path dependence: Many of the existing formatted data systems provide users with tree-structured files or slightly more general network models of data. Applications programs developed to work with these systems tend to be logically impaired if the trees or networks are changed in structure.

Key Contributions and Description

The relational view Provides a means of describing data with its natural structure only. It also Provides a basis for a high level data language which will yield maximum independence. It Helps in remedying derivability, redundancy, and consistency of relations. It permits a clearer evaluation of the scope and logical limitations of present data systems. The term relation is used in its usual mathematical sense. Given sets, R is a relation on these sets if it is a set of n-tuples each of which has its first element from the first set, second element from the second and so on. Several existing information systems fail to provide data representations for relations which have two or more identical domain. The totality of data in a data bank may be viewed as a collection of time-varying relations. These relations are of assorted degrees. As time progresses, each n-ary relation may be subject to addition of additional n-tuples, deletion of existing ones and alteration of components also.

Normally one domain of a given relation has values which uniquely identify each element n-tuple of that relation. Such a domain is called a primary key. A primary key is non redundant if it is either a simple domain (not a combination) or a combination such that none of the participating simple domains is superfluous in uniquely identifying each element.

The paper proposes that the users deal not with relations which are domain ordered but with relationships which are their domain-unordered counterparts. Domains must thus be uniquely identifiable at least within any given relation without using position. The domain name must be qualified by a distinctive role name which serves to identify the role played by that domain in a given relation.

A common requirement is for elements of a relation to cross-reference other elements of the same relation or elements or elements of a different relation. The data in a data bank is usually separated into entity descriptions and relations. In the user’s relational model there appears to be no advantage to making such a distinction. Non atomic values can be discussed within the relational framework. Thus some domains may have relations as elements. These relations may in turn be defined on non simple domains and so on. The possibility of eliminating non simple domains appears worth investigation. There is a very simple elimination procedure called normalisation. Normalization is the process

of reorganizing data in a database so that it meets two basic requirements:

- There is no redundancy of data (all data is stored in only one place), and
- data dependencies are logical (all related data items are stored together).

If normalisation is to be applicable, the unnormalised collection of relations must satisfy the following conditions:

> The graph of interrelationships of the non simple domains is a collection of trees and No primary key has a component domain which is non simple.

The simplicity of the array representation which becomes feasible when all relations are cast in normal form is not is not only an advantage for storage purposes but also for communication of bulk data between systems which use widely different representations of the data.

The communication form would be a suitably compressed version of the array representation of the data. The communication form would be a suitably compressed version of the array representation and would have the following advantages:

- It would be devoid of pointers(address valued or displacement valued)
- It would avoid all dependence on hash addressing schemes and would contain no indices or ordering lists

The user’s relational model is set up in a normal form, names of items of data in the data bank can take a simpler form than would otherwise be the case.A first order predicate calculus suffices if the collection of relations is in normal form. Such a language would provide a yardstick of linguistic power for all other proposed data languages and would itself be a strong candidate for embedding with a host language with appropriate syntactic modifications in a variety of host languages. Associated with a data bank are two collections of relations: the named set and the expressible set. The named set is the collection of all those relations that the community of users can identify by means of a simple name or identifier. The expressible set is the total collection

of relations that can be designated by expression in the data language. Since some relations in the named set may be time independent combinations of others in that set, it is useful to consider associating with the named set a collection of statements that define these time-independent constraints.

Many operations on relations are also described as:

- permutation
- projection
- join
- composition
- restriction.

Redundancy in the named set of relations must be distinguished from redundancy in the stored set of representations. So the authors define strong and weak redundancy. A relation is strongly redundant if it has a projection that is derivable from other projections of relations in the set. A relation is weakly redundant if it contains a relation that has a projection which is not derivable from other members but is at all times a projection of some join of the other projections of relations in the collection. Consistency in database systems refers to the requirement that any given database transaction must change affected data only in allowed ways. Any data written to the database must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.