RECENT INVESTIGATIONS IN RELATIONAL DATA BASE SYSTEMS

E. F. Codd
IBM Research Laboratory
Monterey and Cottle Roads
San Jose, California 95193
(408) 256-7648

The objectives and characteristics of the relational approach to the management of large formatted and integrated data bases are briefly reviewed. We then consider recent advances in the following topics: normalization of the relational model; data base sublanguages for programmers and non-programmers; the problem of superimposition of multiple views on top of the relational model; and data exchange policies in a network of mutually remote data bases. Listed in the conclusion are some areas of investigation in relational technology requiring immediate attack and some that are less urgent.

1. OBJECTIVES

In 1968 it was possible to observe two mutually incompatible trends in formatted data base systems: on the one hand, the tendency of systems designers to expose users of their systems to more and more complicated types of data structure and, on the other hand, the increasing interest in establishing integrated data bases with a much higher degree of data inter-relatedness and on-line interactive use by non-programmers. At about the same time, it was becoming clear that users were spending too much in manpower and money on re-coding and re-testing application programs which had previously worked satisfactorily but which had become logically impaired by data base growth or by changes in the stored data representation for various reasons (the so-called data independence problem).

In a series of papers [1-5] we proposed a specific relational model with (we believe) a novel set of operators and normal forms. For prior applications of relations, see references cited in [1,2]. The objectives of this work are:

1. to provide a high degree of data independence;
2. to provide a community view of the data of spartan simplicity, so that a wide variety of users in an enterprise (ranging from the most computer-naive to the most computer-sophisticated) can interact with a common model (while not prohibiting superimposed user views for specialized purposes);
3. to simplify the potentially formidable job of the data base administrator;
4. to introduce a theoretical foundation (albeit modest) into data base management (a field sadly lacking in solid principles and guidelines);
5. to merge the fact retrieval and file management fields in preparation for the addition at a later time of inferential services in the commercial world;
6. to lift data-based application programming to a new level -- a level in which sets (and more specifically relations) are treated as operands instead of being processed element by element.

In connection with the second objective, it is important to remember that data bases are being established for the benefit of end users, and not for the application programmers who act as middle-men for today's data processing needs. Fig. 1 displays the author's somewhat conservative assumptions about future trends in data base interaction. For a description of the casual user and a subsystem to support his interaction, see [6].

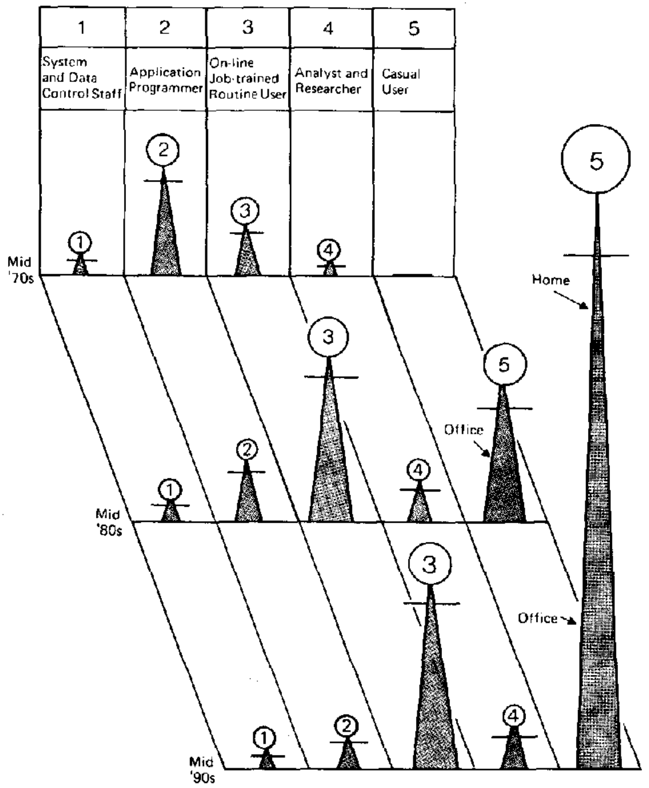


Fig. 1 Anticipated Use of Large Integrated Data Bases

2. RELATIONAL MODEL

In the relational approach there exists an interface at which the totality of formatted data in a data base can be viewed as a finite collection of non-hierarchic relations of assorted degrees defined on a given collection of simple domains (domains whose elements are non-decomposable as far as the data base management system is concerned). The extension (or instantaneous value) of each relation of degree n is a simple table with n columns and no duplicate rows. Accordingly, within each relation each tuple (or row) is uniquely identifiable by its content alone.

The extension is, of course, subject to change with time as tuples are inserted, modified, and deleted. While there is no prohibition against ordering the tuples of any relation in the user's view, there is a clear advantage in requiring that such an ordering be entirely re-constructible from the table values, since it is then possible to derive and retrieve every meaningful relation by application of a simple collection of commands (e.g., the operators of the n-ary relational algebra), without programming a tuple-by-tuple search.

The intension (or meaning) is much less subject to change than the extension -- and for many purposes can be treated as if it were time-independent. We include in the intensional part of the relational model the declaration of domains and relations, units and range information for each domain, information as to the applicability of comparisons based on LESS THAN, together with a list of all the integrity constraints. These constraints may be represented by procedures, by parameters for procedures, or, in some cases, by query language expressions [18].

We may divide the constraints into two types: those that define valid states of the data (we call these static integrity constraints) and those that define side-effects of various kinds of transactions, particularly insertions and deletions (dynamic integrity constraints). The static integrity constraints include all of the elementary functional dependencies upon which the normal forms are based [3,4,11], plus designations of primary keys. Also included in the static constraints are set inclusion declarations (for example, the set of suppliers who are supplying parts must be a subset of the set of suppliers in the supplier-describing relation).

An important and distinctive feature of the relational model is the separation of integrity and security constraints from the logical data structure. We may accordingly change the constraints without changing the data structure and possibly impacting application programs and terminal activities.

A variety of representations can be used for storing the data, so long as these representations are isomorphic to the relational model with respect to insertion, update, and deletion. The application programs, of course, do not refer to the storage representations directly. They are written to operate upon the community view or some superimposed specialized view (see section 5 below).

The fundamental differences between the data base relational model and the network model (as exemplified by DBTG [23] are discussed in [7,8].

3. NORMALIZATION OF RELATIONS

In [3,4] six aims of normalization of relations are listed. Perhaps the two most important are:

1. To reduce the need for restructuring the collection of relations as new types of data are introduced, and thus increase the life span of application programs;
2. To reduce the incidence of undesirable insertion, update, and deletion anomalies.

The concepts of full and transitive dependence of attributes upon one another were introduced, and the second and third normal forms were defined.

In [12] Kent proposed improvements to the definitions of the second and third normal forms. These improvements remove the somewhat arbitrary distinction drawn in [3,4] between prime and non-prime attributes. More recently, Boyce and Codd developed the following definition:

> A relation R is in third normal form if it is in first normal form and, for every attribute collection C of R, if any attribute not in C is functionally dependent on C, then all attributes in R are functionally dependent on C.

While this definition is logically equivalent to that of Kent, it has the advantage of avoiding reference to the concepts of primary key, full dependence, and transitive dependence. As a consequence, the normalizing algorithm is significantly simplified.

Now, even though each relation in a collection of relations may be in third normal form, it does not follow that the collection itself is in optimal third normal form. For example, consider the collection consisting of two relations R(A,B) and S(A,C), where in each case the primary key is underlined as usual. Suppose R is non-loss joinable with S on A. The join T of R with S on A clearly possesses the functional dependencies of B on A and C on A, but it might also possess the dependency of C on B. Let us suppose it does have this additional dependency. Then, the given collection of relations can be replaced by the more optimal collection consisting of the projection T(B,C) together with the relation R. This example shows the need to consider not only the functional dependencies within the given relations but also the dependencies within all the non-loss joins of these relations, when attempting to cast a given collection in optimal third normal form.

4. DATA SUBLANGUAGES

For expository purposes, we shall distinguish five kinds of language, all of which provide independence of programs and terminal activities from the physical (or stored) representation of the data. These kinds are: element-by-element; algebraic; mapping-oriented; relational calculus; and natural language (e.g., English).

4.1 Element-by-Element Data Sublanguages

A primitive procedural interface for element-by-element manipulation of a collection of n-ary relations is described in detail in [13]. The 14 basic commands provide for the creation and dropping of relations, the insertion, modification, movement, and deletion of tuples, the retrieval of tuples via system-generated identifiers, and relation-scanning operations that permit some degree of optimization of search in the implementation. The creation and dropping of inversions are under programmer control at this level, because this interface is intended to be used to interpret higher level data sublanguages efficiently. Maintenance of these inversions is, however, a system responsibility, since it can be handled more efficiently below this interface than above it.

Generally speaking, when making a query on a remote data base, one would prefer not to have to request elements one at a time in a low level language. This is one of the reasons for investigating the following data sublanguages.

4.2 Algebraic Data Sublanguages

At the algebraic level, retrieval of data is viewed as the formation of a new relation from the data base relations by use of some operation of the algebra. It should be stressed that these operations act upon entire relations as their operands. As pointed out in [5,14] these operations provide a powerful and concise vehicle for expressing queries, and they are comparatively easy to implement efficiently in the context of the relational model. Unfortunately, these operators cannot be readily implemented in the CODASYL DBTG framework, because its owner-coupled set occurrences do not behave as mathematical sets [7].

Examples of implementations of the algebraic level interface on the relational model are found in MACAIMS [15], RDMS [28], and the Peterlee IS/1 [16]. The last system has an interesting variant of the join operator, one that is oriented both to convenience of use and efficiency of implementation.

4.3 Mapping-Oriented Data Sublanguages

Every binary relation R (no matter whether it is one-many, many-one, or many-many) can be regarded as a set-valued function which maps each element of the first domain of R into the set of all associated elements in the second domain. There is also a similarly defined function that maps elements in the second domain into sets of elements in the first. This idea can obviously be extended to relations of higher degree. Boyce, Chamberlin, King, and Hammer have developed a relationally complete data sublanguage [17] based on this notion. An English-oriented version of this sublanguage named SEQUEL [18,19] appears to be a strong candidate for use by both programmers and non-programmers who are willing to tolerate a small amount of training.

4.4 Relational Calculus Data Sublanguages

A data sublanguage named ALPHA based on the relational calculus for non-hierarchic n-ary relations was described in [2], its foundation was defined in [5], and a syntax was specified in [6]. Tactics for efficient interpretation were introduced in [20] by Palermo, and are also discussed in [21] by Rothnie. In each case a small scale experimental implementation was developed.

The relational calculus type of interface has the advantage that the user specifies what he wants and avoids specifying how the system should retrieve the information, thus leaving the system with the complete responsibility for searching efficiency. Another advantage is its conciseness. In [29] Frank and Sibley (selecting their own example) developed a DBTG schema for a sample data base, together with a subschema and COBOL-DBTG program for a sample application. In [7] we show that conversion of the DBTG schema to a relational schema results in an 80 percent reduction in the number of lines of code for the schema itself; and conversion of the COBOL-DBTG program to a corresponding COBOL-ALPHA program results in a 90 percent reduction in the number of lines of code for the application.

4.5 Natural Language for Non-Programmers

By natural language (in this paper) we mean any language in use today for oral conversation between people, providing it has tokens that are acceptable to computers. We shall use English as an example. Much work has been done in developing translators and interpreters for queries stated in English. Almost all of these systems provide one-way translation only (from English to some computer-oriented language). The REL system [26] at the California Institute of Technology and the CONVERSE system [27] at System Development Corporation in Santa Monica are examples. Such systems are based on the (unstated) assumption that the user knows what he wants and knows how to express his needs perfectly in system-comprehensible English. This assumption may be viable for analysts and researchers who have a clear job-incentive to learn to live with the system's restrictions. Such a learning overhead with the patience it implies is incompatible with casual interaction by non-programmers.

In [6] the author proposed seven steps to arrive at viable support for casual interaction. These steps are:

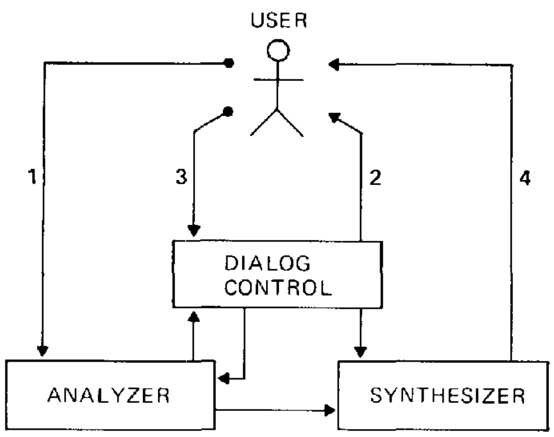
1. Select a simple data model
2. Select a high level logic as internal target
3. Introduce clarification dialog of bounded scope
4. Introduce system re-statement of user's query
5. Separate query formulation from data base search
6. Employ multiple choice interrogation as fall-back
7. Provide a definitional capability.

The author is implementing an experimental query formulation subsystem called RENDEZVOUS that embodies these seven steps (see Fig. 2).

5. SUPERIMPOSITION OF MULTIPLE VIEWS

Both Guide-Share [22] and the CODASYL Data Base Task Group [23] call for multiple views of the data, so that different application programs can interact with distinct views. Date and Hopewell [9,10] are more specific on this topic. None of these reports places clear limits on the range of application views that are permissible or required for a given system logical view (or community schema). For application programs that do more than merely read the data, there are theoretical limitations which must be observed if data base integrity (including consistency of all permitted views) is to be maintained.

To illustrate the kind of difficulty one encounters, suppose that the community schema includes two relations R(A,B) and S(B,C). Suppose that a user requests T(A,B,C) as his schema, where T is defined to be the natural join of R with S on the common



1. User makes initial statement of his query (unrestricted English)
2. System interrogates user *about* his query (to obtain information which is missing or hidden in language the system does not understand, and to resolve ambiguities)
3. User responds to system interrogation
4. System provides a re-statement of user's query in system English (in a very precise way, based on the n-ary relational calculus)

Fig. 2 RENDEZVOUS Subsystem

attribute B. Further, suppose that at some instant R and S have the following tabulations:

R (	A	B)
	s	1
	t	1
	....	

S (	B	C)
	1	u
	1	v
	....	

Then, the tabulation of T must be:

T (	A	B	C)
	s	1	u
	s	1	v
	t	1	u
	t	1	v
	.....		

Now, suppose the user who has T in his schema desires to delete just one row -- specifically, the triple (t,1,v). If he were allowed to do this, the relation T would become a relation that is not the join of any pair of relations whatsoever. Another way of expressing this is that there is no way of reflecting this deletion from T by means of corresponding deletions from R and S. In [1] we called the element 1 in domain B a point of ambiguity in the join of R with S on B. A simple and sufficient time-independent condition under which a point of ambiguity cannot arise is that either A is functionally dependent on B in R or C is functionally dependent on B in S.

Informally, we shall say that schema U is deletion-viable with respect to schema V if all deletions from tabulations of U can be faithfully simulated by deletions from corresponding tabulations of V. Of course, all applicable functional dependencies must be respected. It is not difficult to make this definition more formal. If U is deletion-viable with respect to V, this does not imply that V is deletion-viable with respect to U (consider the case cited above, in which the join T is not deletion-viable with respect to the relations R and S, whereas R and S together are deletion-viable with respect to T).

Insertion viability can be defined analogously to deletion viability. If we assume, as usual, that primary keys may not have undefined values whereas other attributes may, deletion viability of schema U with respect to schema V does not imply insertion viability (consider the case in which U is a projection of a relation R in schema V on non-key attributes of R). On the other hand, in the absence of dynamic integrity constraints triggered by deletion operations, insertion viability does imply deletion viability.

There has been much discussion [9,18] of supporting tree structured schemata on top of the relational model. The join operation is a vital part (but not the whole story) in the formation of tree structures from non-hierarchic relations. Thus, the problem of supporting tree structures with integrity must take into account the problem of supporting joins with integrity. A systematic investigation is needed to determine for any given class of non-hierarchic relations and associated integrity constraints what is the class of user views that can be supported with integrity.

6. DATA EXCHANGE

Consider a network of computers with one or more data bases at each node. Suppose a common requirement is that of transmitting collections of formatted data from one node to another. Let us focus upon the question of how these data collections are represented in storage at each node and how they are represented on the communication lines (ignoring, however, the representation of atomic items).

There are four principal policies that can be adopted to ensure that a collection transmitted from one node is acceptable and interpretable at the receiving node. The first policy is a very rigid one, namely that all nodes are required to use the same data base management system. This guarantees compatibility, but has the disadvantage that no node may make improvements or changes in the class of data representations supported by their node without all other nodes simultaneously introducing the very same changes.

The second policy is the opposite extreme, namely that of permitting free choice at each node of the stored data representations used at that node and also on the communication lines. It is then the responsibility of any two parties that want to communicate with one another to develop the necessary translation programs to make their bilateral communication possible. The disadvantage of this approach is that many such programs will become necessary sooner or later.

The third policy is that of developing a single "general" translator to replace the bilateral translators of the second policy. An identical copy of this network standard translator would be mandatory at each node. As Dennis has shown [24], there is no truly general translator that can handle all possible data representations. Thus, we must ask: "What happens if a new (and possibly very efficient) physical representation and access method are discovered which are beyond the capability of the standard translator?". Remember that we are only just beginning to establish a theoretical basis for physical representations of data, and therefore we can expect significant new developments in this area. The third policy would have a marked deterrent effect upon the introduction of a new data representation at some node, because of the need to obtain the consent of all other nodes to the required modification of the network standard translator.

The fourth policy is that of adopting a single network standard data representation for communication purposes (as proposed in [1], page 381). No constraints are placed upon the stored data representations adopted at any node. The only requirement is that each node develop its own translator to and from the standard communication representation for whatever stored representations are adopted at that node. Thus, changes in the stored representations at any node can be made without negotiation with other nodes. Third normal form relations [3] provide a rather simple basis for such a data communication standard within a network. One group that is investigating this approach is at the University of Michigan [25].

7. NEEDED INVESTIGATIONS

In this author's opinion, the most urgently needed investigations are:

1. development of concurrency control techniques specifically geared to the relational model;
2. ascertaining the performance that is attainable when the relational approach is applied to a large scale data base (at least one billion bytes) with concurrent access and modification;
3. development of superimposition theory (see section 5 above);
4. development of storage, access, and modification theory for collections of non-hierarchic n-ary relations;
5. demonstration of viability of natural language query formulation subsystems of the RENDEZVOUS type.

Inferential services and support for Zadeh-fuzzy concepts represent less urgently needed, but still necessary, areas of investigation.

ACKNOWLEDGMENT

It is a pleasure to acknowledge the author's indebtedness to C. T. Davies of IBM (the need for data independence), A. L. Strnad and R. C. Goldstein of MIT (the first implementation of a data independent, relational data base system with truly n-ary relational operators), C. J. Bell, M. G. Notley, T. W. Rogers of IBM Peterlee, England (the first IBM implementation), C. J. Date, P. Hopewell, I. J. Heath of IBM Hursley, England (contributions to data independent system architecture and normalization theory, plus exposition in Europe), R. F. Boyce, D. D. Chamberlin, W. F. King, F. P. Palermo, I. L. Traiger of IBM San Jose (discussions on normalization improvements, superimposition of multiple views, and concurrency control). The author is also indebted to the many correspondents, particularly in universities, who have shown a strong interest in this approach.

REFERENCES

[1] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

[2] E. F. Codd, "A Data Base Sublanguage founded on the Relational Calculus", Proc. 1971 ACM-SIGFIDET Workshop*.

[3] E. F. Codd, "Further Normalization of the Data Base Relational Model", Courant Computer Science Symposia 6, "Data Base Systems", New York City, May 24-25, 1971, Prentice-Hall.

[4] E. F. Codd, "Normalized Data Base Structure: A Brief Tutorial", Proc. 1971 ACM-SIGFIDET Workshop*.

[5] E. F. Codd, "Relational Completeness of Data Base Sublanguages", Courant Computer Science Symposia 6, "Data Base Systems", New York City, May 24-25, 1971, Prentice Hall.

[6] E. F. Codd, "Seven Steps to Rendezvous with the Casual User", Proc. IFIP TC-2 Working Conference on Data Base Management Systems, Cargese, Corsica, April 1-5, 1974, North-Folland.

[7] E. F. Codd, C. J. Date, "Interactive Support for Non-Programmers: The Relational and Network Approaches", Proc. 1974 ACM-SIGFIDET Workshop*.

[8] C. J. Date, E. F. Codd, "The Relational and Network Approaches: Comparison of the Application Programming Interfaces", Proc. 1974 ACM-SIGFIDET Workshop*.

[9] C. J. Date, P. Hopewell, "File Definition and Logical Data Independence", Proc. 1971 ACM-SIGFIDET Workshop*.

[10]C. J. Date, P. Hopewell, "Storage Structure and Physical Data Independence", Proc. 1971 ACM-SIGFIDET Workshop*.

[11]I. J. Heath, "Unacceptable File Operations in a Relational Data Base", Proc. 1971 ACM-SIGFIDET Workshop*.

[12]W. Kent, "A Primer of Normal Forms", IBM System Development Division, San Jose, California: Technical Report TR02.600 December 17, 1973.

[13]D. Bjorner, E. F. Codd, K. L. Deckert, I. L. Traiger, "The Gamma Zero n-ary Relational Data Base Interface: Specifications of Objects and Operations", IBM San Jose Research Report RJ1200, April 11, 1973.

[14]A. L. Strnad, "The Relational Approach to the Management of Data Bases", Information Processing 71, North-Holland Publishing Company, Amsterdam 1972, pp. 901-904.

[15]R. C. Goldstein, A. L. Strnad, "The MACAIMS Data Management System", Proc. 1970 ACM-SIGFIDET Workshop*.

[16]M. G. Notley, "The Peterlee IS/1 System", IBM UK Scientific Centre Report UKSC-0018, March 1972.

[17]R. F. Boyce, D. D. Chamberlin, W. F. King III, M. M. Hammer, "Specifying Queries as Relational Expressions: SQUARE", Proc. ACM SIGPLAN-SIGIR Interface Meeting, Gaithersburg, Maryland, November 4-6, 1973.

[18]R. F. Boyce, D. D. Chamberlin, "Using a Structured English Query Language as a Data Definition Facility", IBM San Jose Research Report RJ1318, December 10, 1973.

[19]D. D. Chamberlin, R. F. Boyce, "SEQUEL: A Structured English Query Language", Proc. ACM-SIGFIDET Workshop*.

[20]F. P. Palermo, "A Data Base Search Problem", Fourth International Symposium on Computer and Information Science, Miami Beach, December 1972 Academic Press.

[21]J. B. Rothnie, "The Design of Generalized Data Management Systems", Ph.D. Dissertation, Dept. of Civil Engineering, MIT, September 1972.

[22]Joint GUIDE-SHARE, "Data Base Management System Requirements", Guide or Share Distribution, November 1970.

[23]CODASYL, "Data Base Task Group Report", ACM, New York, 1971.

[24]J. B. Dennis, "On the Exchange of Information", Proc. 1970 ACM-SIGFIDET Workshop*.

[25]Sham Navathe, "Logical Normal Forms for Data Translation", Private Communication.

[26]F. P. Thompson, P. C. Lockemann, B. H. Dostert, R. Deverill, "REL: A Rapidly Extensible Language System", Proc. 24th ACM National Conference, New York, ACM 1969, pp. 399-417.

[27]C. H. Kellogg, J. Burger, T. Diller, K. Fogt, "The CONVERSE Natural Language Data Management System: Current Status and Plans", Proc. ACM Symposium on Information Storage and Retrieval, Univ. of Maryland, College Park, 1971, pp. 33-46.

[28]V. K. M. Whitney, "RDMS: A Relational Data Management System", Proc. Fourth International Symposium on Computer and Information Sciences, Miami Beach, Florida, December 14-16, 1972 Academic Press.

[29]R. L. Frank, E. H. Sibley, "The Data Base Task Group Report: An Illustrative Example", ISDOS Working Paper No. 71, February 1973, U.S. National Technical Information Service, Document AD-759-267.

* Proceedings of ACM-SIGFIDET Workshops on Data Description, Access, and Control are obtainable from ACM, New York.

[30]C. J. Date, "An Introduction to Data Base Systems", Addison-Wesley, 1975.

[31]M. Stonebraker, E. Wong, "Access Control in a Relational Data Base Management System by Query Modification", Electronics Research Lab., Univ. of California, Berkeley, ERL-M438, May 14, 1974.

[32]I. T. Hawryszkiewycz, "Semantics of Data Base Systems", MIT Project MAC, Cambridge, Mass., MAC TR-112, December 1973.

[33]M. Zloof, "Query by Example", IBM Yorktown Heights Research Reports RC4917, RC5115, July 2, 1974.

[34]C. Delobel, "La Structure de l'Information dans une Base de Donnee", Revue Informatique et Recherche Operationnelle MbB3, 1971, pp. 37-64.

[35]J. Rissanen, C. Delobel, "Decomposition of Files, A Basis for Data Storage and Retrieval", IBM San Jose Research Report RJ1220, May 10, 1973.

[36]C. Delobel, R. G. Casey, "Decomposition of a Data Base and the Theory of Boolean Switching Functions", IBM J. Res. & Develop., vol. 17, no. 5, Sept. 1973, pp. 374-386.

[37]W. W. Armstrong, "Dependency Structures of Data Base Relationships", Information Processing 74, North-Holland, pp. 580-583.