Data Base Research: A Survey

A. Blaser, H. Schmutz, IBM Wissenschaftliches Zentrum, Heidelberg, Tiergartenstr. 15

Abstract

The research activities in the area of data base systems are reviewed. Most of the issues considered by research institutes center around models of information, interactive data manipulation, system aspects, implementation techniques and modelling and analysis. Comparison with industry activities and documented user requirements shows differences of emphasis between research and development. Conclusions are drawn with respect to established and potentially emerging principles in the area of data base design and architecture and with respect to potential future trends in data base research.

TABLE OF CONTENTS

1. INTRODUCTION /49, 192/

The objective of this paper is primarily to provide an overview over past and present research activities in the data base area. This pa-

per does not survey commercially available data base software. Further, information retrieval systems and non-computeroriented aspects of information systems are not addressed. We are well aware of the danger of such limitations and recommend to the reader, who is interested in an introduction to the field, to study in depth some of the commercially available data base systems, such as IMS (A.J. Barnett and J.A. Lightfoot: Information Management System (IMS) - A Users Experience with Evolutionary Development. In Data Base Management Systems (D.A. Jardine editor), North Holland, Amsterdam, 1974) in addition to the literature referenced in this survey.

What is a data base system? This is already a question, which has been and still is subject of debates. We will make our definition with the help of a scheme , which in our experience is widely accepted. It is a simplification of an architecture scheme employed by a major standardization group (ANSI/X3/SPARC) and is very similar to schemes shown in Date's or Wedekind's book. To the authors knowledge, the IMS designer have been the first who implemented, consciously or unconsciously, such a scheme nearly a decade ago.

The scheme is shown in fig. 1 and shows persons, views of information, data mappings, programs and a data flow during retrieval of information. The conceptual view is the central point. It represents the way information is seen by the group responsible for the system aspects of stored, integrated information. This group is usually referred to as the "data base administrator". For a given system installation, a conceptual schema specifies which type of information may exist at the conceptual level. A schema describes, what is legal or "correct" and is therefore similar to a grammar defining the syntax of a language.

The conceptual view is never used directly. It serves as a central reference point for other views of information. For example, the physical or internal view of information reflects the way the information is actually stored in memory. Given the data base in conceptual form, we can construct the corresponding physical form with the help of a mapping, the conceptual to internal mapping (fig. 1 mapping C/I).

The use of conceptual information is through mappings. All of these mappings are in the responsibility of the data base administrator and are specified in a data definition and mapping language. The mappings between conceptual and external views serve a double purpose: (a) to select the subpart of information necessary and sufficient for a spe-

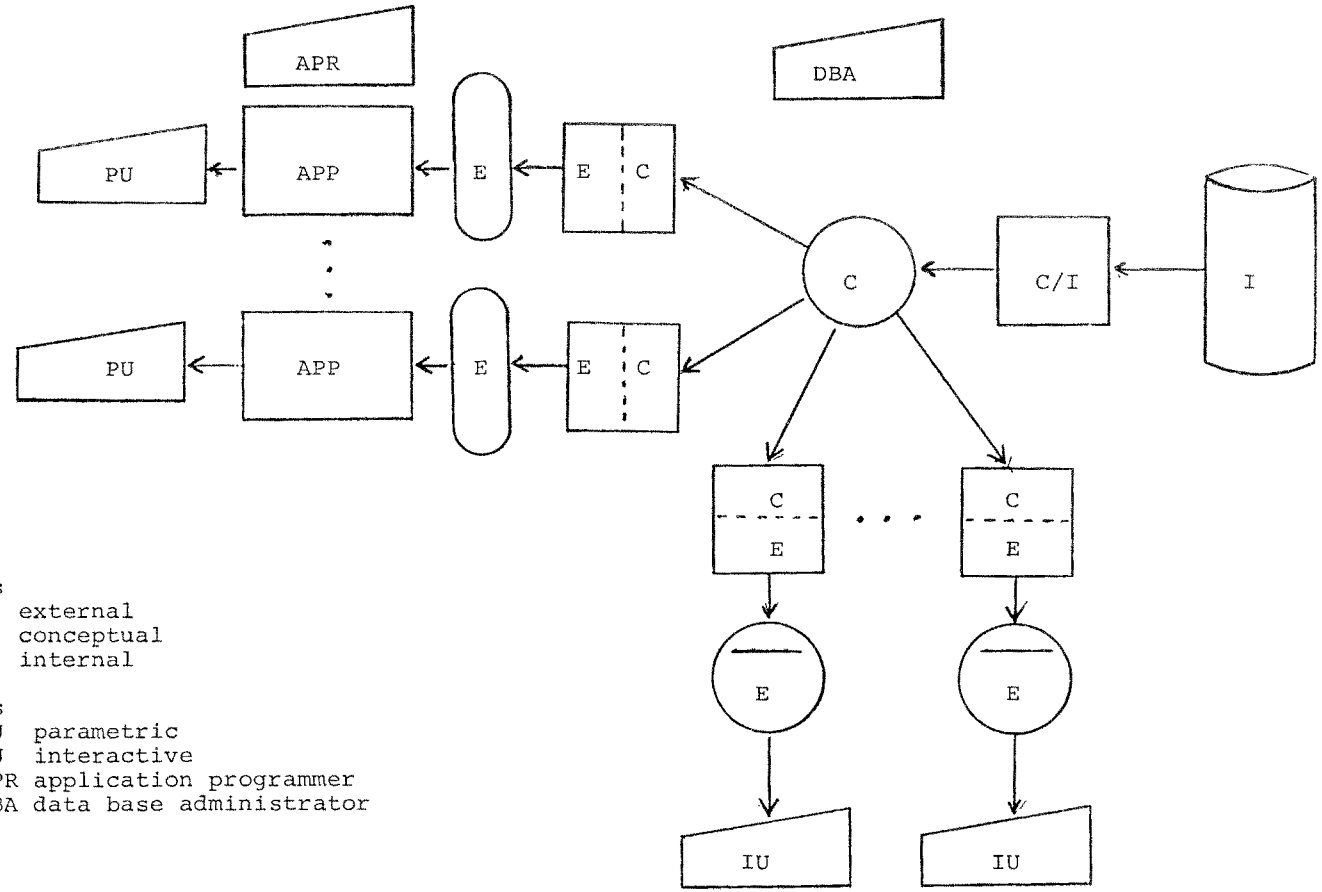


Fig. 1:Structure of a data base system

cific use of the data base and (b) possibly, to transform the selected subview to a view, which is "more natural" for the specific use. Point (a) represents the primary purpose and is of importance for reasons of protection, scheduling, user isolation etc., i.e. essentially for systems aspects.

There are two major end-user groups to consider. First we have the query language user, also called the interactive problem solver and typified by the "non-DP-professional". The query language users view of data is very similar to the conceptual view. He performs data manipulation at a fairly high level without needing help by experts. This is different for the "parametric user", who performs well defined actions with parameters of a simple structure. This user interacts with the system via application programs written by application programmers in some programming language into which a data manipulation language is incorporated as a sublanguage. We talk of a host language - sublanguage relationship. In general, the query data manipulation language is at a higher level than the application programmers data manipulation language.

In practice, large amounts of stored information are extremely unlikely to be used by only one person. It is therefore a requirement to a data base management system that it allows sharing of and concurrent access to the stored information. Concurrency creates a number of problems in connection with system integrity, scheduling, deadlock prevention, recovery, protection, and efficiency in solving all these problems.

While commercially employed systems (like IMS) primarily support applications involving parametric users, research concentrates on support of the interactive problem solver. Correspondingly we find a large number of research activities oriented towards the data model used for the conceptual view and to a single user high level query language system. Sections 2 and 3 are devoted to data models and data manipulation languages to describe this research. In section 4 we will discuss research in the area of system aspects. Section 5 describes contributions of research to implementation techniques such as a storage structures and search algorithms. Section 6 will refer to some of the modelling, measurement and analysis efforts and section 7 will contain conclusions with respect to primary results, recognizable trends and some major problems deserving research.

2. DATA MODELS

The conceptual view has been introduced as the central point of reference in a data base management system. Clearly, such a view should be as close as possible to intuitive notions of information. Proposals for a conceptual view are known as data models. A conceptual data model provides a set of possibilities of how to encode conceptually information which exists in the real world. Of course, the mapping between real world information and conceptual information is not formalized.

Closely connected with the notion of a conceptual data model is that of a conceptual schema. For illustration purposes let us consider an extremely simplified real world situation. We have sets P of professors, S of students, and C of courses. Each of the objects in these sets has a number , which is unique within the set, and a name. Further, we know for every professor which students he advises and which courses are tought by him. A student has exactly one professor as advisor and may attend a number of courses and a course is taught by exactly one professor and attended by a number of students. Fig. 2 shows the information in an attempt to be close to reality without biasing towards any data model.

Conceptual data models are all more or less based on the notions of set theory. One of the earlier attempts is the Information Algebra of CODASYL /34/. Other models and stimulating ideas are due to Mealy /124/, Feldman and Rovner /74/, and, Ash and Sibley /3/. The most successful model in terms of acceptance and as a stimulus for data base research has been developed by E.F. Codd in a series of papers to which we will devote the next subsection.

2.1. Codd's Relational Model (CRM) /39-41, 43/

In CRM information is a finite set of named relations of assorted degree. A n-ary relation is a finite subset of a cartesian product

$$D1 \times D2 \times \ldots Dn$$

where the Di are potentially infinite sets of "scalar" data values such as numerical values or string values. In other words, a n-ary relation is a set of n-tuples. To any relation, the elements of the

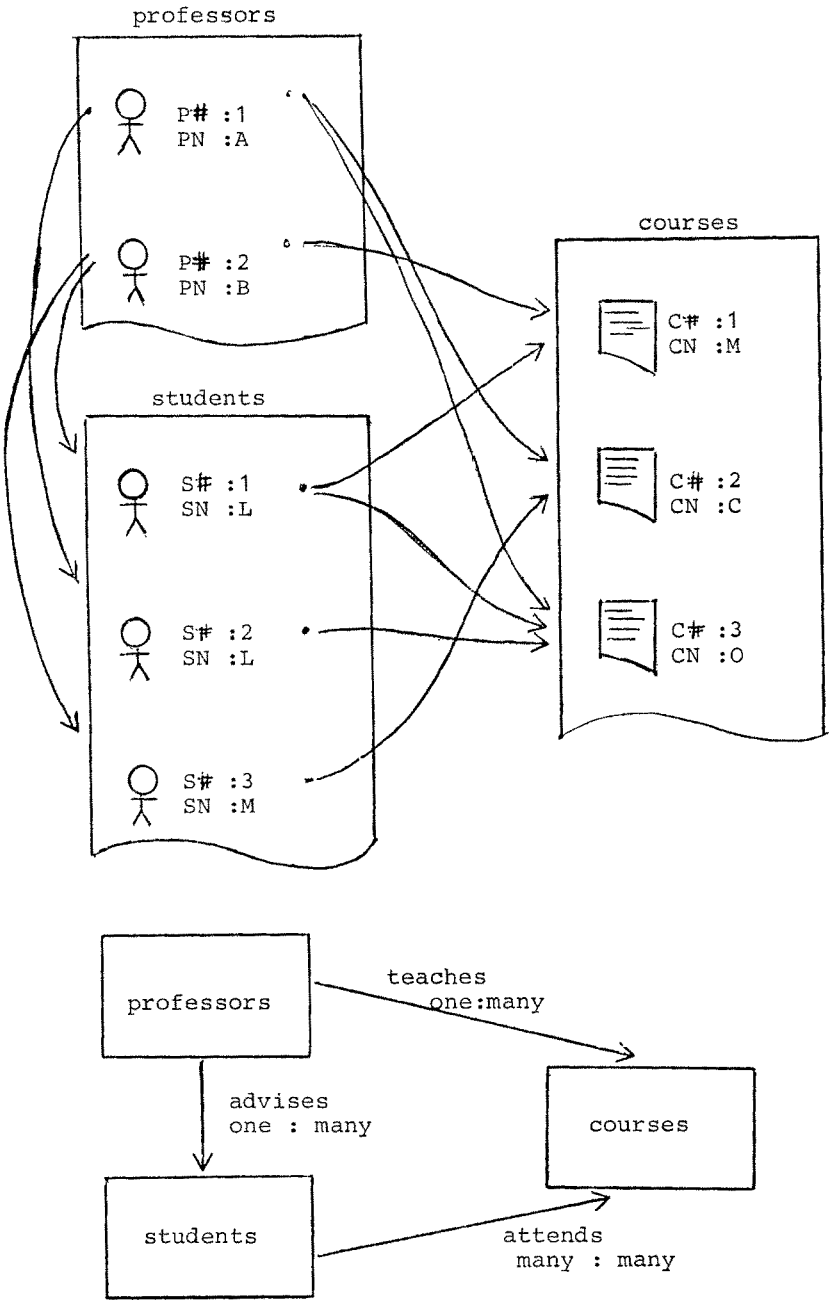


Fig. 2:Example situation and schema

tuples are named with attribute names for ease of reference. A relation is homogeneous, i.e. any two elements of the same relation have the same attribute names associated with them. This allows a tabular listing of a relation as shown in fig. 3, the representation of the example information in CRM.

Some domains within a relation may be keys and/or references. Two distinct tuples in a relation have different values in their key elements. A refernce domain actually is used to refer to some value that is element of another tuple in the same or another relation. For example, P# in P is a key domain, but P# in S is a reference domain which contains only such integer values, which appear as P# in P.

The relations shown in fig. 3 and corresponding to the schema indicated in fig. 4 are all in socalled first normal form. This means, that all elements of a tuple are scalar (and not sets or lists or structural in any other way). This has consequences in the way information can be modeled. Consider the relationship between students and professors. Principally there are at least two ways to store this information:
(a) we can build the set of all students (or their unique numbers) advised by one professor and store this set with the professor tuple or
(b) we store with every student the professor (or his unique number), which advises the student. Case (a) would not be in "first normal form". Fortunately, case (b) is in this form.

The relationships between professors and students or professors and courses are one to many (i.e. <u>one</u> professor advises <u>many</u> students). In these cases we store the converse relation to satisfy normalization. However, the student/course relationship is many to many and therefore requires the introduction of an additional relation, the relation SC. Codd has defined further normalizations, the "second" and "third normal form", which essentially serve to remove some redundancies. The reader is referred to Codd /41/, Date /49/, or Wedekind /192/.

The advantages of CRM are its apparent simplicity and its appeal to those, who are used to "think" in tables, in particular for researchers with a background in the elementary notions of discrete mathematics. Since a relation is a set, set operations like union, intersection, relative complementation etc. can immediately be applied if the relations agree in domain names. More importantly, projection may be

S		
S#	SN	P#
1	L	2
2	L	1
3	M	2

P	
P#	PN
1	A
2	B

C		
C#	CN	P#
1	M	2
2	C	1
3	O	1

SC	
S#	C#
1	1
1	3
2	3
3	2

Fig. 3:Normalized CRM relations

S (S# int, SN char, P# int) key (S#) ref (P# to P.P#)

P (P# int, PN char) key (P#)

C (C# int, CN char, P# int) key (C#) ref (P# to P.P#)

SC (S# int, C# char) key (S# , C#) ref (S# to S.S# , C# to C.C#)

Fig. 4 CRM schema

applied to relations and relations of different structure may be combined using well known methods of composition such as cartesian product or Pierce product, or a generalization called join in Codd's terminology. In a slightly different approach every relation may be viewed as a stored predicate and first order predicate calculus may be applied to define new relations. Codd has investigated both approaches, the "relational algebra" and the "relational calculus" and has shown that they are equivalent /41/.

The relational model has been subject to a number of critical considerations /20, 43/. It is obvious that it does not offer the full range of structures to which we are used within computer science. It has, for example, no equivalent to the hierarchic record organization of COBOL, PL/I, PASCAL or ALGOL68, a structure which is simple, corresponds to intuition and is most frequently used. This is only one important example of a structure violating the "first normal form" condition.

The example of the relationship between students and professors has shown how the schema is affected by constraints. For example, if the one:many constraint of the professor - student relationship is relaxed to a many:many constraint, a completely new relation has to be introduced.

Another consequence of normalization is the fact, that basic information has to be encoded with the help of other information, which is often of no interest to the requestor of the basic information. For example, in order to know the name of the advisor of the student with name "M", the "system" has to learn that this advisor has the professor number 2, since there is no other way to get to his name. This may be considered tolerable, however, more critical situations arise, if we allow a user to see only a subset of domains, which does not contain the key values. For example, a particular user may be allowed to look at every employee's salary and compare it with the employee's manager salary, but for reasons of privacy he is not allowed to look at the man numbers associated with the salaries. There are immediately two problems. First, in order to find a man's manager's salary, the user has to know the man number of this manager, contradicting the privacy constraint. Second, if the man number is projected out it may well happen that two persons with the same salary appear as one tuple in the projection, thus making any statistics on salary distribution in the projection invalid. One can imagine ways around the first prob-

lem though no elegant ways are known to the authors. The second problem is solved in at least some of the experimental systems (like INGRES, SQUARE) by allowing "duplicates in sets". These implementations are actually implementing a homogeneous flat file model as it is described, for example,
by McGee /122/. This model is in turn a special case of the graph model described later in this section. It cannot be claimed that implementations of the homogeneous flat file model find their clean, theoretical foundation in CRM, though some operations, like joins, make sense with both kinds of data structures.

2.2. Graph Oriented Data Models

The common idea behind the data models discussed in this section is the implicit or explicit notion of labeled graphs over entities or named binary relations between entities . The origins of the model in computer science goes back to McCarthy's abstract objects (as models of information) which have to conform to an abstract syntax (as the schema to the model). (McCarthy,J.: Towards a Mathematical Science of Computation. Proc. IFIP Congr. 1962, North Holland, Amsterdam, 1963). McCarthy's work has influenced the activities of a group in the IBM Vienna Laboratories to model the interpreter states during execution of programs (Lucas P. and K. Walk: On the formal description of PL/I. Annual Reviews of Automatic Programming 6, 3 (1969)). The data structures in languages like COBOL, PL/I, PASCAL, and ALGOL68 are adaptations and extensions of this model, in general with restrictions as to what may be specified in a schema. "Schema" appears now as a synonym for "declaration" or "abstract syntax". Some of the data models designers refer explicitely to this origin /1, 56/. Practically all commercially available data base systems are based on the graph model.

Within data base research activities the graph model shows up more or less consciously in a number of papers. McGee discusses a graph model in 1968 /121/. The entity set model of the DIAM system designed by Astrahan, Altman, Fehder and Senko is essentially a binary relational model and therefore a graph model /155/. Other well known graph models are those developed in CODASYL activities and generally known as the DBTG model /35, 37, 38/. We find the graph model also in Abrial's "data semantics" /1/ and many other research papers /20, 56, 63, 158/. Schmid and Swenson employed recently some sort of graph model to es-

tablish a connection between relations in CRM and the real world /151/.

Among the earlier data base research activities the DIAM effort deserves special attention /155/. It contributed essentially to the acceptance of a data base system structure as shown in fig. 1. Its data model, the entity set model, had and still has impact on standerdization activities. The DIAM data model has not been defined with the same mathematical rigor as CRM. This is related to the fact, that its designers stressed the closeness of the model to the real world more than pure mathematical formalism. We will not discuss the DIAM model in more detail here since it can be mapped in a straightforward way to the subsequently described graph model and as such find a clean mathematical foundation.

The essential notion of the graph model is that of an abstraction of objects as nodes in a graph. To be consistent with the most frequently used terminology we call such a node an entity. An entity may be anything "that has reality and distinction of being in fact or in thought, e.g. objects, associations, concepts, and events" /34, 155/. Some entities have unique denotations like
5, 7 or 'ABC'. Other types of entities can only be uniquely identified with the help of relationships between entities.

Information in the graph model is stored as a finite set of named finite binary relations between entities. Since entities are nodes in a graph, relations can be represented as directed labeled edges in the graph. To represent unary relations (which are sets of entities) we assume a given node as the entry node to the information and interprete the binary relation between entry node and any other node as a unary relation. For simplicity in drawing a graph we represent edges from the entry node to other nodes by labeling these nodes with the relation name. Fig. 5 and fig. 6 show the graph model representation and schema for our example.

A first advantage of the graph model over CRM is the fact that only binary relations are used. This removes the need to distinguish between the domains via symbolic domain names. More importantly, due to its explicit notion of entities, which is not present in CRM, it is clean and clear in the mathematical sense without the need to deviate from it for "practical" reasons of convenience. Like for CRM, it is possible to develop calculus or algebra oriented languages with the

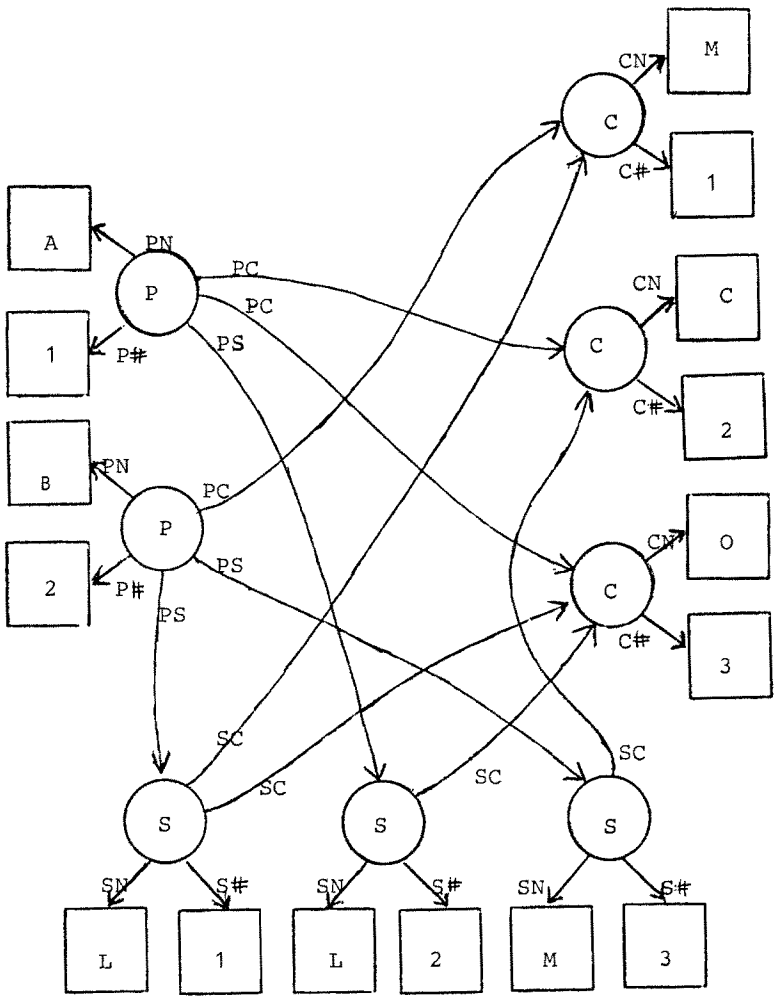
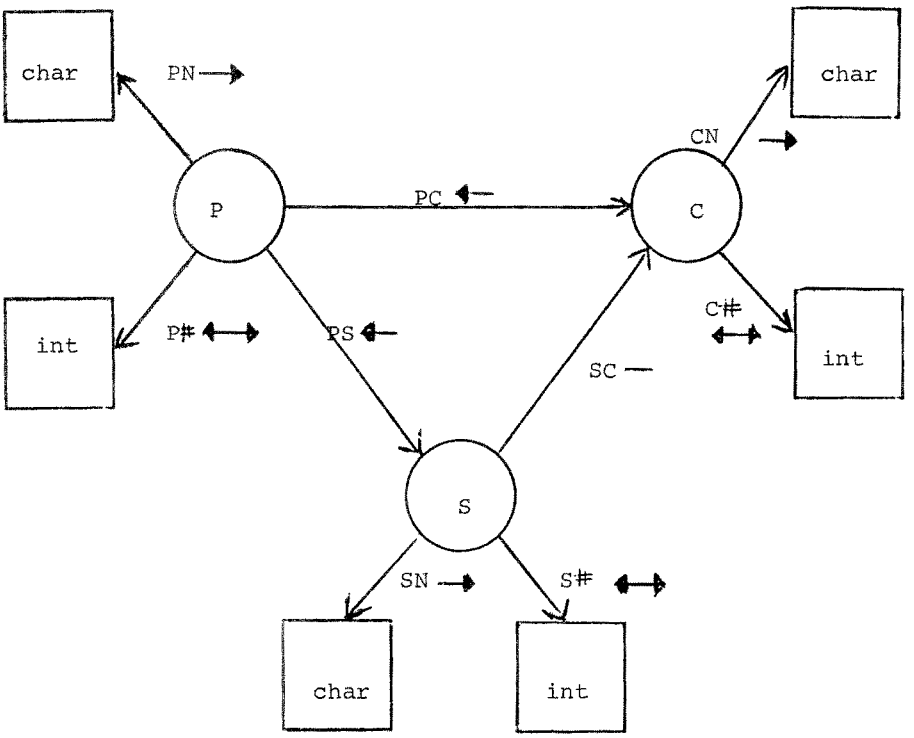


Fig. 5 :Information as a graph



R ⟷ one : one

R ⟶ many : one

R ⟵ one : many

R — many : many

Fig. 6:Schema to the graph model

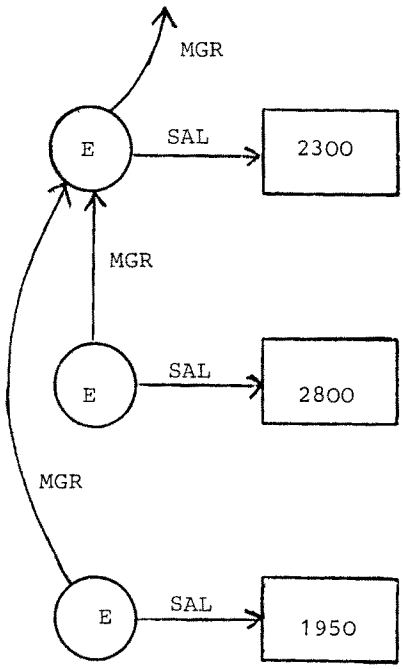


Fig. 7: Subgraph of E, MGR and SAL

same rigor /128/. On the other side it does not provide difficulty to furnish a user with a subview of the data base (i.e. a subgraph) which restricts to the relationships, which may be seen by the user. See fig. 7 for an illustration. Since practically all known structures in computer science can be mapped conveniently to some form of graphs, it does not force us to exclude these structures from our high level data modeling.

2.3. The Equivalence of Data Models

It is not at all surprising that the different models are equivalent in the sense that information encoded in the DBTG or DIAM model can be encoded in CRM and vice versa. Moreover, in most cases there is a simple and straightforward way to convert a schema in one model to a corresponding equivalent schema in the other. The question of choice between two different models must be decided on how "convenient" or "natural" processing becomes in the models. This is, however, not only a question of the data model but also a question of the data manipulation language. In the next section we will therefore come back to the question of equivalence.

The question of equivalence of data models has been investigated by Bobrow /17/, Neuhold /134/, Sibley /167/ and McGee /122/. Different models are likely to coexist for a while (at least in the world of researches), even on the same system. This creates a new mapping problem, namely of how to superimpose a model A on a model B, a problem, which creates the need for a "superimposition theory" as it was stated by E.F. Codd /43/. First results in this direction are reported by Frasson /82/.

3. DATA MANIPULATION LANGUAGES

3.1. Low Level Versus High Level Logic

As we can see in fig. 1, data are accessed in external form either via an application program or interactively at a terminal. In the first case, records are typically retrieved one by one and processed sequentially in a programming language. This type of processing is referred to as "low level" or as "one record at a time logic". Typical for the second case of access is the higher level "multiple records at a time

logic". Research activities are primarily oriented towards the higher level logic. It is important to realize that also in the application program case "multiple records at a time logic" is required to specify in advance on which subset of the data the program is going to operate. The system needs this information for scheduling and resource allocation purposes. In effect, the selection of the subspace to a program is a mapping between the conceptual and the external view, which by the nature of its use has to be specified in a high level logic. Even though research projects are primarily oriented towards interactive access to data and even though their implemented systems are modest compared to commercially available systems, their results may very well be of relevance for the type of processing through application programs as it is still more common in todays user installations.

Subsequently some of the data manipulation languages developed by researchers will be referenced. We start with the CRM implementations, then we continue with languages based on other data models. A special subsection is devoted to languages which are characterized by the way in which they are used. Finally we will come back to the equivalence of data models.

3.2. Some CRM Implementations

Table I lists some of the experimental systems, which claim to implement CRM, though for some it would be more correct to claim that they

System	location	remark	reference
IS/I	IBM UK	algebra	Todd
MacAims	MIT	algebra	Goldstein
RDMS	MIT/MULTICS	algebra	Steuert
MORIS	Milano	calculus	Bracchi
SQUARE	IBM Research	mapping	Boyce
SEQUEL	IBM Research	mapping	Chamberlin
INGRES	Berkeley	calculus	Held
ZETA	Toronto	definitional	Mylopoulos
DAMAS	MIT	calculus	Rothnie

Table I. Some relational systems

implement homogeneous flat file. Of the nine systems shown, the first four represent early experiments. SQUARE is a language based on the concept of "mapping", an approach which is somewhere in between relational algebra and relational calculus. It is implemented on top of XRM, a data management supporting n-ary relations or, better, homogeneous flat files /111/. XRM in turn is implemented on top of RAM, a data management, which supports stored binary relations and resembles the graph model /110/. SQUARE has a compact syntax. The query language SEQUEL is derived from it with more English keywords. INGRES stands for a system, which offers QUEL and the graphics oriented CUPID as enduser interfaces /93, 119/. ZETA is a system which is currently being developed at Toronto. It has a data management supporting relations and provides a "syntax directed" definitional tool to let the user implement his high level query language on top of low level primitives. DAMAS is a system specifically used by its implementor to study an optimization aspect of data access.

To give an impression of the different styles of query languages, let us consider the following query:

> What is the name of the advisor of the student, whose name is "M"?

In IS/1, the relational algebra approach, we obtain:

```
( (P * (S; C2 = 'M' )); C1 = C5) % C2
```

This expression is a sequence of a selection (operator = ';'), a cartesian product (operator = '*'), a second selection and a projection (operator = '%'). Ci refers to the value in the i'th domain.

In QUEL, the calculus oriented query language to INGRES, we obtain:

```
RANGE OF PROF IS P
RANGE OF STUD IS S
RETRIEVE INTO R(PROF.PN) WHERE PROF.P# = STUD.P# AND STUD.SN
= 'M'
```

Here the answer is in the result relation R, a unary relation. Clearly PROF and STUD are variables in the predicate calculus sense over which existential quantifications are applied by default.

In SEQUEL, the "mapping" approach, we obtain:

```
SELECT PN FROM P WHERE P.P# IN
SELECT P# FROM S WHERE S.SN = 'M'; ;
```

All of the nine systems represent what might be called first generation research data base systems. This means that their contribution to the solution of data base problems, though already significant as pointed out above, may be increased by follow-on development. At least for the three systems SEQUEL, INGRES, and ZETA we know that such ongoing research is planned. The follow-on system to SEQUEL is called System R.

3.3. Some Non-CRM Systems

As already mentioned, most research activities are using CRM as their data model. In this subsection we will discuss some languages which are using a graph oriented data model.

First there is DIAM with RIL (Representation Independent Language) as its data manipulation language /72/. The DIAM work continues in an effort called DIAM II with FORAL as its query language /157-159/. Senko's FORAL has the interesting property of a graph (or binary relational) model oriented query language for the composition of relations between entities. The example query of the preceding subsection can be formulated in FORAL as follows:

```
P(PN) where for PS SN = 'M';
```

FORAL establishes the connection between professors and students with a single identifier where IS/1 or QUEL need at least one comparison. A recently described system, developed in Nice, implements the graph model on top of IMS and offers a query language with similar advantages /82/. McGee describes a data manipulation language to a conceptual graph model, which is very similar to the DBTG model /123/.

A very interesting research developed system is SIMS /194/ which offers a query language and a data definition language. The data definition language allows to map data given in their internal form and possibly generated on another computer, to a hierarchical conceptual form. This hierarchical form can then be accessed by the query lan-

guage without actually converting the data. SIMS meets with these features objectives, which are missed by most other experimental systems, though SIMS is one of the earlier implementations.

Report generation, i.e. the design of layouts of computer generated reports is a non-trivial problem which to solve with the help of a computer seems natural. Dana and Presser report about an interesting high level language specifically designed for this task /46/.

3.4. User Interface Aspects

In this section we will discuss some data manipulation languages whose designers apply a specific technique with respect to the interface to the user.

A series of research efforts has as its target to embed the query language into a general purpose programming language to combine the data access with powerful computational facilities. Two of these efforts have as their specific research goal to develop and study protection mechanisms /44, 75/. Earley describes a proposal for the inclusion of CRM data structures into an ALGOL like language /59/. Schauer proposes to build an interactive CRM query language in a data base system for the evaluation of measurement data on top of APL /149/.

A question, which is currently still open, is whether the traditional way of defining rigorously a formal language is the best way to attract all end-user groups to the computer. Some researchers believe in the possibility that a freedom in syntax as offered by a natural language, might make the computer more attractive to at least some user groups. Codd proposes such a natural language system, called RENDEZVOUS, which is currently being implemented /42/. TORUS is a natural language system being developed at Toronto. It uses ZETA as its data management /131/. Thompson, Petrick and Kraegeloh report about experimental language systems, which are already implemented /184, 147, 102/. Further references to systems natural language and the linguistic approach can be found in /156/.

The feasibility of the "communication with the computer in natural language" is subject to rather sceptical considerations. In the case of data manipulation languages many of these considerations are not applicable, since the "universe of discourse" is essentially restrict-

ed to the objects and verbs stored in the data base and described in a simply structured data dictionary.

A completely different approach is taken by Zloof, McDonald and Schauer /198, 119, 149/. Their method requires a display device, which is used to display the description of the stored CRM relations in some graphical form. In Zloof's Query By Example the user fills "examples" into free spaces of the relation description. Simple queries can be formulated easily and with a low probability of error. In McDonald's CUPID, the user has to draw a flow diagram like picture (with the help of a menue) which expresses the semantics of the query. Schauer's extended query by example is an extension and modification of Zloof's method. It is natural to use a display device if information is associated with geographic locations. GADS /25/ is such a system in which a user can point to locations or subareas within a displayed map to obtain information related to the graphic entities.

The question, wether one user-interface oriented approach is more successful than another cannot be answered by abstract reasoning. Investigations are under way, which employ the methods of experimental psychology to find unbiased answers to the posed question /143, 183/. One of the reported experiments seems to indicate that questions of syntax (or more generally of the form as opposed to the contents) are of a slight significance for the unskilled while questions of semantics are significant independent of the users skill /143/.

3.5. Data Model Equivalence

As pointed out earlier, and illustrated by examples, we know that different data models are (or "can be made") equivalent with respect to corresponding schemata. Subsequently we will briefly indicate that equivalence can easily be extended to the equivalence of the query languages. To this end we introduce informally two query languages, one (CRM) for CRM and the other (GRAPH) for the graph model. Both languages are very similar to SEQUEL. In CRM we deal with relation names, attribute names and variables. A variable is denoted by a relation name followed by a period followed by an attribute name.

Example

S relation name
SN attribute name
S.SN variable

In GRAPH we deal with sets (unary relations) and relations (binary relations). A set denotation is a set name or a set name followed by a period followed by a relation denotation. A relation denotation is a relation name or a relation name followed by a period followed by a relation denotation. A set denotation may also be used as a variable with the obvious meaning that the variable "runs" over all elements of the set.

Example

S, S.SC, S.SC.CN sets (or variables)
PS, PS.SC, PS.SC.CN relations

It should be noted that GRAPH is recursive in the definition of sets while CRM is bound to two levels.

The period is in both languages used as the operator for functional composition from left to right.

A query is of the form:

SELECT list1 FROM list2 WHERE predicate;

In CRM list1 is a list of attribute names, list2 is a list of relation names, and the predicate is over variables which can be built starting with the relations in list2.

In GRAPH list1 is a list of relation denotations, list2 is a list of set denotations and the predicate is over set denotations which can be built with the help of relations starting with the sets in list2.

In both languages the use of subscripts may be necessary to avoid ambiguity. The subsequent examples are such that ambiguity does not arise.

Query 1

Name of the professor, who advises student M.

CRM:

```
SELECT PN FROM P,S WHERE P.P# = S.P# and  S.SN = 'M';
```

GRAPH

```
SELECT PN FROM P WHERE P.PS.SN = 'M'
```

This simple query illustrates already the essential difference between the two data models. CRM normalization requires that some logical relationship between entities are encoded with the help of unique porperties of these entities while in the graph model these relationships may be used directly. Therefore, CRM has to make a comparison where GRAPH simply uses functional composition as we do in natural language. This will become even more apparent in the next query.

Query 2

Names of courses attended by students which are advised by 'B'.

CRM:

```
SELECT CN FROM P, S, C, SC WHERE P.PN = 'B' and  P.P# = S.P#
and  S.S# = SC.S# and SC.C# = C.C#;
```

GRAPH:

```
SELECT PS.SC.CN FROM P WHERE P.PN = 'B';
```

The brevity and elegance of the GRAPH form compared to the CRM form should, however, not be used to conclude an essential superiority of the graph model over CRM. In fact, it is possible to extend the CRM language with a macro processor which accepts as definitions relations between entities in terms of their CRM encodings. This macro processor can then accept GRAPH queries and convert these queries into CRM queries in a simple straight forward algorithm. With other words, we can implement the GRAPH language on top of a CRM implementation such that the user has all the advantages of the graph model. The differences

of the languages and their underlying models appear on a level which is primarily of a syntactical nature since they can be transformed away with the help of syntax macros. Issues of one data model versus the other are of little practical relevance given the right sort of implementation. Many other questions like those discussed in subsequent sections deserve and are in the process of receiving more attention.

4. SYSTEM PROBLEMS

4.1. Introduction

The major problems in a data base system like IMS are connected with concurrent access to data shared by many users, with application program management and scheduling, with system enforced data integrity, with locking and recovery or error isolation, with data independence and last, but not least, with high enough transaction rates and short enough response times to make the whole system attractive for the user.

The implementation of such a system, even for experimental purposes, may turn out to be quite costly in time and manpower. It is therefore natural that only few research projects aim at a large portion of the full set of data base management system functions. Among the systems mentioned in section 3, DIAM in its original conception was at least very ambitious with respect to data independence and supported storage structures. System R, the follow-on activity to SEQUEL, though being experimental, plans to provide solutions in nearly all of the problem areas mentioned above.

The fact that researchers so far have not developed full size operational data base systems does not mean that they have ignored problems outside data models and high level query languages. In subsequent sections we will reference a considerable number of relevant papers in the area of data independence and data integrity and recovery in connection with multi-user systems. In addition, the reader will find references to research in the area of security and authorization in the attached bibliography.

4.2. Data Independence /175/

A data base system supports data independence to the extent in which it allows transformations of the internal or conceptual forms of data without affecting (a maximum of) existing programs in the sense that non-affected programs, which run correctly before the transformations, run also correctly after the transformations. Hence, data independence is in effect the independence of programs with respect to data transformations. This makes clear that data independence is not the automatic consequence of the selection of a certain conceptual data model as it is sometimes claimed.

We distinguish between internal and conceptual data independence. The need for internal data independence, i.e. independence of application programs with respect to changes of the internal form of the data, while the conceputal form stays invariant, is a consequence of the widely recognized fact that there is no absolutely best internal data organization /182/. The performance of an application program depends heavily on how many of its access paths are directly implemented (for example via links or inverted files or other storage structures; see section 5). Every such direct implementation means redundancy in storing, i.e. requires additional update activities. The data base administrator will attempt to optimize the internal organization for a given mix of application programs. Since the mix changes with time, there will be a need to adapt the internal data organization.

The need for conceptual data independence arises due to additions of new types of information, or more generally, changes in the conceptual schema. In general, at least some of the existing application programs are affected, while still a large part may remain unaffected. Consider, for example, the addition of a domain to a CRM conceptual data base. There should be no need to alter programs, which only read data, since the old model is a subview (projection) of the new model. This may already be different for programs which update information, since there may be a dependency between the changed domains and the new domain. Certainly, at least some of the programs which insert new elements into relations are affected, since otherwise the new information cannot be entered. Other changes in the conceptual model, for example, relaxing the many to one constraint of a binary relation to a many to many constraint, may even affect read only programs, if these programs are designed such, that they rely on the old many to one constraint.

Support of conceptual data independence requires that the system is capable of determing for each of its application programs, whether it is affected or not. This involves a very complex decision problem, which is not solvable in general. It is therefore necessary to restrict the data mapping languages such that the decision problem remains solvable. This requires a type of theory, which has not been extensively applied /exceptions appear, in other contexts, in 53 and 65/.

Support of internal data independence requires the following:

1. A data definition and mapping language, which specifies the internal form to every conceptual form allowed by the schema. The degree of data independence to a given conceptual schema is the set of different mappings supported for the conceptual schema.

2. To any given application program the system must be capable of recognizing the predefined access paths in the internal schema, which meet "optimally" the program's needs, and exploit these access paths during execution of the program. This process has been called reduction of the external access to the internal access. A system without this optimizing reduction may be "logically" data independent but practically serves no purpose . In other words, what is necessary is a data independence which results in performance gains for the user.

Almost all of the experimental query language implementations described in section 3 support data independence to a limited degree in the following way: When a user (in his "data base administration role") introduces a new relation he may specify which attributes should be inverted. However, a query is formulated independently of whether there exist inversions or not. During execution of a query the system exploits the advantages offered by inversions and maintains the inversions without any burden on the user except unavoidable storage and time overhead /175/.

A more comprehensive approach to data independence starts with the development of a flexible data definition and mapping language. Taylor, and with a slightly modified motivation, Smith have developed such languages for a data model very close to the DBTG model /179,

169/. As pointed out earlier, SIMS has also such a language, which enables it to operate on given data without converting these data as a whole /194/. The importance of the possibility to access data by means of a description and without converting the data as a whole is illustrated by the existence of data collections, which to convert to a standard form is more expensive than rewriting all the application programs operating on these date /166/.

The evaluation of a data definition and mapping language in a full size data base management system is a very complex task. This fact has probably given impetus to experiments with such languages in the area of data translation, which has also a justification in its own right. Data translation is the conversion of data, which have been created and processed in one system, to a data organization which allows processing in another system. The orientation of these projects combines practical orientation with experimental evaluation of the power of mapping languages, while the projects still remain small enough to be conducted in less than a large group.

Ramirez et al. have built a compiler, which generates conversion programs from data descriptions /142/. This work makes use of the mentioned data definition and mapping language developed by D. P. Smith . Similarly, Taylor's language is used in another major activity at the University of Michigan / Merten, Fry, 126/. This work is continuing with increased functions being built into the currently runnning prototypes. The underlying data models in both projects are DBTG oriented. The usefulness of CRM as internal form of data during translation has been investigated by Navathe and Merten /133/ with a negative result. Liu and Heller have used contextfree grammars as data descriptions at the record level /108/. Housel, Lum and Shu have developed a language DEFINE (mapping to a hierarchical structure) for data definition and a language CONVERT (mapping between hierarchical structures) for translation definition and plan to implement these languages in a prototype /95, 165/. Again their model of data is that of a graph, which, for the purpose of translation, is decomposed into hierarchies. In a network of computers, such as the ARPA net, data conversion is of particular importance. Su and Lam, and also Schneider and Desautels describe an approach to data translation specifically oriented towards this use /177, 153/.

4.3. Data Integrity and Recovery in Multi User Systems

Though the problems, which exist with respect to data integrity and recovery are also present in single user systems, they are enormously increased in a system with many concurrent users. In fact, without multi-user support, traditional means under user responsibility may be adequate for dealing with these problems. In a multi-user system the system has by necessity to take over some of the responsibility for the solution.

The notion of data integrity is closely connected with the notion of consistency and the schema. A schema may be viewed as a collection of assertions about the data base contents which stay invariant during processing. These assertions are also called consistency rules. Such rules may state that information is in a certain sense complete (for example, whenever something is known about a person, then its mannumber, name, address and birthdate are known). A more complex rule may require that a person cannot be its own ancestor, or, that the sum of different expenses in a department may not exceed the budget allocated to the department. A data base supports data integrity to the extent in which it allows a user to specify consistency rules, which are subsequently enforced by the system.

A straightforward approach to specifying such rules could consist of the following.

1. Provide the user with a general language like predicate calculus or a query language to specify assertions.

2. Whenever the data base has been modified, the system checks, whether the assertions still hold.

This approach, proposed for example in /1,78/ and recently also in /66/ has to be considered with caution. First, it is for a general language undecidable whether the assertions are in themselves consistent. Second, it has to be carefully defined when the consistency of a data base is checked, since a user must in general perform a number of modifications to a data base before a consistent state is again transformed into a consistent state. Third, checking of a set of complex consistency rules may require access to a large portion of a data base, which can range from hours for small data bases to several weeks for larger data bases in processing time.

The first problem can be solved only, if the language in which the consistency rules are expressed, is simple enough so that the consistency of the rules remains decidable. Practical systems, like IMS, certainly satisfy this criterion.

The second problem has been recognized and has lead to the introduction of the notion of a transaction /65, 66/. A transaction is a sequence of transformations of the data base by one user, which are supposed to transform a consistent state into a consistent state. Beginning and end of a transaction are under user control. Now consistency checking can take place whenever a transaction is complete.

The third problem is in some way connected to the first one. Given a consistency rule, the system must be capable of determing, how costly its checking during data base transformations will be. The variation of costs of checking a consistency rule may be illustrated with an example: Given a father relation, a consistency rule may state that the subgraph containing only edges labeled with father is cycle free. Checking this rule requires an algorithm that is in execution time proportional to n**3, where n is the number of objects participating in the father relation. If the stored information contains in addition for every person the birthdate, the rule that the birthdate of the father precedes the birthdate of the son, serves the same purpose as the previous cycle rule. This rule can, however, easily and efficiently be verfied for every data base change. In most situations "system enforced integrity" makes only sense, if the time needed for the enforcement is bound by a linear function of the time necessary to perform processing without integrity assurance. A compiled approach like the assurance of integrity constraints by query modification, as proposed by Stonebrecker /176/, may help since it allows comparative analysis of the queries with and without constraints. Of course, such an analysis does not have to be at the source level.

The problem of integrity is increased with concurrent access by more than one user. The system has, in addition, to ensure that the users do not interfere with each other during update operations. To this end the system must provide a facility which gives a user exclusive access to a part of the data base for a limited time. Basic mechanisms for granting exclusive access to a user are well known from operating systems under the names of locking or semaphores.

The situation is analyzed in a technical report by Eswaran et al

/65/. A complication of locking in data base systems, also explained in /65/, is the need to lock objects, which may not yet exist, from being created /30/. There are an infinite number of objects which may potentially be created (though the set of created objects is always finite). Such locks may be described by predicates with an infinite extension. Performance requirements dictate that it can be decided for two such predicates whether they overlap. This imposes restrictions on the formulation of predicates to be handled by the system /65/.

Locking has as consequence the danger of deadlocks. As in operating systems there are essentially two ways to deal with deadlocks. The first, proposed for example by Everest /67/ is preclaiming. With preclaiming of resources the system can schedule the user's transactions such that no deadlock appears. The second solution is preemption, i.e. taking away resources from one process to give the resources to the other process. The preempted process has then to be positioned back to a state in which it did not hold the resources. This is possible with the help of journals and checkpoint files, i.e. data sets which record the internal state of a process during its execution /83/. This method is discussed by Chamberlin et al /29/. It should be noted that these files are required in most systems for recovery purpose.

Recovery is necessary whenever it is impossible for a transaction to terminate normally. The cause for this may be a deadlock, a logical error in the user's program (a zerodivide exception or a consistency check failure, for example), a hardware failure, or the failure of a transaction which has directly or indirectly (via the data base) delivered input to the transaction. The first objective of recovery is the isolation of a failure, such that an error does not propagate in the data base. A second objective is to restart all transactions which have been affected by a failure without being the cause such that they continue execution as if no failure had appeared. This is to a large extent possible. Recovery algorithms have been described by Genton /83/, Davies /50/, Bjork /15/, Edelberg /62/ and Sayani /148/. The basis for much of this work has been laid by recovery in operating systems such as MULTICS /81/.

All solutions to the integrity and recovery problems must of course avoid placing an unnecessary burden on the user. The most that should be expected from an application programmer is to inform the system of the beginning and the end of transactions. The interactive problem

solver should not be required to know about transactions. He should, as far as the query language is concerned, be able to act as if he were the only user of the system.

As with data independence, the problems discussed in this section are at the beginning to be understood. There is certainly significant room for improvement over proposed and existing solutions, in particular, in providing the functions with improved performance.

5. STORAGE STRUCTURES AND SEARCH ALGORITHMS

Data independence as discussed in the preceding section derives its value (a) from the existence of storage organization techniques which reduce the sometimes enormous search time required otherwise and (b) from the existence of algorithms which allow to utilize the storage structures without binding the programs to these structures. The next two subsections are devoted to these two topics.

5.1. Storage Structures

One of the frequently employed techniques is the acceleration of a search with one parameter with the help of an inverted file. If the inverted file is repeatedly inverted we obtain an hierarchical index organization described by Bayer and McCreight and known as 'B-Tree'. B-Trees allow a logarithmic search time for retrieval, update and insert /9/. Lum introduced multi-attribute indexes which allow quicker answers to queries of higher complexity /112/. Finkel and Bentley describe an extension of binary trees to quad trees supporting logarithmic searches with two parameters /77/. Haerder describes methods of address list compressions ('bit lists') to reduce the storage costs of indices in certain cases /90/.

Another method of reducing search time is hashing. Hashing has been extensively studied in connection with its application to data management /113, 115/. Ghosh and Lum have recently shown that under their assumptions, 'hashing by division' is in general best /85/.

Of course, there are a number of additional, essentially basic techniques applied to data organization such as links between or the splitting of records. These methods may be combined in various modifi-

cations. Storage structures have been extensively studied in the past and are well described in textbooks and surveys /54, 99, 100, 118, 120, 135, 160/. The problem remains to offer this richness of structures to programs without binding the programs to specific structures, i.e. to offer the structures with data independence. In the case in which the program does not know the internal organization, it is the system's responsibility to utilize the storage structures optimally. Attempts to solve this problem are discussed in the next subsection.

5.2. The Reduction Problem

Reduction is the problem of reducing external accesses to internal access, where the relationship between internal and external representations are given by mappings of these forms to a conceptual form (fig. 1). Reduction is something like an "optimization" with the primary objective to reduce the number of accesses to secondary storage during execution of a query or an application program. The term "optimization" should not evoke unrealistic expectations; the problem is too complex and is loaded with similar problems as optimization in a compiler.

Variations in the objectives of optimization are connected with the handling of intermediate expressions over data sets. Consider, for example, the expression

(A op1 B) op2 (C op3 D)

where A, B, C, D are large relations and op1 to op3 are operators in the relational algebra. A straightforward evaluation might construct two intermediate relations AB = A op1 B, and CD = C op3 D and then evaluate AB op2 CD. Such an algorithm requires in additon to enormous amounts of secondary storage accesses also an enormous amount of auxiliary storage, which may by far exceed the storage occupied by the underlying relations A, B, C, and D. On the other hand, there are drastic improvements in the evaluation of some queries if during their evaluation at least some temporary inversions can be built. Most of the research in this area is primarily oriented towards the interactive use of a data base of modest size and consequently assumes that auxiliary data sets (i.e. indices) can be built temporarily for one query execution.

One of the earliest comprehensive investigations into this problem is due to Palermo /140/. Palermo claims that for the investigated type of queries no tuple has to be accessed more than once. This is achieved by building indices and restricting the domains of variables in calculus expressions and applying a "least growth principle" for the sequence of operations. A reduction algorithm applicable to the DIAM system is described by Astrahan, Ghosh and Senko /5, 84/ . Greenfeld and Rothnie look at the problem of handling quantification in calculus expressions efficiently /89, 147/. Another implemented version of a reduction algorithm is described by Astrahan and Chamberlin /6/. Their problem consists of primarily taking advantage of inversions, but their algorithm involves also the construction of intermediary lists (indexes) by merging of inversions.

A paper related to the reduction problem is due to Wong/Chiang. They assume that each query is a boolean expression over elementary queries. In this case the data base can be organized according to the elementary queries and reduction becomes essentially the problem of putting a boolian expression into some standard form /195/.

CPU usage has not received much attention, perhaps under the assumption that CPU time is not the bottleneck in the system. This assumption is, however, not always valid. To reduce CPU time, less dynamic and less interpretive search stategies are required, which can be compiled into a CPU efficient search module or access module per application. As mentioned earlier, the compiler approach has been taken or proposed by several researchers primarily, however, for other reasons than efficiency /Mehl, Fernandez, Conway, also Taylor in 125, 75, 44 and 180/.

It should be clear that the reduction problem is very complex. Every algorithm described above has to make a number of assumptions with respect to system structure, which are not generally valid. This has to be so, as long as there is no generally respected data base architecture. Questions, which potentially deserve more attention in future research are the recognition of constraints in storage requirements and CPU time in addition to "minimizing" secondary storage accesses.

6. MODELLING AND ANALYSIS

Research in the area of modeling and analysis has as its objective to

learn about existing systems by analyzing their behaviour and to develop simple probabilistic models for the components of a data base management system. Such models may help to predict the influence of changes in a system or system design. Thus the designer of a data base management system and even more, the data base administrator should have primary interest in these research activities.

The need for modeling of data management system and data base systems has been recognized early by Senko and his colleagues and lead to the development of an analysis tool called FOREM and a follow-on tool called PHASE II /154, 138/. Haerder has recently performed a comparative analysis of current indexing techniques using these tools /91/. FOREM and PHASE II are useful to evaluate storage organizations, but limited with respect to overall data base system simulation. Nakamura et al. report about a data base simulation model which they have constructed with the help of a conventional simulation package /132/. Their model is a fairly detailed, event driven simulator of the processes in a data base management system. These processes are so complex that questions of simulation performance may become critical. A possible way out may be the development of comprehensive data base system simulating tools. A step in this direction is proposed by Reiter /144/.

Data base systems have also been objects of analytical modeling activities though they are clearly too complex to be analytically treatable as a whole. Analytical studies restrict themselves therefore to well defined parts of the system. FOREM is an example of a deterministic, analytical tool for the analysis of storage structures. The methods proposed by Cardenas in /22/, Yao in /196/ and Wedekind in /193/ are also essentially deterministic.

To mention is also the analytically tractable queueing model of the DL/I component of IMS developed by Lavenberg and Shedler /103/. Though they allow for "rather general" distributions, their model is at a gross level. For example, it does not explicitly represent the physical storage organization and total I/O is represented by a single server queue. Extensions of the model are, however, likely to make simulation necessary.

Perhaps the most frequently investigated question is the selection of indices to a flat file. Authors, who have contributed to research on this problem under varying assumptions are Lum and Ling /114/, Palermo

/139/, Stonebraker /174/, King /98/, Cardenas /23/, Schkolnick /150/, Yue and Wong /197/ and Farley and Stewart /71/. Shneiderman has investigated the question of index size at different levels /164/.

Data may be allocated to or destributed over a variety of categories: first, data have to be allocated within a storage hierarchy in an attempt to balance between costs and access time, second, data have to be assigned to physical devices to minimize contention given their position in the hierarchy, third, in case of a network (like the ARPA net), data have to be assigned to nodes in the network to improve accessibility and reduce line costs. Lum et al. as well as Buzen and Chen have considered the problem of allocating data within a storage hierarchy, given statistical information on the usage of the data sets /116, 21/. Lum et al. specify a total cost function to an allocation and an algorithm which finds the allocation to a minimal cost. Buzen and Chen's model takes in addition queueing effects at the hierarchy levels into consideration. Their algorithm's target is to minimize response time under given storage constraints.

The second problem, minimizing disk arm contention by suitably distributing data sets over a number of disk drives given their usage statistics, has been considered by Chandra and Wong and recently also by Easton and Wong /31, 60/. There is no best algorithmic solution, but heuristic approaches are given and some bounds for the optimality are derived.

Casey and Chang have considered the third problem of allocating data within a simplified network of computers to reduce line costs /26, 27, 32/. Chang has extended Casey's linear cost functions to a more general function. Both specify algorithms, which attempt to minimize line costs.

With the analysis work reported so far, at least one question remains open: what are the characteristic input data? Or, with other words, how is the workload of a data base system statistically characterized? Nakamura et al. in their simulation raise the further question of the validity of their model. Answers to such questions can only be found by actually observing operational systems and collecting statistics. Rodriguez and Hildebrand describe how relevant data ranging from a log of the user's messages, over a trace of the application program calls (to the data base system) to a trace of physical disk address references can be collected in operational systems /145/. Lewis and Shedler

derive from such observations that the interarrival times between transactions can be satisfactorily modeled by a non-stationary Poisson process (i.e. a Poisson process with a time dependent rate) /107/.

In a semi-empirical approach, Ghosh and Tuel, and also Easton, determine the parameters of a theoretically established model to make the model fit to empirical data /86, 61/. Ghosh and Tuel model certain interactions in a data base system by linear relationships and determine the coefficients by comparison with measurements, which are also used to validate the model. Easton has proposed to use an extension of the independent reference model for the sequence of references of application programs to blocks on secondary storage and again has validated this model by comparison with the behaviour of a large data base system.

It is clear that the objective to obtain representative models and validated workload characterizations has not yet been convincingly met. The reasons for this are connected with the current state of the art of data base research in general, which will be summarized in the next section. However, it is also clear that research on modeling and analysis of data base systems as described in this section has made significant progress, and that its continuation is extremely important from a practical point of view.

7. SUMMARY AND CONCLUSIONS

Before we try to summarize the research activities of the past, at least two major factors have to be considered:

Data base systems are in their objectives of a complexity that is in our opinion by far greater than the complexity of programming language implementations or operating systems. Consider alone the goals of data integrity and data independence. In conventional systems, the integrity remains the responsibility of the user, in a data base system the system has to take over a large part of this responsibility. In a conventional system, a user's program may be device independent due to implementation of equivalent storage structures on different devices. The goal in a data base system requires that the user's program is not only independent of different storage on the same (or another) device, but also that the system takes advantage of current structures during

access where restructuring is under control of the user in his data base administrator role.

The area of data base systems research is new. Major activities started only a few years ago. Understanding the real problems takes a large amount of time; demonstrating the viability of a solution requires expensive prototype implementation efforts. Before such large implementations are performed, the risk of failure has to be reduced by prior assessment. This justifies that a fair amount of research was spent in clarification. For example, it is sometimes held against the researchers that they are engaged in a "religious war" around data models. The question, which data model is taken, is certainly important and of a similar nature as the question, which programming language should be supported. However, data base researchers are now discussing the problem of different models with a changed attitude: it is not so much the question of selecting between two models but more the question of how one model can be represented on top of the other.

A number of promising activities have been started and will continue, which design and evaluate the man-machine interface for the interactive problem solver. In another branch of research, investigations into the system aspects have reached a level that prototype efforts are justified and now under way.

Data translation, driven by data description and mapping languages, continues to be investigated with increased power of the languages. With respect to storage structures there is already more available than can be intelligently handled by data base management systems. Research probably has to put more emphasis on how these structures can be efficiently utilized.

Modeling systems in a way that significant help results for the data base administrator is in its beginning. It will take some time, before the research which has already been conducted and has to be continued is combined into a useful set of tools for the system designer or administrator.

Comparing the obtained results with industry activities we may see first a difference of emphasis. Systems like IMS are primarily designed for and employed by parametric users while research systems are primarily designed for the interactive problem solver. With the current state of art, it is likely that research changes priority some-

what in favour of the parametric user. The modeling and analysis work described in section 6 is already now primarily oriented towards running, productive systems.

Conclusions

With the wealth of research existing, it becomes meaningful to ask: what are among all these results the major achievements? Are there any trends recognizable with respect to a change of research direction? What are currently the major problems? While we are trying to answer these questions, we are well aware that the reader may wholeheartedly disagree.

Major results

1. Model Development for Data Independence

One of the primary achievements of past research is the agreement on a type of data base system structure, which is shown in fig. 1. In particular, this means that we have to deal with at least three levels of information (conceptual, internal, external) that users assume different roles (parametric, problem solving, application programming, and data base administrating) and finally that the user in his data base administrator function has control over storage structures to tune the performance of his installation.

2. Multiple Records at a Time Logic

Due to the orientation of research to the interactive problem solver, high level multiple record at a time logic has been developed exceeding in power and flexibility the features offered in many commercially available systems. In particular the notions of views and predicate locks are of similar importance to the parametric use of data bases as to the problem solving use.

3. Storage Structures

Storage structures like the B-trees or to say it more generally "what can be found in Knuth vol. 3, chapter 6" or other textbooks represent important results and are basic to future research activities.

Recognizable Trends

1. Data Model Coexistence

After years of controversies, it is increasingly realized that different models have their justification even within the same system. The coexistence of different models in one system is called the superimposition problem and likely to find more attention in the future.

2. Integration of the DBMS into the OS

Past research has made apparent that many of the problems in the area of scheduling, resource management and recovery, i.e. classic operating system functions, cannot be solved outside the data base management system. Increased experience in this area has already lead to systems, which in much respect contain operating system functions. It can be expected that further research makes the need of integrated solutions to operating, time sharing and data base management systems even more apparent.

3. Data Dictionary/Directory

With the current merge of operating system and data base management system arises a large number of places where descriptive information about data and programs is stored in the system. A trend is recognizable to combine these different types of descriptions into a central data dictionary, thereby ensuring more consistency among the descriptive data and generally offering a simpler interface to the user for maintaining the descriptive information.

Major problems

1. Performance

Performance in the sense of throughput and transaction rate constitutes currently the major problem. It is generally felt that current systems do not offer the level of achievable performance, though this can only be proved by better performing alternatives. In particular, that CPU time may constitute a bottleneck has not been recognized in the past and research is necessary in this area.

2. Integrity, Data Independence, Recovery

It is necessary to provide more possiblities of specifying system enforceable integrity rules and data representations, which can be handled by the system with efficiency. The emphasis here is on more functions and efficiency (to provide these functions ignoring efficiency is trivial) so that the users installation as a whole has benefit. Similarly techniques which allow rapid recovery from failures are extremely desirable and lacking.

3. Concurrency

The problems of concurrency (deadlock prevention, scheduling) again in connection with efficiency, have not been solved in a satisfactory way. These problems increase in multiprocessing systems and with data bases, which are distributed on a network of computers.

4. Design Tools

In todays systems, and even more so in future systems, the user has to make a number of decisions like: how to model the conceptual information, or how to select hardware and physical representation. With the current state of the art, he is not given much information, which helps in making these decisions. Some of the research reported in section 6 is certainly relevant for the development of such tools.

5. Data Reorganization

In a dynamic system with addition, deletion and update of stored information it is inevitable to physically reorganize the data from time to time. The reason is a type of physical disorder like storage fragmentation, which does not affect the logical order, but degrades performance and storage utilization. To reestablish physical order, it is, in general, necessary to dump and reload significant parts of the data base as a whole, which is therfore not available during this process for normal use. For large data bases, which are used around the clock, the interruption may become too long to be tolerable. (The time range is from hours to weeks for a reorganization). A solution to the reorganization problem is necessary which avoids interruption.

Acknowledgement

The authors are grateful to their collegues at the IBM Heidelberg Scientific Center, to members of the IBM Research staff at Yorktown Heights and San Jose and to representatives from Universities in Europe and North America for many helpful discussions. Specifically they are grateful to E. F. Codd and M. E. Senko for a critical review of a preversion of this report.

8. BIBLIOGRAPHY

The subsequent list of references contains the basis for this status report. It is hoped that it is also of value as a reference to recent research results. Where present, the annotations are intended to help the reader in selecting literature. They should not be considered as critical reviews, which can be found elsewhere. Subsection I contains alphebetically ordered lists of cross references in numbers referring to entries in subsection II, which is a partially annotated list of references, ordered according to first author.

8.1. Cross References

Privacy

Recovery

Resource Allocation and Scheduling

Search Algorithms

Storage Structures

Surveys and Textbooks

8.2. References

1. Abrial, J.R. Data Semantics. Data Base Management, Proc. of IFIP Work. Conf., Cargese, Corsica April 1974. North Holland, Amsterdam 1974.
The paper is mathematical and philosophical with a scope exceeding the data base management area. It describes and advocates a data model with binary relations between entities.

2. ANSI/X3/SPARC. Interim Report: Study Committee on Data Base Management Systems. ACM SIGMOD Newsletter, 1975.

3. Ash, W. L., and Sibley. TRAMP: An Interpretive Associative Processor with Deductive Capabilities. 1968 ACM Natl. Conference, 144 - 156 (1968).
TRAMP is the implementation of a binary relational model in a question answering system. It accepts definitions of relations in terms of other relations (e.g. the grandfather as a function of father and mother) which leads to deductive capabilities.

4. Ashany, R. Concepts of Data Manipulation. The Connection Matrix Method. IBM System Development Division, Poughkeepsie, T.R. 00.2200 June 1971
Information is represented as a binary matrix, where the rows represent entities, the columns represent attributes, and a 1 in a position indicates that the attribute is true with respect to the entity, otherwise false. Sparse matrix techniques have to be applied to reduce storage requirements.

5. Astrahan, M. M., and Gosh, S. P. A Search Path Selection Algorithm for the Data Independent Accessing Model (DIAM). Proc. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
A heuristic algorithm is described which constructs a DIAM access path to a given query in RIL (Fehder).

6. Astrahan, M. M., and Chamberlin, D. D. Implementation of a Structured English Query Language. CACM 18, 580 - 588 (1975).
Describes essentially the SEQUEL interpreter and the reduction algorithm employed by it to make use of secondary indexes for "minimization" of data accessing operations.

7. Bachmann, C. W. The Programmer as Navigator. CACM 16, 653 - 658

(1973).
C. W. Bachmann's famous 1973 ACM Turing Award Lecture.

8. Bachmann, C. W. Trends in Data Base Management. AFIPS NCC 1975 Proc. vol. 44, 569 - 576 (1975).
Trends are:
1. The evolution of a tripartite data description (conceptual, internal, external) as used by ANSI/X3/SPARC.
2. The current debate data structured model (graph, network) vs relational model contributes to the understanding of the nature of data. 3. The introduction of new hardware to support data base algorithms.

9. Bayer, R., and McCreight, E. Organization and Maintenance of Large Ordered Indexes. Acta Informatica 1, 173 - 189 (1972).
The described hierarchical index organization (B-tree) has become a standard storage structure. Logarithmic search and efficient insert, delete are characteristics of the method.

10. Bayer, R. Symmetric Binary B-trees, Data Structure and Maintenance Algorithms. Acta Informatica 1, 290 - 306 (1972).
Symmetric Binary B-trees are a modification of the storage structure described by Bayer and McCreight.

11. Bayer, R. Storage Characteristics and Methods for Searching and Addressing. Information Processing 74, 440 - 444, North Holland, Amsterdam, 1974.
The paper contains a discussion of hashing and B-trees in random access, pseudo random access (i.e. indexed sequential) and virtual memories.

12. Bennet, B. T., and Kruskal, V. J. Stack Processing for Data Base Systems. To appear in IBM J.of Res. and Dev. (1975).
Traditional stack processing algorithms are inefficient for large average stack distances as they appear in the case of a large number of distinct pages. The authors describe a new algorithm to handle this situation with drastically improved efficiency.

13. Bergen, M., Erbe, R., Pistor, P., Schauer, U., and Walch, G. An Environment for the Interactive Evaluation of Scientific Data and its Application in Computer Aided Design. Proc. Workshop on

data bases for interactive design (W. M. Cleemput and J. G. Linders, editors), Waterloo, Canada, September 15-16, 1975, available from ACM. See also Schauer /149/.

14. Biller, H., and Neuhold, E. J. Formal View on Schema-Subschema Correspondence. Information Processing 74, Proc. of IFIP Congress, North Holland, Amsterdam, 1974.

15. Bjork, L. A. Recovery Scenario for a DB/DC System. 1973 ACM National Conf. Proc., 142-146 (1973).
This paper is the second of two papers describing a recovery concept in a data base system. See C. T. Davies for the first of the two papers.

16. Bjorner, D., Codd, E. F., Decker, K. L., Traiger, I. L. The Gamma-Zero n-ary Relational Data Base Interface: Specifiacations of Objects and Operations. IBM Research Report RJ 1200, 1973.
A detailed description of a low level query language accessing a relational data base.

17. Bobrow, R. J. An Experimental Data Management System. In Data Base Sytems (R. Rustin editor), Prentice-Hall, Englewood Cliffs, 1972.
The paper describes an experimental system implemented in LISP. It contains a brief but excellent discussion of the EDMS (hierarchy or network) approach vs. Codd's relational approach.

18. Boyce, R. F., Chamberlin, D. D., King, W. F., and Hammer, M. M. Specifying Queries as Relational Expressions: SQUARE. Data Base Management, Proc. of IFIP Work. Conf. Cargese, Corsica, April 1974, North Holland, Amsterdam, 1974.
SQUARE is a syntatically terse, set oriented, high level query language based on the so-called "concept of mapping". See also Chamberlin/Boyce for "SEQUEL".

19. Bracchi, G., Fedeli, A., and Paolini, P. A Relational Data Base Management System. Laboratorio di Calcolatori, Instituto di Elettrotecnica, Politechnica di Milano, Internal Report No. 72-5, 1972.
MORIS is a Codd relational system with a calculus oriented manipulation language. The users view (external schema) may include hierarchical structures (i.e. unnormalized data).

20. Bracchi, G., Fedeli, A., and Paolini, P. A Multilevel Relational Model for Data Base Management Systems. In Data Base Management, Proc. of IFIP Work. Conf., Cargese, Corsica, April 1974, North Holland, Amsterdam, 1974.
Advocates the binary relational model (graph model) for the conceptual schema and many models for the external schema (hierarchical, Codd relational, etc.) as well as internal schema.

21. Buzen, J. P., and Chen, P. P.-S. Optimal Load Balancing in Memory Hierarchies. Information Processing 74, 271-275. North Holland, Amsterdam, 1974.
A queuing model for the access to data sets in a memory hierarchy is used to analyze the allocation of data sets. The paper offers a generalization of Chen's results.

22. Cardenas, A. F. Evaluation and Selection of File Organization - a Model and System. CACM 16, 540 -548, 1973.
A program is described, which may be used to estimate total storage costs and average access time given the data organization and device related specifications.

23. Cardenas, A. F. Analysis and Performance of Inverted Data Base Structures. CACM 18, 253 - 263, 1975.
See also King, Farley/Stewart, Schkolnick and Yue/Wong for recent treatments of this subject.

24. Cardenas, A. F., and Sagamang, J. P. Modeling and Analysis of Data Base Organization: The Doubly Chained Tree Structure. Inform. Systems 1, 57 -67, 1975.

25. Carlson, E. P., Bennet, J. L., Giddings, G. M., and Mantey, P. E. The Design and Evaluation of an Interactive Analysis and Display System. Information Processing 74, 1055 - 1061, North Holland, Amsterdam, 1974.
GADS is an interactive graphics system for data related to geographic locations and intended as a tool to be used by non-programmers. It provides a data extraction technique for accessing data stored in a variety of files. The paper discusses experience gained with GADS and the requirements, which must be met by a system of this kind.

26. Casey, R. G. Allocations of Copies of a File in an Information Network. AFIPS SJCC 1972 Proc., vol. 40, 617 - 225, 1972.
The author gives an exact and a heuristic solution to the problem of allocating data sets within a network of computers, given the costs of storing at and transmission between nodes.

27. Casey, R. G. Design of Tree Networks for Distributed Data. AFIPS NCC 1973 Proc. vol. 42, 251 - 257, 1973.

28. Chamberlin, D. D., and Boyce, R. F. SEQUEL - a Structured English Query Language. ACM SIGFIDET Workshop 1974, ACM, New York, 1974.
SEQUEL is a language with semantics very similar to those of SQUARE, however, with a syntax closer to natural English. See Boyce/ Chamberlin for SQUARE.

29. Chamberlin, D. D., Boyce, R. F., and Traiger, I. L. A Deadlock Free Scheme for Resource Locking in a Data Base System. Information Processing 74, 340 - 343. North Holland, Amsterdam, 1974.
The authors propose to use deadlock-detection and backout of processes in case of deadlocks. Their algorithm avoids indefinite delays of a process.

30. Chamberlin, D. D., Gray, J. N., Traiger, I. L. Views, Authorization and Locking in a Relational Data Base System. 1975 AFIPS NCC Proc. vol. 44, 425 - 430, 1975.
A view is a virtual relation derived form other relations via the query language SEQUEL. The problem of updating views is discussed. Views can be used for authorization. Locks temporarily restrict the access to a view for the exclusive use of one user.

31. Chandra, A. K., and Wong, C. K. Worst Case Analysis of a Placement algorithm related to Storage Allocation. To appear in SIAM Journal on Computing.
The authors specify a heuristic algorithm to allocate data sets to disk drives such that the probability of simultaneous access of one disk drive is minimized. The worst case performance of the algorithm is analyzed. See also Easton/Wong.

32. Chang, S.K. Data Base Decomposition in a Hierarchic Computer System. ACM SIGMOD 1975 Int. Conf. on Mgmt. of Data, San Jose,

1975.
The author has extended Casey's results by allowing a non-linear cost function.

33. Chen, P. S. Optimal File Allocation in Multilevel Storage System. 1973 AFIPS NCC Proc. vol. 42, 277 - 282, 1973.
A treatment of the hierarchy allocation problem taking queuing effects into considerations. See also Buzen/Chen.

34. CODASYL Development Committee. Language Structure Group. An Information Algebra. CACM 5, 190 - 204, 1962.
An "oldtimer" and source for many ideas. Contains, for example, the definition of an entity or the idea that files may be interpreted as sets of n-tuples on which then joins, union and intersection can be performed.

35. CODASYL Programming Language Committee. DBTG-Report. 1971. Available from ACM.
The original DBTG proposal.

36. CODASYL Programming Language Committee. DBLTG proposal, February 1973.
Contains the COBOL data manipulation and suvschema data definition language. The languages are essentially those of ref. 35.

37. CODASYL Data description Language Committee. Data Description Language. Journal of Development, June 1973.
Essentially the same data definition language as in 35.

38. CODASYL Systems Committee. Feature Analysis of Generalized Data Base Management Systems. Technical Report, May 1971. Available from ACM.
Primarily compares commercially available systems, contains also a network data model.

39. Codd, E. F. A Relational Model of Data for Large Shared Data Banks. CACM 13, 377 - 387, 1970.
The paper in which Codd introduced the (Codd) relational model of data.

40. Codd, E. F. A Data Base Sublanguage Founded on the Relational

Calculus. 1971. ACM SIGFIDET Workshop, ACM, New York, 1971.

41. Codd, E. F. Further Normalization of the Data Base Relational Model, and Relational Completeness of Data Base Sublanguages. In Data Base Systems (R. Rustin editor). Prentice-Hall, Englewood Cliffs, 1971.

42. Codd, E. F. Seven Steps to Rendezvous with the Casual User. In Data Base Management, Proc. of IFIP Work. Conf. Cargese, Corsica, April 1974, North Holland, Amsterdam, 1974.
The description of seven steps to a proposed natural language question answering system. The steps are: simple data model, high level internal logic, clarification dialogue, query restatement, declarative query, multiple choice interrogation and a definition capability.

43. Codd, E. F. Recent Investigations in Relational Data Base Systems. Information Processing 74, 1017 - 1021, North Holland, Amsterdam, 1974.
A brief survey of Codd's relational model including a discussion of normalization and data sublanguage types. The author lists concurrency, performance, superimposition and storage access theory among the topics needing investigation.

44. Conway, R. W., Maxwell, W. L., and Morgan, H. L. On the Implementation of Security Measures in Information Systems. CACM 15, 211 - 220, 1972.
The main idea in this paper is to perform checking of security only "once at compile time", an approach which is conscious of the CPU as a resource. The paper contains also a discussion of security systems implemented by 1972.

45. Conway, R. W., Maxwell, W. L., and Morgan, H. L. A Technique for File Surveillance. Information Processing 74, 988 - 992. North Holland, Amsterdam, 1974.
A technique implemented by the authors in their system ASAP is described. Each file has associated with it a set of function declarations, which are compiled into a file surveillance program. All accesses to the file have to pass through the surveillance program, which can then be used to perform certain automatic functions.

46. Dana, C., and Presser, L. An Inforamtion Structure for Data Base and Device Independent Report Generation. AFIPS FJCC 1972 Proc. vol. 41, 1111 - 1116, 1972.
The paper describes high level elements for the generation and manipulation of reports.

47. Date, C. J., and Hopewell, P. Storage Structure and Physical Data Independence. 1971 ACM SIGFIDET Workshop, ACM, New York, 1971.

48. Date, C. J., and Hopewell, P. File Definition and Logical Data Independence. 1971 ACM SIGFIDET Workshop, ACM, New York, 1971.

49. Date, C. J. An Introduction to Data Base Systems. Addison-Wesley, Reading, Massachusetts, 1975.
Similar to Wedekind's book, one of the first attempts of a comprehensive introduction to data base systems. Many annotated references.

50. Davies, C. T. Recovery Semantics for a DB/DC System. 1973 ACM Natl. Conf. Proc., 136 - 141, 1973.
Together with Bjork's paper an easy to understand introduction to a recovery concept.

51. Dearnley, P. A. Operation of a Model Self Organizing Data Management System. Comp. Journal 17, 205 - 210, 1974.
Among others the system observes patterns of usage and restructures the data accordingly. Simulation results are reported.

52. Delobel, C., and Casey, R. G. Decomposition of a Data Base and the Theory of Boolean Switching Functions. IBM J. Res. Develop. 17, 374 - 386, 1973.
Deals with the problem of decomposition of a flat file with (enormous) redundancy into a set of flat files having the minimal cover property, i.e. allowing to derive the same information as the original file without allowing further decomposition.

53. Di Paola, R. A. The Solvability of the Decision Problem for Classes of Proper Formulas and Related Results. Rand Corp., Santa Monica, Calif. Technical Report R-803-PR, August 1971.
The paper deals with the solvability of the decision problem of

a class of questions to be processed by Rands Relational Data File. See Levien/Maron.

54. D'Imperio, M. E. Data Structures and their Representation in Storage. Annual Review in Automatic Programming, vol. 5, Pergamon Press, 1969.

55. Dittmann, E. L. Klassifizierung von Datenunabhaengigkeit fuer den System-Entwurf. Technische Hochschule Darmstadt. Berichte der Informatik- Forschungsgruppen DV75-1

56. Doerrscheidt, A. Das Konzept des Objektbeschreibungsbaumes als Grundstruktur eines graphenorientierten Datenbankmodells. Lecture notes in computer science 26, 532 - 541, Springer Verlag, Berlin, 1975.
Describes a typically graph oriented data model based on LISP ideas.

57. Durchholz, R., and Richter, G. Concepts for Data Base Management Systems. Data Base Management Proc. IFIP Work. Conf. Cargese, Corsica, April 1974. North Holland, Amsterdam, 1974.
Influenced by the data model of the "CODASYL Feature Analysis" the authors discuss a hierarchical data model and schema.

58. Earley, J. Towards an Understanding of Data Structures. CACM 14, 617 - 628, 1971.
Sketches some ideas related to a theory of data structures similar to the available theory of formal string languages.

59. Earley, J. Relational Level Data Structures for Programming Languages. Acta Informatica 2, 293 - 309, 1973.
A proposal for the incorporation of relational level data structures into ALGOL like languages.

60. Easton, M. C., and Wong, C. K. The Effect of Capacity Constraints on the Minimal Cost of a Partition. JACM, 22, 441 - 449, 1975.
A new algorithm to the problem considered by Chandra/Wong is proposed, which accepts capacity constraints.

61. Easton, M. C. Model for Interactive Data Base Reference String. IBM Research Report RC 5050, Sept. 1974.

Describes a modification of the independent references model, which describes measured behaviour well. An advantage of the model is its analytical tractabilitiy under working set assumptions.

62. Edelberg, M. Data Base Contamination and Recovery. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
The paper describes an algorithm, which for a given error and a given set of data transfers (i.e. log) determines the error propagation into processes and data blocks. A recovery algorithm is also described, which restores blocks and reruns processes.

63. Ehrich, H. D. Grundlagen einer Theorie der Datenstrukturen. Acta Informatica 4, 201 -211, 1975.
A graph oriented data model and graph oriented schemata within the model are investigated from a more mathematical point of view.

64. Engles, R. W. A Tutorial on Data Base Organization. Annual Review in Automatic Programming vol. 7 part 1, Pergamon Press, 1972.

65. Eswaran, K. P., Gray, J. N., Lorie, R. A., and Traiger, I. L. On the Notions of Consistency and Predicate Locks in a Data Base System. IBM Research Report RJ 1487, December 1974.
The paper defines the notion of transaction, consistency within concurrency, and predicate locks and their consequences. A language for predicate specification is proposed, and an algorithm is presented which determines whether two such predicates overlap.

66. Eswaran, K. P., and Chamberlin, D. D. Functional Specifications of a Subsystem for Data Base Integrity. IBM Research Report RJ 1601, 1975.
Contains a classification of consistency rules. Consistency rules are interpreted as routines to be invoked after changes of the data base.

67. Everest, G. C. Concurrent Update Control and Data Base Integrity. Data Base Management, 241 - 270, Proc. IFIP Work. Conf. Cargese, Corsica, April 1974. North Holland, Amsterdam, 1974.
Preclaiming of resources to prevent deadlocks is advocated by

the author.

68. Falkenberg, E., Meyer, B., and Schneider, J. Resultatspezifizierende Handhabung von Datensystemen. Lecture Notes in computer science 1, Springer Verlag, Heidelberg, 1973.
Informal discussion of the "Gegenstandsmodell", a data model, and of a high level manipulation language for it.

69. Falkenberg, E. Time-Handling in Data Base Management Systems. University of Stuttgart, Institut fuer Informatik, Internal CIS-Report 07/74, 1974.
Adds the dimension of time to (for example: A is employee of B from T1 to T2) stored relations and extends a data manipulation language to cope with the time dimension.

70. Falkenberg, E. Strukturierung und Darstellung von Information an der Schnittstelle zwischen Datenbankbenutzer und Datenbank-Management-System. Thesis, University of Stuttgart, 1975.
A detailed description of a data model and a data manipulation language where both are closely related to concepts in natural language. The model is graphoriented though it allows for n-ary relations which can be (and graphically are) interpreted as joins of binary relations.

71. Farley, J. H. G., and Stewart, S. A. Query Execution and Index Selection for Relational Data Bases. Technical Report CSRG-53, University of Toronto, March 1975.
See also Cardenas for recent investigations into this subject.

72. Fehder, P. L. The Representation Independent Language. IBM Research Reports RJ 1121 (1972) and RJ 1251 (1973).
The papers describe RIL, the data manipulation language to the DIAM system.

73. Fehder, P. L. The Hierarchic Query Language (HQL) part 1. IBM Research Report RJ 1307, Nov. 1973.
Describes a query language to operate on IMS like hierarchic data.

74. Feldman, J. A., and Rovner, P. P. An ALGOL based Associative Language. CACM 12, 439 - 449, 1969.
The high level, ALGOL like programming language LEAP is based on

binary associations, which are implemented using a hash coding technique.

75. Fernandez, E. B., Summers, R. C., and Coleman, C. P. An Authorization Model for a Shared Data Base. ACM SIGMOD 1975 Intl. Conf. on Mgmt. of Data, San Jose, 1975.
Authorization is governed by predicates over applications and data base contents and enforced primarily at compile time.

76. Fiedler, H. Datenschutz und Gesellschaft. Lecture Notes in Computer Science, vol. 26, 1975
A survey of the discussions on privacy.

77. Finkel, R. A., and Bentley, J. L. Quad-trees: a Data Structure for Retrieval on Composite Keys. Acta Informatica 4, 1 - 9, 1974.
A generalization of binary trees for the search on composite keys.

78. Florentin, J. J. Consistency Auditing of Data Bases. Comp. Journal 17, 52 - 58, 1974.
Consistency rules are predicate calculus expressions over the data base contents. Problems of their implementation are discussed.

79. Frank, R. L., and Sibley, E. H. The DBTG Report: An Illustrative Example. University of Michigan, ISDOS - working paper - 7.
Shows in detail the steps, which have to be made to get a COBOL application program running in the DBTG approach.

80. Frank, R. L., and Yamaguchi, K. A Method for a Generalized Data Access Method. 1974 AFIPS NCC Proc. vol. 43, 45 - 52, 1974.
Describes ideas and a keyword oriented language to tailor access methods to the users specifications.

81. Fraser, A. G. Integrity of a Mass Storage Filing System. Comp. Journal 12, 1 - 5, 1969.
Describes the recovery in MULTICS.

82. Frasson, C. A System to Increase Data Independence in an Hierarchical Structure. Lecture Notes in Computer Science, vol. 34 (GI 1975), Springer Verlag, Heidelberg, 1975.

Describes how IMS structures can be accessed independent of their position in the hierarchy.

83. Genton, A. Recovery Procedures for direct Access Commercial Systems. Comp. Journ. 13, 123 - 126, 1970.
Describes elementary checkpointing and journaling techniques.

84. Ghosh, S. P., and Senko, M. E. String Path Search Procedures for Data Base Systems. IBM J. Res. Dev. 18, 408 - 422, 1974.
Within DIAM the reduction of queries to access paths in a network is considered. An algorithm is given, which is claimed to yield an access path of minimum "path cardinality".

85. Ghosh, S. P., and Lum, V. Y. Analysis of Collision when Hashing by Division. Inform. System 1, 15 - 22, 1975.
It is analytically shown that "hashing by division" is in general best.

86. Ghosh, S. P., and Tuel, W. G. A Design of an Experiment to Model Data Base System Perfromance. IBM Research Report RJ 1482, Dec. 1974.
The authors construct a linearized performance model and evaluate the model by comparison with measurements in an IMS system.

87. Goldstein, R. C., and Strnad, A. J. The MacAims Data Management System. 1970 ACM SIGFIDET Workshop, ACM, New York, 1970.
MacAims is an early relational system.

88. Gorenstein, S., and Galati, G. Data Base Reorganization for a Storage Hierarchy. IBM Research Report RC 5063, Oct. 1974.
The problem considered is that of clustering records into blocks (i. e. units of transfer) in a way as to minimize the number of transfers necessary.

89. Greenfeld, N. R. Quantification in a Relational Data System. 1974 AFIPS NCC Proc. vol. 43, 71 - 75, 1974.
Discusses optimization techniques for a relational system like LEAP (see Feldman/Rovner).

90. Haerder, T. Die Implementierung von Zugriffspfaden durch Bitlisten. Technische Hochschule Darmstadt, Berichte der Informatik-Forschungsgruppen DV74-2.

The author proposes bit lists as an index organization and investigates when bit lists are superior to conventional methods of indexing.

91. Haerder, T. Zugriffszeitverhalten bei der Auswahl von Saetzen aus einer Datenbank. Technische Hochschule Darmstadt, Berichte der Informatik-Forschungsgruppen DV74-3.
Analysis of access with the help of simulation. Includes a comparison of storage structures for indexes.

92. Hall, P. A. V. Common Subexpression Identification in General Algebraic Systems. IBM UK Report UKSC0060, Nov. 1974.

93. Held, G. D., Stonebraker, M. R., and Wong, E. INGRES - a Relational Data Base System. 1975 AFIPS NCC Proc. vol. 44, 409 - 416, 1975.
INGRES is a relational data management system with calculus based QUEL as its high level query language. An interesting plan of the authors is to incorporate access control and integrity assurance via query modification at preprocessing time.

94. Hoffmann, L. J. (editor). Security and Privacy in Computer Systems. Melville Publishing Company, Los Angeles, 1973.

95. Housel, B. C., Smith, D. P., Shu, N. C., and Lum, V. Y. DEFINE: A Nonprocedural Data Description Language for Defining Information Easily. Proc. of ACM Pacific, San Francisco, April 1975, ACM, New York, 1975.
Describes a language DEFINE to map graph structures to a linear form, which is then referenced by (and processed according to) a translation specification, written in the language CONVERT. See Shu et al.

96. Inglis, J. Iverted Indexes and Multilist Structures. Comp. Journ. 17, 59 - 63, 1974.
Discusses how to use multilist structures in order to maintain inverted files.

97. Karp, R. M., McKellar, A. C., and Wong, C. K. Near-optimal solutions to a 2-dimensional placement problem. IBM Research Report RC 4740, also to appear in SIAM Journal of Computing.
The problem considered is the placement of records in a 2-dimen-

sional storage array, so that the expected distance between two consecutive references is minimized.

98. King, W. F. On the Selection of Indices for a File. IBM Research Report RJ 1341, January 1974.
See also Cardenas for recent research in this area.

99. Knuth, D. E. The Art of Computer Programming, vol. 1: Fundamental Algorithms. Addison-Wesley, Reading, Massachusetts, 1968.

100. Knuth, D. E. The Art of Computer Programming, vol. 3: Sorting and Searching. Addison-Wesley, Reading, Massachusetts, 1973.

101. Kogon, R., Lattermann, D., Lehmann, H., Ott, N., and Zoeppritz, M. User Specialty Languages: General Information. IBM Germany, Scientific Center Heidelberg, Technical Report 75.08.007, 1975.
An interactive system is introduced designed to a data manipulation language,which is very close to natural language.

102. Kraegeloh, K. P., and Lockemann, P. C. Retrieval in a set-theoretically Strucutred Data Base: Concepts and Practical Considerations, Proc. of International Computing Symposium 1973, 531 - 539. North Holland, Amsterdam, 1973.
The described system has a natural language like query language, which is translated into a "set theoretic" intermediate language suitable for interpretation.

103. Lavenberg, S. S., and Shedler, G. S. A Queuing Model of the DL/I Component of IMS. IBM Research Report RJ 1561, 1975.
A simplified, analytically tractable queuing model of the processes during data base access.

104. Lefkovitz, D. File Structures for On-Line Systems. Spartan Books, 1969.

105. Levien, R. E., and Maron, M. E. A Computer System for Inference Execution and Data Retrieval. CACM 10, 715 - 721, 1967.
Introduces the Relational Data File, a system based on binary relations (see also Di Paola).

106. Levitt, G., Stewart, D. H., and Yormark, B. A Prototype System for Interactive Data Analysis. 1974 AFIPS NCC Proc. vol. 43, 63

- 69, 1974.
Describes an implemented system for analysis of measurement data relying on standard analytic procedures. It makes heavy use of graphics and statistical methods.

107. Lewis, P. A. W., and Shedler, G. S. Statistical Analysis of Transaction Processing in a Data Base System. IBM Research Report RJ 1629, August 1975.
Describes the modeling of a transaction stream as a Poisson process with a time varying rate.

108. Liu, S., and Heller, J. A Record Oriented, Grammar Driven Data Translation Model. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
Grammars may be taken as mappings of a string to a tree. Two grammars mapping different strings to equivalent trees are used as a string to string mapping specification.

109. Lockemann, P. C., and Knutsen, W. D. A Multiprogramming Environment for Online Data Acquisition and Analysis. CACM 10, 758 - 764, 1967.
An earlier approach to the problem of measurement data. Prefabricated programs may be assembled communicating via data sets and parameters.

110. Lorie, R. A., and Symonds, A. J. A Schema for Describing a Relational Data Base. Proc. 1970 ACM SIGFIDET Workshop, ACM, New York, 1970.
Describes RAM - a data base management system based on binary relations (in some sense like LEAP of Feldman/Rovner).

111. Lorie, R. A. XRM - an Extended (n-ary) Relational Memory. IBM Scientific Center Report G 320 - 2096, Cambridge, Massachusetts, January 1974.
XRM implements homogeneous flat files on top of RAM (see Lorie/Symonds).

112. Lum, V. Y. Multi-attribute Retrieval with Combined Indexes. CACM 13, 660 - 665, 1970.

113. Lum, V. Y., Yuen, P. S. T., and Dodd, M. Key to Address Transform Techniques, a Fundamental Performance Study on Large Exist-

ing Formatted Files. CACM 14, vol. 4, 1971.
Contains a survey and evaluations of hashing techniques as applied to large data sets.

114. Lum, V. Y., and Ling, H. An Optimization Problem on the Selection of Secondary Keys. Proc. 1971 ACM Natl. Conf., vol. 26, 349 - 356, 1971.
One of the earlier investigations into the problem considered by Cardenas and others.

115. Lum, V. Y. General Performance Analysis of Key-To-Address Transformation Methods Using an Abstract File Concept. CACM 16, 603 - 612, 1973.

116. Lum, V. Y., Senko, M. E., Wang, C. P., and Ling, H. A Cost Oriented Algorithm for Data Set Allocation in Storage Hierarchies. CACM 18, 318 - 322, 1975.
A cost function combining the cost of storage, CPU, channel etc. is defined and an algorithm for data set allocation is outlined, which minimizes this cost.

117. Maruyama, K., and Smith, S. E. Analysis of Design Alternatives for Virtual Memory Indexes. IBM Research Report RC 5087, Oct. 1974.
A number of implementation alternatives for indexes organized as B-trees are analyzed resulting into formulas, which are numerically evaluated.

118. Maurer,W. D., and Lewis, T. G. Hash Table Methods. ACM Computing Surveys 7, 5 - 19, 1975.

119. McDonald, N., and Stonebraker, M. CUPID - the Friendly Query Language. ACM Pacific Conference, San Francisco, April 1975, ACM, New York, 1975.
CUPID is a grahic, data flow diagram-like language to the INGRES system. See also Held.

120. McGee, W. C. Generalized File Processing. Annual Review in Automatic Programming vol. 5, Pergamon Press, 1969.

121. McGee, W. C. File Structures for Generalized Data Management. Information Processing 68, 1233 - 1239, North Holland, Amster-

dam, 1968.
Introduces graphs as conceptual models for stored information.

122. McGee, W. C. A Contribution to the Study of Data Equivalence. Data Base Management. Proc. IFIP Work. Conf. Cargese, Corsica, April 1974, North Holland, Amsterdam, 1974.
The author presents a number of equivalent organizations in the class of homogeneous flat file (CRM) organizations and of data description language (DBTG) organizations.

123. McGee, W. C. File Level Operations on Network Data Structures. ACM SIGMOD 1975 Intl. Conference, Proc., ACM, New York, 1975.
The paper outlines requirements and a proposal for a data manipualtion language operating on network data structures.

124. Mealey, G. H. Another Look at Data. Proc. AFIPS 1967 FJCC 525 - 534, 1967.
One of the earlier papers proposing to view information as sets and relations between sets.

125. Mehl, J. W., and Wang, C. P. A Study of Order Transformations of Hierarchic Structures in IMS Data Bases. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
A proposal to increase the data independence supported by IMS with the help of compiled routines, which intercept the communication between application program and data management.

126. Merten, A. G., and Fry, J. P. A Data Description Language Approach to File Translation. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
Describes the idea and design behind the University of Michigan data translation project.

127. Merten, A. G., and Severance, D. G. Performance Evaluation File of Organizations through Modeling. Proc. ACM 1972 Natl. Conf., ACM, New York, 1972

128. Meyer, B., and Schneider, H. J. Predicate Logic and Data Base Technology. Course Notes, University of Berlin, available from the authors.
Reviews predicate logic and its use as a model for man-machine interface like in Codd's work and in natural language question-

answering systems.

129. Minsky, N. On Interaction with Data Bases. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
The author discusses concepts, integrity rules, user views etc. He proposes a constructive approach to integrity by defining "consistent operators" to be used as primitives for more complex operations.

130. Mullin, J. K. An Improved Index Sequential Access Method using Hashed Overflow. CACM 15, 301 - 307, 1972.

131. Mylopoulos, J., Schuster, S., and Tsichritzis, D. A Multilevel Relational System. 1975 AFIPS NCC Proc. vol. 44, 403 - 408, 1975.
The mechanism used in the development of the prototype system ZETA/TORUS are described. ZETA is a relational data management system with a definition capability to define a high level query language on top of lower level primitives. TORUS is bulit on ZETA as an "intelligent" natural language interface.

132. Nakamura, F., Yoshida, I., and Kondo, H. A Simulation Model for Data Base System Performance Evaluation. 1975 AFIPS NCC Proc. vol. 44, 459 - 463, 1975.
Description of experiments simulating the processes within a data base management system in a conventional simulation package.

133. Navathe, S. B., and Merten, A. G. Investigation into the Application of the Relational Model to Data Translation. ACM SIGMOD 1975 Intl. Conf. Proc., 123 - 138.
The paper concludes that Codd's relational model "... poses serious problems when used in the context of data translation as a vehicle for more powerful restructuring".

134. Neuhold, E. J. Data Mapping: A Formal Hierarchical and Relational View. University of Karlsruhe, Forschungsberichte, Bericht 10, February 1973.
The paper compares hierarchical and relational data models in formal notation. In particular, it makes clear that the relational model is a special case of the hierarchical model.

135. Nievergelt, J. Binary Search Trees and File Organization. ACM Computing Surveys 6, 3, 1974.

136. Notley, M. G. The Peterlee IS/I System. IBM UK, Peterlee, Report UK-SC 0018.
Describes IS/I, one of the earlier Codd relational implementations.

137. Olson, C. A. Random Access File Organization for Indirectly Accessed Records. Proc. of 1969 ACM Natl. Conf. ACM, New York, 1969.

138. Owens, P. J. Phase II - a Data Base Management Modeling System. Information Processing 71, 827 - 832, North Holland, Amsterdam, 1972.
Phase II is a modeling tool designed specifically for data management evaluation.

139. Palermo, F. P. A Quantitative Approach to the Selection of Secondary Indexes. IBM Research Report RJ 0730, July 1970.
One of the earlier papers on index selection. See Cardenas for recent results.

140. Palermo, F. P. A Data Base Search Problem. IBM Research Report RJ 1072, July 1972.
The paper contains one of the earlier optimizing reduction algorithms for queries in predicate calculus form.

141. Petrick, S. R. Semantic Interpretation in the REQUEST system. IBM Research Report RC 4457, July 1973.
REQUEST is an experimental, natural language question answering system.

142. Ramirez, J. A., Rin, N. A., and Prywes, N. S. Automatic Generation of Data Conversion Programs using a Data Description Language. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
Describes an implementation of a data definition language (due to D. P. Smith), which compiles data definitions into data translating programs.

143. Reisner, P., Boyce, R. F., and Chamberlin, D. P. Human Factors Evaluation of two Data Base Query Languages - SQUARE and SE-

QUEL. 1975 AFIPS NCC Proc. vol. 44, 447 - 452, 1975.
A psychological experiment with 64 subjects is described and analyzed. Only nonprogrammers show a slight but statistically significant dependency on the language, which differ primarily in syntax.

144. Reiter, A. Data Models for Secondary Storage Representation. University of Wisconsin, MRC Report no. 1554, May 1975.
The data models are designed with the objective to be used for the performance evaluation of different implementations.

145. Rodriguez-Rosell, J., and Hildebrand, D. A Framework for Evaluation of Data Base Systems. Proc. of ACM European Chapters International Computing Symposium 1975.
An implemented framework for the measurement and evaluation of sequences of events at different levels of a data base system is presented. The different levels involve commands issued in the application program at the hgih end, and disk address reference traces at the low end.

146. Rothnie, J. B., and Lozano, T. Attribute Based File Organization in a Paged Memory Environment. CACM 17, 63 - 69, 1974.
A combination of "multiple key hashing" and inverted file technique allowing for a reduction of the number of page faults for multi-key-retrieval.

147. Rothnie, J. B. Evaluating Inter-Entry Retrieval Expressions in a Relational Data Base Management System. 1975 AFIPS NCC Proc. vol. 44, 417 - 423, 1975.
The employed strategy attempts to utilize the information gained with every tuple-access for the purpose of optimization.

148. Sayani, H. H. Restart and Recovery in a Transaction Oriented Information Processing System. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
Restart and recovery policies are defined and discussed. The author puts emphasis on performance.

149. Schauer, U. Ein System zur interaktiven Bearbeitung umfangreicher Messdaten. IBM Germany, Informatik Symposium 1975, Bad Homburg. To appear as Lecture Notes in Computer Science, Springer Verlag, Heidelberg.

Introduces an interactive measurement data base system combining interactive computational facilities (APL), a relational data storage, a graphics oriented data manipulation language (like "query by example", see Zloof) with access to an open ended library of PL/I or FORTRAN subroutines. See also /13/.

150. Schkolnick, M. Secondary Index Optimization. ACM SIGMOD 1975 Intern. Conf. on Mgmt. of Data, San Jose, 1975.
See also Cardenas for similar research.

151. Schmid, H. A., and Swenson, J. R. On the Semantics of the Relational Data Model. ACM SIGMOD 1975 Intl. Conf. on Mgmt. of Data, San Jose, 1975.
The authors are concerned with the gap between the pure formalism of Codd's relational model and the modelled part of the real world. The authors employ a kind of graph model to fill the gap.

152. Schmutz, H. Parenthesis Regular Languages and Relations. IBM Germany, Heidelberg Scientific Center, Technical Report 74.10.004, Oct. 1974.
A special form of context-free grammars is used to describe the schema to a hierarchical data model. Pair grammars are used to describe the mapping between conceptual and internal or external view. The described system is a model for a theoretical treatment of important problems in data base systems.

153. Schneider, G. M., and Deasautels, E. J. Creation of a File Translation Language for Networks. Information Systems 1, 23 - 31, 1975.
The authors propose a language for data translation in a network such as the ARPA network.

154. Senko, M. E., Lum, V. Y., and Owens, P. J. A File Organization Evaluation Model (FOREM). Information Processing 68, 514 - 519, 1968. North Holland, Amsterdam, 1969.
FOREM is an evaluation and simulation tool specifically designed to evaluate data management systems.

155. Senko, M. E., Altman, E. B., Astrahan, M. M., and Fehder, P. L. Data Structures and Accessing in Data Base Systems. IBM Systems Journ. 12, 30 - 93, 1973.

This paper describes the thoughts and ideas behind the DIAM system, one of the earlier comprehensive approaches to data base research systems.

156. Senko, M. E. Information Systems: Records, Relations, Sets, Entities and Things. Inform. Systems 1, 3 - 13, 1975.

157. Senko, M. E. Data Description Language in the Context of a Multilevel Structured Description - DIAM II with FORAL. IBM Research Report RC 5073, Oct. 1973.

158. Senko, M. E. An Introduction to FORAL for Users. IBM Research Report RC 5263, 1975.

159. Senko, M. E. Specification of Stored Data Structures and Desired Output Results in DIAM II with FORAL. Proc. of the Int. Conference on Very Large Data Bases, Boston, 1975, available from ACM.
The last three references introduce DIAM II, a proposed system, which is based on binary associations and has FORAL as its query language.

160. Severance, D. G. Identifier Search Mechanism: A Survey and Generalized Model. ACM Computing Surveys 6, 3, 1974.

161. Severance, D. G. A Parametric Model of Alternative File Structures. Inform. Systems 1, 51 - 55, 1975.
A scheme is described, which maps a "two dimensional space of parameters" to a set of data organizations including well-known conventional organizations as special case.

162. Shneiderman, B. Optimum Data Base Reorganization Points. CACM 16, 362 - 365, 1973.

163. Shneiderman, B., and Scheuermann, P. Structured Data Structures. CACM 17, 566 - 577, 1974.
The paper describes an approach to deal with integrity in case of certain classes of data structures.

164. Shneiderman, B. A Model for Optimizing Indexed File Structures. IJCIS 3, 93 - 103, 1974.
The paper is concerned with the selection of index size at dif-

ferent levels to improve performance.

165. Shu, N. C., Housel, B. C., and Lum, V. Y. CONVERT a High Level Translation Definition Language for Data Conversion. CACM 18, 557 - 567, 1975.
A companion paper to Housel et al.

166. Sibley, E. H., and Taylor, R. W. A Data Definition and Mapping Language. CACM 16, 750 - 759, 1973.
The paper discusses goals of a data definition language and illustrates data definition and mapping by examples.

167. Sibley, E. H. On the Equivalence of Data Based Systems. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
The two philosophical directions, "relational" (Codd) and the "data structured" or "procedural" (DBTG) are compared. Also data translation with its connection to data restructuring and data independence is discussed.

168. Sibley, E. H., and Sayani, H. H. Data Element Dictionaries for the Information Systems Interface. NBS-Report, 1974.
A discussion of the need for and objectives of a Data Dictionary capability.

169. Smith, D. P. An Approach to Data Description and Conversion. PH. D. dissertation, University of Pennsylvania, 1971.
One of the earlier data definition and mapping languages. See also Ramirez.

170. Smith, S. E., and Mommens, J. H. Automatic Generation of Physical Data Base Structures. ACM SIGMOD 1975 Intl. Conf. San Jose, 1975.
A prototype design aid is described which generates from descriptive input IMS physical data structure definitions taking into account constraints and objective functions.

171. Stahl, F. A. A Homophonic Cipher for Computational Cryptograhy. AFIPS NCC Proc. vol. 42, 565 - 568, 1973.

172. Steel, T. B. Data Base Standardization - A Status Report. ACM SIGMOD 1975 Intl. Conf. on Mgmt. of Data, San Jose, 1975.

173. Steuert, J., and Goldman, J. The Relational Data Management System: A Perspective. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.
An introductory description of RDMS, a system being used at MIT and based on Codd's relational model.

174. Stonebraker, M. The Choice of Partial Inversions and Combined Indices. IJCIS 3, 167 - 188, 1974.
See also Cardenas for research on this topic.

175. Stonebraker, M. A Functional View of Data Independence. 1974 ACM SIGFIDET Workshop Proc., ACM, New York, 1974.
The paper first analyzes the problem with a promising formal approach, which unfortunately is not kept through up to the end. It describes the types of data independence to be provided in INGRES.

176. Stonebraker, M. Implementation of Integrity Constraints and Views by Query Modification. ACM SIGMOD 1975 Intl. Conf. Proc., San Jose, 1975.
Describes the INGRES approach to integrity in more detail. See also Held et al.

177. Su, S. Y. W., and Lam, H. A Semiautomatic Data Base Translation System for Achieving Data Sharing in a Network Environment. 1974 ACM SIGFIDET Workshop, ACM, New York, 1974.

178. Sundgren, B. Conceptual Foundation of the Infological Approach to Data Base. Data Base Management Proc. of IFIP Work. Conf. Cargese, Corsica, April 1974. North Holland, Amsterdam, 1974.
The infological approach is a kind of a conceptual data model philosophy. It may have a corresponding datalogical approach associated with it, which deals with internal data forms.

179. Taylor, R. W. Generalized Data Base Management System Data Structures and their Mapping to Physical Storage. Ph. D. dissertation, University of Michigan, Ann Arbor, 1971.
Contains a proposal for a data definition and mapping language, which is being used in the Michigan data translation experiments. See Merten/Fry.

180. Taylor, R. W. Data Administration and the DBTG Report. 1974 ACM

SIGFIDET Workshop Proc., ACM, New York, 1974.
Among others, the author proposes to use a preprocessor to obtain data independence at precompile time.

181. Taylor, R. W., and Stemple, D. W. On the Development of Data Base Editions. Data Base Management, Proc. of IFIP Work. Conf. Cargese, Corsica, April 1974. North Holland, Amsterdam, 1974.
The authors' concern is the evolution of a data base at a user installation and its impact on programs.

182. Teichroew, D. An Approach to Research in File Organization. Proc. of the 1971 SIGIR Symposium on Information, Storage and Retrieval, ACM, New York, 1971.
The essential message in this paper: there is no absolutely best representation of information. Changes as a function of knowledge about the future use of the data have to be made with assistance of the computer.

183. Thomas, J. C., and Gould, J. P. A Psychological Study of Query by Example. 1975 AFIPS NCC Proc. vol. 44, 439 - 445, 1975.
Reports the results of an experiment with 35 subjects, who were given questions in English to be translated into query by example (see Zloof).

184. Thompson, F. B., Lockemann, P. C., Dostert, B., and deverill, R. S. REL: A Rapidly Extensible Language System. ACM 1969 Natl. Conf. Proc., 399 - 417, 1969.

185. Todd, S. J. P. PRTV: A Technical Overview. IBM UKSC Peterlee, Technical Report UKSC 0075, 1975.
A new description of the experimental system IS/1.

186. Tsichritzis, P. A Network Framework for Relation Implementation. University of Toronto, Technical Report CSRG-49, February 1975.
Discusses how Codd's relational model can be implemented on top of physical networks (i.e. linked structures).

187. Turn, R., and Shapiro, N. Z. Privacy and Security in Data Bank Systems - Measures of Effectiveness, Costs and Protection-Intruder Interactions. AFIPS 1972 FJCC, vol. 41, 435 - 444.

188. Van der Pool, J. A. Optimum Storage Allocation for a File in

Steady State. IBM J. Res. Div. 17, 27 - 38, 1973.
Files with key-to-address transformations (hashing) and with overflow areas are analyzed. Storage utilization, overflow rate and other relevant factors are given for the steady state.

189. Vose, M. R., and Richardson, J. S. An Approach to Inverted Index Maintenance. Comp. Bull. 16, May 1972.

190. Wang, C. P., and Wedekind, H.H. Segment Synthesis in Logical Data Base Design. IBM J. Res. Dev. 19, 71 - 77, 1975.
The authors specify a minimal cover algorithm, which calculates a set of minimal covers to a given set of relations with transitive dependencies. Each minimal cover is again a set of relations without transitive dependencies. Given the minimum cover, a set of relations in Codd's third normal form can easily be constructed.

191. Wedekind, H. Datenorganisation. de Gruyter, Berlin, 1972.

192. Wedekind, H. Datenbanksysteme I. Bibliographisches Institut Mannheim, 1974.

193. Wedekind, H. On the Selection of Access Paths in a Data Base System. Data Base Management, Proc. of IFIP Work. Conf., Cargese, Corsica, April 1974. North Holland, Amsterdam, 1974.
The paper's concern is modeling and analysis for the determination of efficient access paths.

194. Wellis, M. E., Katke, W., Olson, J., and Yang, S. C. SIMS - an Integrated, User-Oriented Information System. AFIPS FJCC 1972, vol. 41, 1117 - 1131, 1972.
SIMS is interesting for a number of reasons. It offers a high level data definition, mapping and manipulation language. Data on normal files may be mapped to a conceptual high level hierarchical form and used in the query language. Particular attention has been paid to transferability of data and programs.

195. Wong, E., and Chiang, T. C. Canonical Structure in Attribute Based File Organizations. CACM 14, 593 - 597, 1971.
Each query is assumed to be a boolean expression over elementary queries. In this case the data base can be organized according to the elementary queries and access becomes essentially the

problem of putting a boolean expression into some standard form.

196. Yao, S. B. Evaluation and Optimization of File Organization through Analytic Modeling. Ph. D. dissertation, University of Michigan, 1974.

197. Yue, P. C., and Wong, C. K. Storage Cost Considerations in Secondary Index Selection. IBM Research Report RC 5079, to appear also in IJCIS.
For other recent results in this area of research see Cardenas.

198. Zloof, M. M. Query By Example. 1975 AFIPS NCC Proc. vol. 44, 431 - 437, 1975.
The basic features of query by example are illustrated. The user's perception of data processing in this query language is that of manipulating tables in a graphically pre-established frame of reference consisting of table skeletons, into which the user fills information.