Relational Data-Base Management Systems

DONALD. D. CHAMBERLIN

IBM Research Laboratory, San Jose, California 95193

The essential concepts of the relational data model are defined, and normalization, relational languages based on the model, as well as advantages and implementations of relational systems are discussed.

Keywords and Phrases: Data base, data-base management, data independence, data model, relational systems

CR Categories: 3.51, 4.33, 4.34

INTRODUCTION

Before describing the relational model of data, we will briefly discuss some trends in data-base management which give motivation to the development of the relational model. The first large-scale, machine-readable collections of data were stored on external media such as cards or tape. Beginning in the late fifties and early sixties, data banks were being stored on-line using direct-access devices such as disks. Generalized software packages such as BDAM and ISAM [T2] were developed to aid programmers in accessing the data. During the late sixties and early seventies, the idea of an integrated data-base management system was developed. This concept allowed several applications to share a common bank of data, maintained and protected by a central system. In an integrated data-base environment, the data-base management system (DBMS) provides each application program with its own view of the common data, implements various operators for retrieval and update of data, and resolves interference between concurrent users.

The overall trend which is visible in data-base management today is the following: users are becoming increasingly oriented toward the *information content* of their data, and decreasingly concerned with its *representation details*. Increasingly, the user interface of a modern DBMS deals with abstract information rather than with the various bits, pointers, arrays, lists, etc., which may be used to represent information. Responsibility for choosing an appropriate representation for the information is being assumed by the system and is not exposed to the end user; indeed, the representation of a given fact may change over time without the user being aware of the change. The general term for this trend away from representation details is *data independence*.

If we attempt to extrapolate the trend toward data independence, we observe that most current DBMS present the user with a view of records connected in some sort of structure, such as a network or hierarchy. In such a view, information may be represented in at least three ways:

1) by the *contents* of records (e.g., Smith's employee record has DEPTNO = 50.);
2) by the *connections* between records e.g., Smith's employee record occurs in the hierarchy below the department record for Dept. 50.); and

Copyright © 1976, Association for Computing Machinery, Inc. General permission to republish, but not for profit, all or part of this material is granted provided that ACM's copyright notice is given and that reference is made to the publication, to its date of issue, and to the fact that reprinting privileges were granted by permission of the Association for Computing Machinery.

CONTENTS

3) by the *ordering* of records (e.g., all the sales records are stored in chronological order).

User requests made to the DBMS are then framed in terms which depend on user knowledge of the representation chosen (e.g., "FIND NEXT RECORD OF ABC SET").

The relational data model makes it possible to eliminate this last representation-dependence from the user interface. In the relational model, information is represented in only one way at the user interface: by data values. User requests become free of any dependence on internal representation, and hence may be framed in a high-level, nonprocedural language. At the same time, the system becomes free to choose any physical structure for storage of data, and to optimize the execution of a given request.

A very early proposal for a representation-independent approach to file processing was made by the Language Structure Group of the CODASYL Development Committee in 1962 [Y1]. The CODASYL proposal, called "An Information Algebra," was never implemented. An early worker in a related area was David Childs, of the University of Michigan, who proposed a "set-theoretic data structure," based on a "reconstituted definition of relation" [Y3, Y4]. In Childs' system, which was implemented on an IBM 7090, the user was given sets of data-tuples (not constrained to be of common type) and set-theoretic operators, such as union and intersection, to manipulate the data.

In the late sixties, several artificial intelligence-oriented systems were implemented based on *binary* relations, which are simply collections of ordered pairs of objects related in a certain way. An example of a binary relation is:

FATHER-OF: {⟨Mary, George⟩, ⟨George, Bill⟩, ⟨John, Bill⟩}

Systems incorporating binary relations for data storage included the Relational Data File of Levien and Maron [Y2], the TRAMP system of Ash and Sibley [Y5], and the LEAP language of Feldman and Rovner [Y6].

The considerable attention paid to n-ary relations as a tool for general data-base management dates from a 1970 paper by E. F. Codd, of IBM [M2]. Codd was the first to give a rigorous definition for n-ary relations in the data-base context, and to emphasize their advantages for data independence and symmetry of access.

Codd's paper introduced concepts which set the direction for research in relational data-base management for several years to come. The paper defined a *data sublanguage* as a set of facilities, suitable for embedding in a host programming language, which permits the retrieval of various subsets of data from a data bank. The paper noted that a standard logical notation, the first order predicate calculus, is appropriate as a data sublanguage for n-ary relations. The paper also introduced a set of operators ("join," "projection," etc.) which were later developed into the well-known relational algebra. Finally, the paper explored the properties of "redundancy" and "consistency" of relations, which laid the groundwork for Codd's later theory of normalization.

DEFINITIONS

We will now discuss the basic concepts and definitions which underlie the relational data model. Many of these concepts were first introduced by Codd's original paper [M2].

ELECTIONS

YEAR	WINNER-NAME	LOSER-NAME
1952	Eisenhower	Stevenson
1956	Eisenhower	Stevenson
1960	Kennedy	Nixon
1964	Johnson	Goldwater
1968	Nixon	{Humphrey, Wallace}
1972	Nixon	McGovern

FIGURE. 1 The ELECTIONS relation.

An excellent introduction to relational concepts can also be found in Date's recent textbook [Z11].

In mathematics, the term *relation* may be defined as follows: Given sets $D_1, D_2, \ldots, D_n$ (not necessarily distinct), a *relation* R is a set of n-tuples each of which has its first element from D_1, second element from D_2, etc. The sets D_i are called *domains*. The number n is called the *degree* of R, and the number of tuples in R is called its *cardinality*.

It is customary (though not essential) when discussing relations to represent a relation as a table in which each row represents a tuple. An example of this representation is shown in Figure 1, which illustrates a relation describing Presidential elections. In the tabular representation of a relation, the following properties, which derive from the definition of a relation, should be observed:

1) no two rows are identical;
2) the ordering of rows is not significant; and
3) the ordering of columns is significant (i.e., the meanings of the tuples ⟨1972, Nixon, McGovern⟩ and ⟨1972, McGovern, Nixon⟩ are quite different).

When a relation is represented as a table, its degree is the number of columns and its cardinality is the number of rows.

In the tabular representation of a relation, it is customary to name the table and to name each column, as shown in Figure 1. The columns of the table are called *attributes*. (Sometimes the name of a column is referred to as a *role name*.) It is important to distinguish between attributes and domains. For example, in the ELECTIONS relation of Figure 1, the second and third columns are both based on the same domain: the set of names of Presidential candidates. However, each column has a different role-name to describe its meaning in this particular relation: WINNER-NAME and LOSER-NAME.

The individual entries in each tuple are called its *components*. Thus, we may say that in the tuple whose YEAR-component is "1952," the LOSER-NAME-component is "Stevenson."

A column or set of columns whose values uniquely identify a row of a relation is called a *candidate key* (often shortened to simply *key*) of the relation. In Figure 1, YEAR is a key for ELECTIONS since no two rows have the same YEAR. It is possible for a relation to have more than one key. For example, if the ELECTIONS relation had an additional column ADMINISTRATION-NUMBER, it would also be a key. When a relation has more than one key, it is customary to designate one as the *primary key*.

Often a column or set of columns in one relation will correspond to a key of another relation. For example, consider the PRESIDENTS relation of Figure 2, whose key is NAME. The values of WINNER-NAME in the ELECTIONS relation correspond to values of the key-column NAME in PRESIDENTS. Consequently, WINNER-NAME in ELECTIONS is called a *foreign key*. Two facts should be noted: 1) a foreign key need not be (and often is not) a key of its own relation; and 2) the foreign key need not have the same role-name (e.g., WINNER-NAME) as the corresponding key in the other relation (e.g., NAME).

In an integrated data-base management system, different users may have a need to

PRESIDENTS

NAME	PARTY	HOME-STATE
Eisenhower	Republican	Texas
Kennedy	Democrat	Massachusetts
Johnson	Democrat	Texas
Nixon	Republican	California

FIGURE 2. The PRESIDENTS relation.

see different subsets of the universe of data. The term *data model* denotes the universe of data—the complete set of relations stored in the system. A *schema* is a set of declarations which describe the data model. The term *data submodel* denotes the set of relations which is available to a particular user, and a *subschema* is a set of declarations for the data submodel. A complete data-management system must provide a means for defining the schema and a subschema for each distinct class of users of the system.

NORMALIZATION

The issue of designing a schema and subschemas for a data base leads us to a discussion of *normalization*. The concept of normalization was introduced by Codd in [M2] and dealt with more rigorously in his later papers [N1] and [N2]. A number of other authors have also made contributions to the theory of normalization (see bibliography).

Normalization theory begins with the observation that certain collections of relations have better properties in an updating environment than do other collections of relations containing the same data. The theory then provides a rigorous discipline for the design of relations which have favorable update properties. The theory is based on a series of *normal forms*—first, second, and third normal form—which provide successive improvements in the update properties of a data base. We will discuss these normal forms on an intuitive basis; for a thorough treatment, see [N1], [N8], or [Z11].

Almost all references to relations implicitly deal with relations in *first normal form*. A relation in first normal form is a relation in which each component of each tuple is nondecomposable; i.e., the component is not a list or a relation. Relations in first normal form are sometimes called "flat tables". If we look carefully at the relation in Figure 1, we see that it is not in first normal form. This is because an election, while it has only one winner, may have several losing candidates. Thus, for example, the tuple for the election of 1968 contains the component {"Humphrey", "Wallace"}. In fact, the LOSER-NAME component of each election tuple is a list whose length depends on the number of votes a candidate must receive to merit inclusion in the data base.

We can convert the ELECTIONS relation into first normal form by breaking it up into two relations, one containing information on winning candidates and the other on losing candidates. This also gives us a good opportunity to record other attributes of interest about the candidates, such as their party and number of votes received. This leads us to the data base shown in Figure 3, which is in first normal form. The key of ELECTIONS-WON is YEAR; the key of ELECTIONS-LOST is ⟨YEAR, LOSER-NAME⟩.

To illustrate the advantages of the higher normal forms, we need to make updates to the data base by inserting new tuples, deleting existing tuples, and making changes to existing tuples. These updates are not particularly well motivated for our example data base, in which data is mostly static and unchanging. Of course, in an operational data base describing, for example, the inventory of a store, updates would be very frequent. For the sake of consistency, we will continue with our Presidential example. (You may imagine that some data was found to be in error and is being updated to correct the data base.)

Relations in first normal form may be used with any of the relational languages which are described in the next section.

However, a relation in first normal form may exhibit three kinds of misbehavior, which are called update anomalies, insertion anomalies, and deletion anomalies. All these anomalies arise because more than one "concept" may be mixed together in the same tuple. Consider the ELECTIONS-WON relation of Figure 3. Mixed together in one tuple of this relation are facts about *candidates* (e.g., "Eisenhower came from Texas") and facts about *elections* (e.g., "In 1952 Eisenhower received 442 electoral votes"). In some applications it may be important that each of these facts be independently updated, inserted, and deleted. This gives rise to the three anomalies, which we can now illustrate by the following examples.

Update anomalies: Suppose the fact that "Eisenhower's home state is Texas" is found to be in error, and his home state must be changed to Nebraska. Since Eisenhower appears in more than one tuple of ELECTIONS-WON, this erroneous fact may be represented many times (in general, a time-varying number of times). This makes it difficult to update this particular fact, since all tuples where it is represented must be searched out and updated. Even worse, it leads to the possibility that different tuples may contain inconsistent values of HOME-STATE for the same President.

Insertion anomalies: Suppose we wish to insert a fact about a candidate which is independent of any election, e.g., "Dewey was a Republican." This is difficult in our example data base because there is no relation for candidates. We are forced to invent a tuple in ELECTIONS-LOST (or ELECTIONS-WON?) having null values for YEAR and the other irrelevant attributes. In many systems we would be unable to store this fact because null values are not permitted in the primary key.

Deletion anomalies: Suppose we wish to delete the information about elections as they fall beyond a certain number of years in the past. When we delete the 1952-tuple from ELECTIONS-WON, we still retain the fact that Eisenhower was a Republican. But when we delete the 1956-tuple, all facts about Eisenhower are lost. In some applications, this might have very serious consequences. For example, consider a relation describing orders for various items, shown in Figure 4. As orders are filled we delete their tuples from the relation. When we have deleted the last order for toasters,

ELECTIONS-WON

YEAR	WINNER-NAME	WINNER-VOTES	PARTY	HOME-STATE
1952	Eisenhower	442	Republican	Texas
1956	Eisenhower	447	Republican	Texas
1960	Kennedy	303	Democrat	Mass.
1964	Johnson	486	Democrat	Texas
1968	Nixon	301	Republican	Calif.
1972	Nixon	520	Republican	Calif.

ELECTIONS-LOST

YEAR	LOSER-NAME	LOSER-VOTES	PARTY
1952	Stevenson	89	Democrat
1956	Stevenson	73	Democrat
1960	Nixon	219	Republican
1964	Goldwater	52	Republican
1968	Humphrey	191	Democrat
1968	Wallace	46	Am. Indep.
1972	McGovern	17	Democrat

FIGURE 3. Elections data base in first normal form.

ORDERS

ITEM	PRICE	DATE	QUANTITY-ORDERED
Toaster	20.00	1/10/75	2
Toaster	20.00	2/15/75	5
Mixer	28.00	4/6/75	3

FIGURE 4. The ORDERS relation.

we find we no longer have any information about the price of toasters—possibly an unintended result. This kind of relation burdens the user with the responsibility of making sure that the tuple he deletes is not the last tuple of some "category" (e.g., toasters), and therefore the sole bearer of information about that category (e.g., price).

An important objective of normalization is the elimination of the update, insertion, and deletion anomalies. The most widely-known result of normalization theory is third normal form. Since second normal form is of little significance except as a stopping-off place on the way to third, we will proceed directly to the definition of third normal form.

In order to understand how third normal form avoids the three anomalies, we must discuss the concept of *functional dependence* among the attributes of a relation. We say that an attribute B of relation R is *functionally dependent* on attribute A if, at every instant of time, each A-value in R is associated with only one B-value. We express this relationship by the notation A → B, and say "A *determines* B" or "B *depends on* A." Similarly, a *set* of attributes in R may be functionally dependent on another attribute or set of attributes. The attribute (or set of attributes) on the left side of the arrow (A in our example) is called the *determinant*.

Clearly, from our definition of *key* in the previous section, every relation contains at least one functional dependence: all attributes of the relation are dependent on the key. (The dependence may be trivial if the relation contains only a key.) If a relation has more than one key, then all its attributes are dependent on each key.

Third normal form has been defined in a variety of ways. The original definition was given by Boyce and Codd in [N1]. Later writers, including Kent [N8], Codd [M14], and Sharman [N15], proposed alternate definitions which framed the same concept in simpler terminology. We present two of these equivalent definitions:

Definition, Boyce and Codd [M14]:
A relation R is in third normal form if it is in first normal form and, for every attribute collection C of R, if any attribute not in C is functionally dependent on C, then all attributes in R are functionally dependent on C.

Definition, Sharman [N15]:
A relation is in third normal form if every determinant is a key.

Both definitions are formal ways of expressing a very simple idea: that each relation should describe a single "concept," and if more than one "concept" is found in a relation, the relation should be split into smaller relations. The result of applying this "splitting" process to the sample data base of Figure 3 is shown in Figure 5. A moment's examination will show that the update, insertion, and deletion anomalies we discussed are not present in the data base of Figure 5.

The design of a data base in third normal form depends on knowledge of the functional dependencies among the attributes of the data. This knowledge cannot be discovered automatically by a system (unless the data base is completely static), but must be furnished by a data-base designer who understands the semantics of the information. In fact, there is not a unique third normal form representation for a given data base. In [N1] Codd briefly addressed the problem of choosing an "Optimal Third Normal Form" from among the various alternatives.

LANGUAGES

Such a great variety of relational languages is available that it would be impossible to treat them all here. We will describe a representative example of several important categories of relational languages; references to other languages can be found in the bibliography.

The term *data sublanguage*, introduced earlier, denotes a set of data-base operators intended to be embedded in a host programming language. The term *query language* usually refers to a stand-alone language in which an end user interacts directly with the data-base management system. Most query languages provide a variety of facilities (e.g., update, creation, and deletion of relations) in addition to a query capability. As compared with a typical data sublanguage, a query language is usually at a higher level, less procedural, and intended for a more casual user. Sometimes, however, the same basic set of opera-

ELECTIONS-WON

YEAR	WINNER-NAME	WINNER-VOTES
1952	Eisenhower	442
1956	Eisenhower	447
1960	Kennedy	303
1964	Johnson	486
1968	Nixon	301
1972	Nixon	520

PRESIDENTS

NAME	PARTY	HOME-STATE
Eisenhower	Republican	Texas
Kennedy	Democrat	Mass.
Johnson	Democrat	Texas
Nixon	Republican	Calif.

ELECTIONS-LOST

YEAR	LOSER-NAME	LOSER-VOTES
1952	Stevenson	89
1956	Stevenson	73
1960	Nixon	219
1964	Goldwater	52
1968	Humphrey	191
1968	Wallace	46
1972	McGovern	17

LOSERS

NAME	PARTY
Stevenson	Democrat
Nixon	Republican
Goldwater	Republican
Humphrey	Democrat
Wallace	Am. Indep.
McGovern	Democrat

FIGURE 5. Data base in third normal form.

tors can serve both as a data sublanguage and as a query language.

This section will explore the approach taken by various relational languages to providing facilities for query, data manipulation (e.g., insertion, deletion, and update of tuples), data definition (e.g., creation of new relations and other structures), and data control (e.g., authorization and control of data integrity). We will then briefly consider some ways in which languages can be evaluated and compared, and discuss the role of natural language as a data-base interface.

Query Facilities

Query, or retrieval of information from the data base, is perhaps the aspect of relational languages which has received the most attention. We will illustrate the variety of approaches to query by presenting examples of four classes of languages: relational calculus, relational algebra, mapping-oriented languages, and graphics-oriented languages. Although we deal only with query facilities in this section, all the languages discussed have facilities for update and other operations in addition to query.

Relational Calculus

Codd's 1970 paper [M2] laid the groundwork for two families of relational languages which came to be called the relational calculus and the relational algebra. The relational calculus family grew from the observation that a first-order applied predicate calculus can be used as a data sublanguage for normalized relations. In [L1] Codd presented the details of such a calculus-based sublanguage, called ALPHA.

A typical query in ALPHA has two parts: a *target*, which specifies the particular attributes of the particular relation which are to be returned, and a *qualification*, which selects particular tuples from the target relation by giving a condition which they must satisfy. We will illustrate ALPHA (and other languages) by some sample queries based on the data base of Figure 5.

In Q1) below, the RANGE statement declares P be a variable ranging over the rows of the PRESIDENTS relation. The next statement retrieves into workspace W the HOME-STATE of row P whenever the NAME of row P is "KENNEDY."

The qualification part of an ALPHA query may be quite complex and may use the universal and existential quantifiers: "for all" (∀), and "there exists" (∃). For example, see display Q2) below.

Various other languages based, like ALPHA, on the relational calculus, have been proposed. This class of languages includes QUEL [S15], COLARD [L3], and RIL [L7].

Relational Algebra

A second major class of languages is based on the relational algebra, which was introduced by Codd in [M2] and refined in [M3]. The relational algebra is a collection of operators that deal with whole relations, yielding new relations as a result. The major operators of relational algebra include the following:

- *Projection:* The projection operator returns only the specified columns of the given relation, and eliminates duplicates from the result. For example, to find all the unique ⟨party, home-state⟩ pairs in the PRESIDENTS relation,

Q1) What was the home state of President Kennedy?

```
RANGE PRESIDENTS P
GET W P. HOME-STATE: (P.NAME='KENNEDY').
```

Q2) List the election years in which a Republican from Illinois was elected.

```
RANGE PRESIDENTS P
RANGE ELECTIONS-WON E
GET W E.YEAR: ∃P (P.NAME=E.WINNER-NAME &
  P. PARTY='REPUBLICAN' & P. HOME-STATE='ILLINOIS').
```

we might write the following projection:

PRESIDENTS [PARTY, HOME-STATE].

(Note that some algebra-based languages use column numbers rather than column names. In such languages we would write PRESIDENTS [2, 3] in place of the given expression. This notation, although less mnemonic, has the advantage of avoiding ambiguity if some intermediate result has two columns with the same name.)

- *Restriction:* The restriction operator selects only those tuples of a relation which satisfy a given condition. As originally proposed, the condition only allowed comparison of one component of a tuple with another component. Some implementations of the algebra permit other condition-types as well, e.g., comparison of a tuple-component with a constant. For example, to select those tuples from the ELECTIONS-WON relation where YEAR is greater than 1945, we might write:

ELECTIONS-WON [YEAR > 1945].

- *Join:* The join operator takes two relations as arguments, which we will refer to as relations A and B. A new relation is formed by concatenating a tuple of A with a tuple of B wherever a given condition holds between them. For example:

ELECTIONS-WON [WINNER-NAME = NAME] PRESIDENTS.

This expression concatenates a tuple of ELECTIONS-WON with a tuple of PRESIDENTS whenever WINNER-NAME in ELECTIONS-WON matches NAME in the PRESIDENTS tuple. If a given PRESIDENTS tuple matches more than one ELECTIONS-WON tuple, it is concatenated with each of them, forming multiple output rows. If a given tuple matches no tuple in the other relation, it does not participate in the output at all.

- *Set-theoretic Operators:* In relational algebra, the set-theoretic operators—union, intersection, and set-difference—take two relations as operands, treating each as a set of tuples, and produce a single relation as a result. The operand relations must have compatible sets of attributes.
- *Division:* Some algebraic languages include an operator, called "division," which operates on two input relations to produce a third relation. This operator is sometimes useful in expressing queries which contain the word "all." However, since it can be expressed in terms of the other algebraic operators, the division operator does not extend the logical power of the language. The reader is referred to [M3] for a complete treatment of division.
- *Nesting:* The algebra has the convenient property that its operators can be nested to form expressions of arbitrary complexity, with parentheses used as needed to remove ambiguities. To illustrate nesting of operators, we will repeat examples Q1) and Q2) (displayed below) in the relational algebra:

Languages based on the relational algebra have been implemented at MIT [S1] and [S12], the IBM Scientific Centre in England [S4] and [S16], and General Motors Research Laboratory [S5]. In addition, studies of optimization algorithms for the relational algebra have been published by Smith and Chang [T18], Pecherer [T16], Gotlieb [T17], and others.

Q1) What was the home state of President Kennedy?
PRESIDENTS [NAME = 'KENNEDY'] [HOME-STATE]

Q2) List the election years in which a Republican from Illinois was elected.
(ELECTIONS-WON [WINNER-NAME = NAME] PRESIDENTS)
[PARTY = 'REPUBLICAN'] [HOME-STATE = 'ILLINOIS'][YEAR]

Mapping-Oriented Languages

A third class of relational languages, called "mapping-oriented" languages, has been proposed by R. F. Boyce and others [L9]. These languages, directed at the nonprogramming professional, offer power equivalent to that of the relational calculus or algebra while avoiding mathematical concepts such as quantifiers. Mapping-oriented languages include: SQUARE, a terse, APL-like notation [L25]; SEQUEL, a structured language based on English keywords [L10, L8]; and SLICK, a language intended for implementation on associative hardware [L12]. We will illustrate this class of languages by presenting examples of SEQUEL.

The basic building block of mapping-oriented languages is the "mapping," which maps a known attribute or set of attributes into a desired attribute or set of attributes by means of some relation. Q1) is an example of a simple mapping:

Q1) What was the home state of President Kennedy?

```
SELECT  HOME-STATE
FROM    PRESIDENTS
WHERE   NAME = 'KENNEDY'.
```

In general, the result of a mapping may be used in the specification of another mapping, as shown in Q2) below. This process of "nesting" mappings inside each other makes it possible to express queries of great complexity.

Graphics-Oriented Languages

Recently, another important class of relational languages has been proposed: the class of "graphics-oriented" languages. In these languages, the user states his query not by a conventional linear syntax, but by making choices or filling in blanks on a graphic display. Examples of this class of languages are Query By Example [L21, L24] and CUPID [L17]. We will illustrate this type of language by presenting examples using Query By Example.

In Query By Example, the user is presented with a blank relation on his display. He fills in one or more rows of the relation with an example of the desired result. Known values are filled in directly. Unknown values are represented by arbitrarily chosen example values, which are underscored to show that they are examples. The attributes to be printed are identified by a "P." A query may be confined to a single relation, or span more than one relation, as illustrated by Q1) and Q2) at top of page 53.

Data Manipulation

Most relational languages provide facilities for data manipulation, which includes insertion, deletion, and update of tuples. Since update is not well motivated for the Presidential data base, we introduce the following relation to illustrate data manipulation:

```
EMP (EMPNO, NAME, JOB,
  SALARY)
```

This relation describes a set of employees, giving, in each instance, his or her employee number, name, job, and salary.

Many languages with set-oriented query features also allow set-oriented data manipulation. For example, the following state-

Q2) List the election years in which a Republican from Illinois was elected.

```
SELECT  YEAR
FROM    ELECTIONS-WON
WHERE   WINNER-NAME=
          SELECT  NAME
          FROM    PRESIDENTS
          WHERE   PARTY='REPUBLICAN'
          AND     HOME-STATE='ILLINOIS'
```

Q1) What was the home state of President Kennedy?

PRESIDENTS

NAME	PARTY	HOME-STATE
KENNEDY		P. NEVADA

Q2) List the election years in which a Republican from Illinois was elected.

ELECTIONS-WON

YEAR	WINNER-NAME	WINNER-VOTES
P.1948	WILSON	

PRESIDENTS

NAME	PARTY	HOME-STATE
WILSON	REPUBLICAN	ILLINOIS

ment in SEQUEL [L10] has the effect of giving a 10% raise to all programmers:

```
UPDATE  EMP
SET     SALARY = SALARY*1.1
WHERE   JOB = 'PROGRAMMER'
```

All the languages we have discussed so far have been high level and nonprocedural in nature. Indeed, one of the advantages of the relational model is that it is readily compatible with high-level languages. But it should not be concluded that the relational model is incompatible with a lower-level, more procedural programming interface. In fact, several low-level, host-language relational interfaces have been proposed, including GAMMA-0 [L4], XRM [S6], and MINIZ [S8]. These interfaces are well suited for writing programs that are to be called repeatedly and which update the data base according to parameters furnished with the call.

We will illustrate how one low-level relational language, GAMMA-0, might be used to write a transaction which finds the employee-tuple having a given employee number and updates its salary component according to some computation. GAMMA-0 consists of a set of operators which may be called from a host language such as PL/I. GAMMA-0 is based on the concept of a "scan," which is like a cursor that moves through a relation testing tuples for some condition. Our first call to GAMMA-0 uses the operator CREATE-SCAN, which creates a scan on the EMP relation to search for tuples according to their EMPNO attribute. The system returns an identifier, called a SCANID, by which we may refer to the newly created scan in future calls. Next we call the operator SET-SCAN and furnish the value which is to be searched for (in this case the EMPNO, which is the parameter of our transaction). Our next call is to the operator NEXT-SUBTUPLE, which returns an actual tuple satisfying the criterion we established by the previous calls. (NEXT-SUBTUPLE could be called repeatedly if we expected many tuples to satisfy the criterion.) Having obtained the desired employee-tuple, we can compute a new salary-value in our host program and then call UPDATE SUBTUPLE, which puts the new salary-value into the data-base. GAMMA-0 allows a program to have as many active scans as it wishes, and to control the position of each by explicit calls. When a program has no further use for a scan, it may drop it by calling the operator DROP-SCAN.

Although it is a low-level, procedural language, GAMMA-0 is considered a relational language because the means of access to tuples is not predetermined. A relation may be accessed associatively through any of its attributes—the attribute to be matched is declared when a scan is opened.

Data Definition and Control

In addition to query and data manipulation facilities, a complete data sublanguage needs facilities for data definition and data control. Data definition has two main aspects:

- Specification of the characteristics of data to be stored, e.g., the column-names and data-types for each relation; and
- definition of alternative "views" which are derived from the stored data. In relational terminology, a view is a dynamic "window" on the data base. Updates made to stored relations are visible through the various views which are defined on these relations.

Data control also has two main aspects:

- control over authorization of various users to perform various operations on the data base; and
- ability to make integrity assertions that protect the validity of data and define the set of permitted transitions in the data base.

The relational model permits a language to take a consistent, unified approach to query, data manipulation, data definition, and data control. Several relational languages have gone to great lengths to provide such a unified approach; these languages include SEQUEL [L10, L8, C6, I5], QUEL [S15, C3, I4], and Query By Example [L21, L24].

An important observation to be made in data definition is that the definition of a view is simply a process of deriving a relation from the set of stored relations, and that this is similar to the process of stating a query. Therefore, the full power of a query language may be applied to the definition of views. This is possible because all the relational query languages we have discussed have the property of *closure*, i.e., they operate on relations to construct or define new relations. A view may be a selected subset of a stored relation, or it may span over more than one stored relation, as in the case of a join. Once the definition of a view has been made, queries may be directed to the view as though it were a stored relation. The supportability of updates to the data base made by means of derived views is a complicated question, one which requires more research [M14].

The issue of authorization is closely related to the issue of derived views. In fact, one approach to authorization is to grant to each user a particular restricted view [C6]. Another approach is to automatically add certain predicates to the queries and updates issued by a user in order to restrict their scope to the set of authorized tuples [C3].

This unified approach to language design can be extended into the area of assertions concerning data integrity. An assertion is a statement about the data base which the system automatically enforces by refusing any update which fails to satisfy the assertion. In language terms, an assertion is simply a predicate, which is syntactically a fragment of a query, and which may contain other queries nested inside it. For example, suppose we wish to assert that for any given election the number of votes received by the winner is greater than the number of votes received by any loser. This assertion may be made as follows in SEQUEL (the variable X represents a tuple of the ELECTIONS-WON relation):

```
ASSERT ON ELECTIONS-WON X:
  WINNER-VOTES >
    (SELECT   MAX (LOSER-VOTES)
    FROM      ELECTIONS-LOST
    WHERE     YEAR=X.YEAR)
```

Language Evaluation

The great variety of proposed relational languages leads us to the question: How can languages be evaluated and compared? There are at least three criteria involved in any objective attempt to evaluate a language: completeness, level, and learnability. Space constraints permit us to touch only briefly on each of these.

Codd [M3] was the first to establish a careful definition of completeness for data-base sublanguages. He defined a language to be *relationally complete* if it permits expression of any query expressible in the

relational calculus. He then proceeded to prove that the relational algebra was relationally complete and hence could serve as the standard of comparison for completeness of algebra-oriented languages. Since the appearance of this early work, proofs of relational completeness have been published for the SQUARE and Query By Example languages [L25], [L13].

The first attempt at a quantitative definition of language "level" was made by Halstead in his investigation of "software physics" [L23]. According to Halstead's definition, "level" is a property of a particular expression of an algorithm. The "simplest conceivable" expression of a given algorithm is assigned a level of 1, and more complicated expressions of the same algorithm are given level-values ranging from 0 to 1, computed on the basis of parameters such as the number of operators and operands used in expressing the algorithm. In [L14], Halstead applies the formulas of software physics to a comparison of Codd's ALPHA language [L1] and DBTG-COBOL [Z1].

The last method of language evaluation we will discuss is that of psychological tests in which the language is taught to a group of subjects under controled conditions and their learning progress is measured. The emphasis of the experiment may be placed on measuring speed of learning or degree of comprehension, or on identifying particular language features which seem to cause learning difficulties. Studies of this type have been published on the languages SQUARE and SEQUEL [L20], and Query By Example [L22].

Natural Languages

Recently there has been considerable interest in the use of a natural language such as English as a query language. The relational data model is well adapted to such an attempt because it contains no implementation-oriented concepts. Most natural-language-oriented query systems, such as REL [E4] and CONVERSE [E5], attempt to translate, without feedback to the user, from natural language into a computer-oriented language. E. F. Codd and J. M. Cadiou are presently developing a system called RENDEZVOUS [E9] which engages in an English dialog with the user to help him develop an unambiguous formulation of his query.

ADVANTAGES

We can now review and summarize the advantages of the relational model for data-base management. Relations have four primary advantages:

1) *Simplicity:* This term should need no further explanation. The relational user is presented with a single, consistent data structure. He formulates his requests strictly in terms of information content, without reference to system-oriented complexities.
2) *Data independence:* C. J. Date [Z11] has defined data independence as "immunity of applications to change in storage structure and access strategy." As we have seen, the relational model makes it possible to eliminate the details of storage structure and access strategy from the user interface.
3) *Symmetry:* Data-base systems which are based on connections between records make some questions easier to ask than others—namely, questions whose structure matches that of the data base. For example, in a hierarchic data base, the easiest question to ask is a question that begins at the root of the tree and moves toward the leaves, applying successive qualifications at each step. Questions not reflecting this preferred structure can be asked awkwardly if at all. Since all information is represented by data values in relations, there is no preferred format for a question at the user interface. It should be noted here that symmetry of the data model does not necessarily imply symmetry of the underlying physical data structures maintained by the system. A data-base designer may choose to optimize the performance of some frequently posed query-type (for example, by providing an index for a certain attribute). The

important thing to note is that such optimizations do not appear in the user interface.

4) *Strong theoretical foundation:* The relational data model rests on the well-developed mathematical theory of relations and on the first-order predicate calculus. This theoretical background makes possible the definition of relational completeness and the rigorous study of good data-base design (normalization).

The relational model also has a variety of secondary advantages which derive from the fundamental advantages just outlined. Perhaps the most important of these is the ease with which high-level, nonprocedural relational languages may be defined. Because they are easy to learn and use, high-level languages make data bases available to a new class of casual users who lack the training required by conventional programming languages. High-level languages also give the system maximum flexibility to optimize the execution of a given request, and to adapt the stored data structures to the changing needs of the user population. The nonprocedural approach to language design permits a unified treatment of data definition, manipulation, and control, as discussed in the section on "Languages" (pages 49–55). Finally, high-level languages make it easy to define and manipulate views of data which are not directly supported by physical structures. (Of course, many of these advantages may also be obtained by the use of high-level languages not based on the relational data model.)

One additional advantage of relations will be mentioned here. The relational model makes it possible to draw a clear distinction between data semantics and data structure. For example, the semantics of a data base may be such that when a department record is deleted, all employee records for that department should also be deleted. In the relational model such semantic rules can be stated independently of data-base structure. In some other data models (e.g., if employee records occur hierarchically under department records) this type of semantic rule is closely related to (and often constrained by) the data structure.

IMPLEMENTATIONS

The greatest open research question of the relational data model is whether it can be implemented to form an efficient and operationally complete data-base management system. Many individuals and groups have made contributions to this area of research; unfortunately, space limitations only permit mention of some of the major landmarks and large, ongoing projects in the implementation of relations. For references to many systems not discussed here, see the Implementations sections of the bibliography; special attention should be given to [S19], which is a transcript of a panel discussion among implementors of relational systems.

Since the earliest n-ary relational languages proposed were Codd's relational algebra and relational calculus, it is natural that much of the earliest implementation work was directed toward these languages. In [M3], Codd observed that the relational calculus has many advantages over the relational algebra from the end-user's point of view, but that the relational algebra provides a sequence of operations which can be more directly implemented on a machine. In [M3], Codd also provides an algorithm, called a "reduction algorithm," for translating a relational calculus expression into a sequence of operations in relational algebra. This approach was extended by Palermo [T6], who made certain improvements in the efficiency of the reduction algorithm and implemented the operators of the relational algebra using APL/360.

A number of early projects in relational data-base management adopted the approach of implementing the relational algebra directly. Perhaps the earliest of these was the MACAIMS system, developed by Goldstein and Strnad at MIT [S1]. The MACAIMS system, implemented on MULTICS, introduced the important concept of encoding each data item by a fixed-length identifier, and using these identifiers rather than the actual data items in stored relations. MACAIMS also made a contribution to the field of data independence by enabling different relations to be stored in different forms and converted to a canonical form, when necessary, for comparison. More

recent developments in the use of relational algebra at MIT are presented in [S12].

Another early algebra-oriented system is the Relational Data Management System (RDMS) of General Motors [S5]. RDMS is a display-oriented query system which implements not only the operators of the relational algebra, but also a number of other set-oriented operators such as SORT, GRAPH, and HISTOGRAM.

An ongoing project in the implementation of relational algebra is located at the IBM Scientific Centre in Peterlee, England. The Peterlee system was first called IS/1 and later renamed the Peterlee Relational Test Vehicle (PRTV) [S4, S16]. The system has been used in an environmental research study with a data base of ten million characters [A1], as well as by the Greater London Council in an urban planning application having a data base of 50 million characters [S19]. In addition to the usual algebraic operators (join, projection, etc.), PRTV provides an easy means to extend the system by adding new relational operators. A user may construct temporary relations by applying various operators either to stored relations or to existing temporary relations. The definition of a temporary relation is kept in the form of a tree of operators, and the actual tuples are not materialized until they are needed for output. An optimizer may rearrange the operators in the definition of a temporary relation, e.g., choosing to do restriction as early as possible and reordering as late as possible. PRTV allows different visible subsets of the data base, but does not permit simultaneous use of the system by more than one user.

The problem of optimizing the execution of relational systems has recently attracted a great deal of interest. Smith and Chang, based at the University of Utah [T18], have applied techniques of automatic programming to transform relational algebra expressions into equivalent but more efficient expressions. Gotlieb, working at the University of Toronto [T17], has published a study of various algorithms for implementing the join operator. Rothnie, of MIT and the Defense Department Computer Institute [T8, T20], has developed an algorithm for limiting the search space for queries in relational calculus which span more than one relation.

In addition to the work on high-level languages, such as the algebra and calculus, efforts have been made to develop a lower-level, procedural, relational interface for host-language systems, or to serve as an intermediate interface in implementing some other relational language. The first such interface to be implemented was the Relational Memory (RM) developed by IBM in Cambridge, Massachusetts [S2, S3]. RM permits variable-length byte strings (entities) to be stored and referenced by numeric identifiers. Binary relations whose data-elements are integers or entity-identifiers may then be constructed. RM provides efficient associative access to the binary relations: a hashing technique is used to locate a given "left-side" value, and all its associated "right-side" values are then accessed by means of a linked list. RM also provides a recovery capability for restoring the data base to an earlier state in the event of a failure.

In 1973 the RM system was extended to support n-ary relations; the resulting system was named XRM (Extended Relational Memory) [S6]. XRM uses the "entities" of RM to store n-ary tuples; it also uses RM binary relations as "inversions" which provide efficient associative access to these n-tuples. XRM maintains a "master relation" which describes the various relations and inversions in the system. A user may access a tuple associatively by its key-value (or the data-value in some inverted column), or may scan over a relation, retrieving all tuples which satisfy a given condition.

XRM was used as the underlying access method in a prototype system developed at IBM Research in San Jose, which implements the SEQUEL data sublanguage [S10, S11]. The SEQUEL system, which became operational in 1974, provides set-oriented facilities for: query, insertion, deletion, and update; dynamic creation and dropping of relations; and automatic enforcement of assertions about data integrity. These features are made available either as a stand-alone, display-oriented interface for casual users, or as a host-language interface that can be called from PL/I programs. The system contains an optimizer (described in

[S11]) which uses XRM inversions to limit the search space for a given query. The SEQUEL prototype has been extended by IBM at Cambridge and by the MIT Sloan School of Management to accommodate a multiple-user environment. The resulting system, called GMIS, is being used at MIT as an information system for modeling New England energy resources. [A12, S19].

Another prototype system based on XRM is being developed at IBM Research in Yorktown Heights, to implement Query By Example. The system contains an optimizer which interprets Query By Example queries in terms of operations similar to those of the relational algebra (join, restriction, etc). At present, the system supports only a single user and does not provide update facilities.

A large-scale prototype data-base management system, called System R, is presently under construction at IBM Research in San Jose [S20]. System R is the first attempt to apply the relational data model to an environment of many concurrent users and a high volume of requests. It will provide an operationally complete data-management capability, with facilities for authorization, logging and recovery, definition of alternative views, and enforcement of data consistency and integrity. System R will support the SEQUEL language as an external interface, as well as a set of procedural operators for host-language programming. Requests to the system will be executed by an optimizer which chooses among various physical access methods, including inversions maintained in the form of B-trees [T1], physical pointer-chains, and a sort-merge facility. A user is not constrained to protect himself against the updates of other concurrent users by explicit locking statements; the system automatically generates locks as needed at the level of individual tuples. Deadlocks are automatically detected and resolved. Some of the locking techniques developed as part of the System R project have been described in [C1, C4, C8]. System R is being implemented on an IBM 370, using a VM/370 operating system modified for the data-base environment [T13].

Another large-scale attempt at constructing a relational prototype is the INGRES (Interactive Graphics and Retrieval System), of the University of California at Berkeley [S7, S9, S15]. INGRES, which runs on a PDP-11/40 under the UNIX operating system, implements QUEL, a relationally complete query language based on the relational calculus. The INGRES system implements a variety of features by automatic modification of the QUEL statement submitted by the user. Alternative views are supported by substituting the view-definition into the user's statement [I4]. Authorization and integrity control are provided by adding extra predicates to the user's statement which limit its scope [C3]. Concurrent update requests are kept from interfering with each other by analyzing their respective scopes and allowing an update to proceed only when it is "safe" [I2]. Finally, the QUEL statement, which may contain many variables, is broken up by a "decomposition" algorithm into a series of one-variable statements which are executed one at a time. The physical data structures used by INGRES include hashed tables (including "order-preserving" hash functions which permit sequential scanning in key-value order) and "generalized directories," which employ a tree-structure to map a key into an address interval, and then use an order-preserving function to compute an address within the interval [S9].

Implementation of another relational system, called ZETA, is presently under way at the University of Toronto [S8, S14]. The ZETA system is constructed in three levels. The lowest level is a language called MINIZ, which provides such basic operations as scanning a relation and accumulating a list of identifiers of tuples which satisfy a given condition. The middle level implements views ("derived relations") and has an optimizer/interpreter which accepts queries spanning multiple relations. Three types of end-user interfaces are supported by ZETA:

- a host-language facility which provides features similar to SEQUEL;
- a query language generator system whereby a user may create his own self-contained query language using

a syntax-driven compiler/compiler; and

- a natural-language recognition system based on semantic networks. This system, called TORUS, is presently being tested on a data base of student records.

A second relational system, called OMEGA, [T23], is also being implemented at Toronto on a PDP-11/45. Like ZETA, OMEGA has a multilevel architecture. One of the internal levels of OMEGA is the Link and Selector Language (LSL) [T12], an expression-oriented language which provides subsetting operations on a relation ("selectors") and connections from one relation to another ("links").

A recent, and very promising, development is the emergence of several designs for associative hardware to support relational data bases. One such proposal, called CASSM (Context Addressed Segment Sequential Memory) was made by Su, et al, at the University of Florida [H1, H2, H5]. CASSM is an array of processors, each having access to a circular memory space (e.g., a disk track or circulating magnetic bubble register). As data circulates in the memory, the processors search in parallel for data which satisfies a given condition. In [L12], Copeland and Su discuss implementation of a high-level, mapping-oriented relational language called SLICK, superimposed on CASSM.

A similar design, called RAP, which is also based on a cellular array of processors with circulating memories, was recently reported by Ozkarahan, et al., of the University of Toronto [H3].

A third associative hardware system for relational applications, called RARES (Rotating Associative Relational Store), has been proposed by Lin and Smith at the University of Utah [H4]. Like CASSM and RAP, RARES contains multiple rotating memory tracks with a read/write head per track. However, unlike CASSM and RAP, which store tuples in a linear fashion along a track, RARES lays each tuple across many tracks so that the entire tuple is read in one character-read-time. Each tuple, read from memory, is held in a pipeline while a decision is made as to whether the tuple satisfies the output criterion.

SUMMARY

This paper has discussed the terminology of the relational data model and traced its development in terms of normalization theory, language design, and implementation techniques. We have discussed the advantages of n-ary relations for data-base management, including simplicity, data independence, symmetry, and a strong theoretical foundation.

In order to be considered a true relational system, a data-base system must possess at least the following attributes:

1) All information is represented by data values. No essential information is contained in invisible connections among records.
2) At the user interface, no particular access path is "preferred" over any other.
3) The user interface is independent of the means by which data is physically stored.

In [M14], E. F. Codd summarized the areas in which further research is most needed in relational data-base management. The following areas were included:

1) Development of concurrency control techniques specifically geared to the relational model.
2) Measuring the performance attainable when the relational approach is applied to a large-scale data base.
3) Development of the theory whereby multiple alternative views of shared data may be supported for retrieval and update.
4) Demonstration of the viability of natural language query formulation subsystems.

ACKNOWLEDGMENTS

The author is indebted to E. F. Codd of IBM for his helpful comments during the preparation of this paper. The bibliography which follows is based on a bibliography compiled by E. F. Codd.

The author is also grateful to his colleagues at the IBM Research Laboratory in San Jose for their support and discussions.

CLASSIFICATION OF REFERENCES

Models and Theory
- M 1) General
- N 2) Normalization, Decomposition, and Synthesis
- Z 3) Relationships between CODASYL DDL/DBTG and the Relational Model
- L Languages and Human Factors
- Implementations
- S 1) Software
- H 2) Hardware
- T Implementation Technology
- C Authorization, Views, and Concurrency
- I Integrity Control
- A Applications
- D Deductive Inference and Approximate Reasoning
- E Natural Language Support
- Y Sets and Relations (prior to 1969)

Certain references include asterisks with the following meaning:
- * Proceedings of ACM-SIGFIDET and ACM-SIGMOD Workshops are obtainable from ACM Headquarters, 1133 Avenue of the Americas, New York, N.Y. 10036
- ** Proceedings of the 1975 ACM Pacific Conference, San Francisco, April 17–18, 1975 are obtainable from: Mail Room, Boole & Babbage, 850 Stewart Drive, Sunnyvale, California 94086

Models and Theory

1) General

[M1] CODD, E. F. "Derivability, redundancy and consistency of relations stored in large data banks," IBM Research Report RJ599, August 1969.

[M2] CODD, E. F. "A relational model of data for large shared data banks," *Comm. ACM* **13**, 6 (June 1970), pp 377–397.

[M3] CODD, E. F. "Relational completeness of data-base sublanguages", Courant Computer Science Symposia *6*, "Data Base Systems," New York, May 1971, Prentice-Hall, Englewood Cliffs, N.J., 1971, pp. 65–98.

[M4] STRNAD, A. L. "The relational approach to the management of data bases," *Proc. IFIP Congress*, August 1971, Vol. 2, North-Holland Publ. Co., Amsterdam, The Netherlands, 1971, pp. 901–904.

[M5] DURCHHOLZ, R. "Das Datenmodell bei Codd," Technical Report No. 69, Gesellschaft für Mathematik und Datenverarbeitung, Bonn, W. Germany, July 1972.

[M6] HAWRYSZKIEWYCZ, I. T.; AND DENNIS, J. B. "An approach to proving the correctness of data-base operations," *Proc. ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov.–Dec. 1972,* ACM, New York, 1972, pp. 323–348.

[M7] DATE, C. J. "Relational data base systems: a tutorial," *Proc. Fourth Internatl. Symposium on Computer and Information Sciences*, Dec. 1972, Plenum Press, New York, 1972.

[M8] CODD, E. F. "Understanding relations," continuing series of articles published in *FDT*, the quarterly bulletin of *ACM-SIGMOD*, beginning with Vol. **5**, 1 (June 1973),* ACM, New York, 1973.

[M9] HAWRYSZKIEWYCZ, I. T. "Semantics of data base systems," MIT Project MAC Report MAC TR-112, Cambridge, Mass., Dec. 1973.

[M10] BRACCHI, G.; FEDELI, A.; AND PAOLINI, P. "A multi-level relational model for data-base management systems," *Data Base Management, Proc. IFIP TC-2 Working Conf. on Data-Base Management Systems*, April 1974, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974.

[M11] STONEBRAKER, M. "A functional view of data independence," *Proc. ACM-SIGFIDET Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp. 63–81.

[M12] MELTZER, H. S. "Relations and relational operations," IBM Report to GUIDE 38 Information Systems Division, Dallas, Texas, May 1974.

[M13] HITCHCOCK, P. "Fundamental operations on relations in a relational data base," IBM Scientific Centre Report UKSC 0051, Peterlee, England, May 1974.

[M14] CODD, E. F. "Recent investigations in relational data base systems," *Information Processing 74, Proc. IFIP Congress*, August 1974, Vol. 5, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974, pp. 1017–1021.

[M15] WEDEKIND, H. "Datenbanksysteme 1," *Reihe Informatik/16* (1974), Bibliographisches Institut, Mannheim, W. Germany.

[M16] HALL, P. A. V.; TODD, S. J. P.; AND HITCHCOCK, P. "An algebra of relations for machine computation," IBM Scientific Centre Report UKSC 0066, Peterlee, England, Jan. 1975.

[M17] SCHMID, H. A.; AND SWENSON, J. R. "On the semantics of the relational data model," *Proc. ACM-SIGMOD Conf.*, May 1975,* ACM, New York, 1975, pp 211–223.

Models and Theory

2) Normalization, Decomposition, and Synthesis

[N1] CODD, E. F. "Further normalization of the data base relational model," Courant Computer Science Symposia *6*, "Data Base Systems," New York, May 1971, Prentice-Hall, New York, 1971, pp. 33–64.

[N2] CODD, E. F. "Normalized data base structure: a brief tutorial," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 1971,* ACM, New York, 1971, pp. 1–17.

[N3] HEATH, I. J. "Unacceptable file operations in a relational data base," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 1971,* ACM, New York, 1971, pp. 19–33.

[N4] DELOBEL, C. "Aspects theoretiques sur la structure de l'information dans une base de données", *Revue Francaise d'Informatique et de Recherche Operationelle*, **B-3** (Sept. 1971).

[N5] DELOBEL, C. "A theory about data in an information system," IBM Research Report, RJ964, San Jose, Calif., Jan. 1972.

[N6] RISSANEN, J.; AND DELOBEL, C. "Decomposition of files, a basis for data storage and retrieval," IBM Research Report RJ1220, San Jose, Calif., May 1973.

[N7] DELOBEL, C.; AND CASEY, R. G. "Decomposition of a data base and the theory of Boolean switching functions," *IBM J. R. & D.* **17**, 5 (Sept. 1973), pp. 374–387.

[N8] KENT, W. "A primer of normal forms," IBM Technical Report TR 02.600, San Jose, Calif., Dec. 1973.

[N9] ARMSTRONG, W. W. "Dependency structures of data base relationships," *Information Processing 74, Proc. IFIP Congress*, August 1974, Vol. 3, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974, pp. 580–584.

[N10] DELOBEL, C.; AND LEONARD, M. "The decomposition process in a relational model," Technical Report, Laboratoire d'Informatique, Univ. of Grenoble, France, Sept. 1974.

[N11] WANG, C. P.; AND WEDEKIND, H. "Segment synthesis in logical data base design," *IBM J. R. & D.* **19**, 1 (Jan. 1975) pp 71–77.

[N12] BERNSTEIN, P. A.; SWENSON, J. R.; AND TSICHRITZIS, D. "A unified approach to functional dependencies and relations," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975, pp. 237–245.

[N13] FADOUS, R. Y.; AND FORSYTH, J. "Finding candidate keys for relational data bases," *Proc. ACM-SIGMOD Conf.*, May 1975,* ACM, New York, 1975, pp. 203–210.

[N14] FADOUS, R. Y. "Mathematical foundations for relational data bases," PhD. Thesis, Michigan State Univ., Lansing, 1975.

[N15] SHARMAN, G. C. H. "A new model of relational data base and high level languages," Technical Report TR. 12.136, IBM Hursley Park Laboratory, England, Feb., 1975.

Models and Theory

3) Relationships between CODASYL DDL/DBTG and Relational Model

[Z1] CODASYL Data Base Task Group Report, April 1971, ACM, New York.

[Z2] CANNING, R. G. "Problem areas in data management," *EDP Analyzer* **12**, 3 (March 1974).

[Z3] EARNEST, C. P. "A comparison of the network and relational data structure models," Technical Report, Computer Sciences Corp., El Segundo, Calif., April 1974.

[Z4] NIJSSEN, G. M. "Data structuring in DDL and relational data model," *Proc. IFIP TC-2 Working Conf. on Data Base Management Systems*, April 1974, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974.

[Z5] CODD, E. F.; AND DATE, C. J. "Interactive support for non-programmers: the relational and network approaches," *Proc. 1974 ACM-SIGMOD Debate "Data Models: Data Structure Set versus Relational,"* May 1974,* ACM, New York, 1974.

[Z6] DATE, C. J.; AND CODD, E. F. "The relational and network approaches: comparison of the application programming interfaces," *Proc. 1974 ACM-SIGMOD Debate "Data Models: Data Structure Set versus Relational,"* May, 1974,* ACM, New York, 1974.

[Z7] BACHMAN, C. W. "The data structure set model," *Proc. 1974 ACM-SIGMOD Debate "Data Models: Data Structure Set versus Relational,"* May 1974,* ACM, New York, 1974.

[Z8] SIBLEY, E. H. "On the equivalences of data based systems," *Proc. ACM-SIGMOD Debate "Data Models: Data Structure Set versus Relational,"* May 1974,* ACM, New York, 1974.

[Z9] EVEREST, G. C. "The futures of database management," *Proc. ACM-SIGMOD Workshop on Data Description, Access, and Control*, May, 1974, ACM, New York, 1974, pp. 445–462.

[Z10] OLLE, T. W. "Current and future trends in data base management systems," *Information Processing 74, Proc. IFIP Congress*, August, 1974. Vol. 5, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974, pp 998–1006.

[Z11] DATE, C. J. "*An introduction to data base systems*," Addison-Wesley, Reading, Mass., 1975.

[Z12] KAY, M. H. "An assessment of the CODASYL DDL for use with a relational schema," *Data Base Description*, B. C. M. Douque and G. M. Nijssen (Eds.), North-Holland Publ. Co., Amsterdam, The Netherlands, 1975, pp. 199–214.

[Z13] ROBINSON, K. A. "An analysis of the uses of the CODASYL set concept," *Data Base Description*, B. C. M. Douque and G. M. Nijssen, (Eds.), North-Holland Publ. Co., Amsterdam, The Netherlands, 1975, pp. 169–182.

[Z14] TAYLOR, R. W. "Observations on the attributes of database sets," *Data Base Description*, B. C. M. Douque and G. M. Nijssen (Eds.), North-Holland Publ. Co., Amsterdam, The Netherlands, 1975, pp. 73–84.

[Z15] OLLE, T. W. "An analysis of shortcomings in the schema DDL with an outline of proposed improvements," *Data Base Description*, B. C. M. Douque and G. M. Nijssen (Eds.), North-Holland Publ. Co., Amsterdam, The Netherlands, 1975, pp. 283–298.

[Z16] HUITS, M. "Requirements for languages in data-base systems," *Data Base Description*, B. C. M. Douque and G. M. Nijssen (Eds.), North-Holland Publ. Co., Amsterdam, The Netherlands, 1975, pp. 85–110.

[Z17] ROBINSON, K. A. "Data base—the ideas behind the ideas," *Computer J.* **18**, 1 (Jan. 1975), pp. 7–12.

[Z18] HELD, G.; AND STONEBRAKER, M. "Networks, hierarchies, and relations in data base management systems," *Proc. ACM Pacific 75 Regional Conf.*, April, 1975,** ACM, New York, 1975, pp. 1–9.

[Z19] MARTIN, J. T. "*Computer data-base organization*," Prentice-Hall, Englewood Cliffs, N.J., 1975.

Languages and Human Factors

[L1] CODD, E. F. "A data base sublanguage founded on the relational calculus," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 1971,* ACM, New York, 1971, pp. 35–68.

[L2] CODD, E. F. "Relational algebra," Courant Computer Science Symposia 6, "Data Base Systems," New York, May 1971, Prentice-Hall, New York, 1971.

[L3] BRACCHI, G.; FEDELI, A.; AND PAOLINI, P. "A language for a relational data base," *Sixth Annual Princeton Conf. on Information Sciences and Systems*, March 1972, Princeton Univ., N.J., 1972.

[L4] BJORNER, D.; CODD, E. F.; DECKERT, K. L.; AND TRAIGER, I. L. "The GAMMA ZERO n-ary relational data base interface: specifications of objects and operations," IBM Research Report RJ1200, San Jose, Calif., April, 1973.

[L5] EARLEY, J. "Relational level data structures for programming languages," *Acta Informatica*, **2**, 4 (1973), pp. 293–309.

[L6] DEE, E.; HILDER, W.; KING, P. J. H.; AND TAYLOR, E. "COBOL extensions to handle a relational data base," British Computer Society, Working Party #5, Oct. 1973.

[L7] FEHDER, P. L. "The representation-independent language," IBM Research Reports RJ1121 & RJ1251, San Jose, Calif., Nov. 1972 & July 1973 respectively.

[L8] BOYCE, R. F.; AND CHAMBERLIN, D. D. "Using a structured English query language as a data definition facility," IBM Research Report RJ1318, San Jose, Calif., Dec. 1973.

[L9] BOYCE, R. F.; CHAMBERLIN, D. D.; KING, W. F., III; AND HAMMER, M. M. "Specifying queries as relational expressions: SQUARE," *Data Base Management, Proc. IFIP Working Conf.*, April 1974, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974, pp 169–177.

[L10] CHAMBERLIN, D. D.; AND BOYCE, R. F. "SEQUEL: A structured English query language," *Proc. ACM-SIGMOD Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp. 249–264.

[L11] JERVIS, B. "Query languages for relational data-base management systems," Masters Thesis, Univ. of British Columbia, Vancouver, B.C., May 1974.

[L12] COPELAND, G. P.; AND SU, S. Y. W. "A high level data sublanguage for a context-addressed segment-sequential memory," *Proc. ACM-SIGMOD Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp. 265–276.

[L13] ZLOOF, M. M. "Query by example," Research Report RC4917, IBM T. J. Watson Research Center, Yorktown Heights, N. Y., July 1974.

[L14] HALSTEAD, M. H. "Software physics comparison of a sample program in DSL ALPHA and COBOL," IBM Research Report RJ1460, San Jose, Calif., Oct. 1974.

[L15] PIROTTE, A.; AND WODON, P. "A comprehensive formal query language for a relational data base: FQL," Technical Report R283, M.B.L.E. Laboratoire de Recherches, Brussels, Belgium, Dec. 1974.

[L16] SUMMERS, R. C.; COLEMAN, C. D.; AND FERNANDEZ, E. B. "A programming language extension for access to a shared data base," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp 114–118.

[L17] MCDONALD, N.; AND STONEBRAKER, M. "CUPID: the friendly query language," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp 127–131.

[L18] WESTGAARD, R. E. "A COBOL data base facility for the relational data model," *Proc. ACM Pacific 75 Regional Conf.*, April, 1975,** ACM, New York, 1975, pp 132–139.

[L19] SHU, N. C.; HOUSEL, B. C.; AND LUM, V. Y. "CONVERT: a high level translation definition language for data conversion," *Proc. ACM-SIGMOD Conf.*, May, 1975,* ACM, New York, 1975, p 3. (*Comm. ACM*) **18**, 10 (Oct. 75) 557–567.

[L20] REISNER, P.; BOYCE, R. F.; AND CHAMBERLIN, D. D. "Human factors evaluation of two data base query languages: SQUARE and SEQUEL," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 447–452.

[L21] ZLOOF, M. M. "Query by Example," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 431–438.

[L22] THOMAS, J. C.; AND GOULD, J. D. "A psychological study of Query by Example," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., pp 439–445.

[L23] HALSTEAD, M. H. "Software physics: basic principles," Research Report RJ1582, IBM Research Laboratory, San Jose, Calif., May 1975.

[L24] ZLOOF, M. M. "Query by Example: the invocation and definition of tables and forms," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 1–24.

[L25] BOYCE, R. F.; CHAMBERLIN, D. D.; KING, W. F.; AND HAMMER, M. M. "Specifying queries as relational expressions: the SQUARE data sublanguage," *Comm. ACM* **18**, 11 (Nov. 1975), pp. 621–628.

Implementations

1) Software

[S1] GOLDSTEIN, R. C.; AND STENAD, A. L. "The MACAIMS data management system," *Proc. 1970 ACM-SIGFIDET Workshop on Data Description and Access*, Nov. 1970,* ACM, New York, 1970, pp. 201–229.

[S2] SYMONDS, A. J.; AND LORIE, R. A. "A schema for describing a relational data base," *Proc. ACM-SIGFIDET Workshop on Data Description and Access*, Nov. 1970,* ACM, New York, 1970, pp. 230–245.

[S3] LORIE, R. A.; AND SYMONDS, A. J. "A relational access method for interactive applications," Courant Computer Science Symposia, *6, Data Base Systems*, Prentice-Hall, New York, 1971, pp 99–124.

[S4] NOTLEY, M. G. "The Peterlee IS/1 system," IBM UK Scientific Centre Report UKSC-0018, March 1972.

[S5] WHITNEY, V. K. M. "RDMS: a relational data management system," *Proc. Fourth Internatl. Symposium on Computer and Information Sciences* (COINS IV), Dec. 1972, Plenum Press, New York, 1972.

[S6] LORIE, R. A. "XRM—an extended (n-ary) relational memory," IBM Scientific Center Report G320-2096, Cambridge, Mass., Jan. 1974.

[S7] MCDONALD, N.; STONEBRAKER, M.; AND WONG, E., "Preliminary design of INGRES: Part I," Electronics Research Lab. Report ERL-M435, Univ. of California, Berkeley, April 1974.

[S8] CZARNIK, B.; SCHUSTER, S.; AND TSICHRITZIS, D. "ZETA: a relational data base management system," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp. 21–25.

[S9] HELD, G.; AND STONEBRAKER, M. "Storage structures and access methods in the relational data base management system INGRES," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp 26–33.

[S10] ASTRAHAN, M. M.; AND LORIE, R. A. "SEQUEL-XRM: a relational system," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp 34–38.

[S11] ASTRAHAN, M. M.; AND CHAMBERLIN, D. D. "Implementation of a structured English query language," *Comm. ACM* **18**, 10 (Oct. 1975), pp 580–588.

[S12] STEWERT, J.; AND GOLDMAN, J. "The relational data management system: a perspective," *Proc. ACM-SIGMOD Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp 295–320.

[S13] MCLEOD, D. J.; AND MELDMAN, M. J. "RISS: a generalized minicomputer relational data base management system," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 397–402.

[S14] MYLOPOULOS, J.; SCHUSTER, S. A.; AND TSICHRITZIS, D. "A multi-level relational system," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp. 403–408.

[S15] HELD, G. D.; STONEBRAKER, M. R.; AND WONG, E. "INGRES: a relational data base system," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 409–416.

[S16] TODD, S. J. P. "Peterlee relational test vehicle PRTV, a technical overview," IBM Scientific Centre Report UKSC 0075, Peterlee, England, July 1975.

[S17] WINSLOW, L. E. "An efficient implementation of Codd's relational model data base," *Proc. COMPCON 75, 11th Annual IEEE Computer Society Conf.*, Sept. 1975, IEEE, New York, 1975.

[S18] MANACHER, G. K. "On the feasibility of implementing a large relational data base with optimal performance on a mini-computer," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 175–201.

[S19] CODD, E. F. (Ed.), "Implementation of relational data base management systems," *FDT, Quarterly Bulletin of ACM-SIGMOD* **7**, 3–4 (1975).

[S20] ASTRAHAN, M. M. et. al., "System R: a relational approach to data-base management," Research Report RJ 1738, IBM Research Laboratory, San Jose, Calif. Feb. 1976.

Implementations

2) Hardware

[H1] SU, S. Y. W.; COPELAND, G. P.; AND LIPOVSKI, G. J. "Retrieval operations and data representations in a context-addressed disk sytsem," *Proc. ACM-SIGPLAN-SIGIR Interface Meeting on Programming Languages and Information Retrieval*, Nov. 1973, ACM, New York, 1973.

[H2] COPELAND, G. P.; LIPOVSKI, G. J.; AND SU, S. Y. W. "The architecture of CASSM: a cellular system for non-numeric processing," *Proc. First Annual Symposium on Computer Architecture*, Dec. 1973, IEEE, N.Y., 1973.

[H3] OZKARAHAN, E. A.; SCHUSTER, S. A.; AND SMITH, K. C. "RAP: an associative processor for data base management," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 379–387.

[H4] LIN, C. S.; AND SMITH, D. C. P. "The design of a rotating associative array memory for a relational data-base management application," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp. 453–454.

[H5] SU, S. Y. W.; AND LIPOVSKI, G. J. "CASSM: a cellular system for very large data bases," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 456–472.

Implementation Technology

[T1] BAYER, R.; AND MCCREIGHT, E. "Organization and maintenance of large ordered indices," *Proc. ACM-SIGFIDET Workshop*, Nov. 1970, ACM, New York, 1970 pp 107–141.

[T2] "IBM system/360 operating system data management services," IBM Publication No. GC26-3746, 1971.

[T3] DATE, C. J.; AND HOPEWELL, P. "File definition and logical data independence," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 1971,* ACM, New York, 1971, pp 117–138.

[T4] DATE, C. J.; AND HOPEWELL, P. "Storage structure and physical data independence," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 1971,* ACM, New York, 1971, pp 139–168.

[T5] ROTHNIE, J. B. "The design of generalized data management systems," PhD Thesis, MIT, Cambridge, Mass., Sept. 1972.

[T6] PALERMO, F. P. "A data base search problem," *Fourth Internatl. Symposium on Computer and Information Science* (COINS IV), Dec. 1972, Plenum Press, New York, 1972.

[T7] HALL, P. A. V.; AND TODD, S. J. P. "Factorisations of algebraic expressions," IBM Scientific Centre Report UKSC 0055, Peterlee, England, April 1974.

[T8] ROTHNIE, J. B. "An approach to implementing a relational data management system," *Proc. ACM-SIGMOD Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp 277–294.

[T9] WHITNEY, V. K. M. "Relational data management implementation techniques," *Proc. ACM-SIGFIDET Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp. 321–350.

[T10] CASEY, R. G.; AND OSMAN, I. "Generalized page replacement algorithms in a relational data base," *Proc. ACM-SIGFIDET Workshop on Data Description, Access, and Control*, May 1974,* ACM, New York, 1974, pp 101–124.

[T11] HALL, P. A. V. "Common sub-expression identification in general algebraic systems," IBM Scientific Centre Report UKSC 0060, Peterlee, England, Nov. 1974.

[T12] TSICHRITZIS, D. "A network framework for relation implementation," *Proc. IFIP TC-2 Special Working Conf. on the DDL*, Jan. 1975, published as *Data Base Description*, North-Holland Publ. Co., Amsterdam, The Netherlands, 1975, pp. 269–282.

[T13] GRAY, J. N.; AND WATSON, V. "A shared segment and inter-process communication facility for VM/370," Research Report RJ1579, IBM Research Laboratory, San Jose, Calif., Feb. 1975.

[T14] CHIBA, Y. "A data base search algorithm based on complicated retrieval conditions," *The Soken Kiyo* **5**, 1 (1975), 159–175.

[T15] FARLEY, J. H. GILLES; AND SCHUSTER, S. A. "Query execution and index selection for relational data bases," Technical Report CSRG-53, Computer Systems Research Group, Univ. of Toronto, Toronto, Ont., Canada, March 1975.

[T16] PECHERER, R. M. "Efficient evaluation of expressions in a relational algebra," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp 44–49.

[T17] GOTLIEB, L. R. "Computing joins of relations," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975.

[T18] SMITH, J. M.; AND CHANG, P. "Optimizing the performance of a relational algebra data base interface," *Comm. ACM* **18**, 10 (Oct. 1975), pp 568–579.

[T19] SCHKOLNICK, M. "Secondary index optimization," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975, pp 186–192.

[T20] ROTHNIE, J. B. "Evaluating interentry retrieval expressions in a relational data base management system," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp. 417–423.

[T21] PALERMO, F. P. "An APL environment for testing relational operators and data base search algorithms," *Proc. APL 75 Conf.*, June 1975, ACM, New York, 1975, pp 249–256.

[T22] HALL, P. A. V. "Optimisation of a single relational expression in a relational data base system," IBM Scientific Centre Report UKSC 0076, Peterlee, England, July 1975.

[T23] SCHMID, H. A.; AND BERNSTEIN, P. A. "A multi-level architecture for relational data base systems," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 202–226.

[T24] LIEN, Y. E.; TAYLOR, C. E.; REYNOLDS, M. L.; AND DRISCOLL, J. R. "Binary search tree complex—a realization of a relational database management system," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 540–542.

Authorization, Views and Concurrency

[C1] CHAMBERLIN, D. D.; BOYCE, R. F.; TRAIGER, I. L. "A deadlock-free scheme for resource locking in a data base environment," *Information Processing 74, Proc. IFIP Congress*, August 1974, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974, pp. 340–343.

[C2] OWENS, R. C. "Evaluation of access authorization characteristics of derived data sets," *Proc. ACM-SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 1971,* ACM, New York, 1971, pp 263–278.

[C3] STONEBRAKER, M.; AND WONG, E. "Access control in a relational data base management system by query modification," Electronics Research Lab.,

Report ERL-M438, Univ. of Calif., Berkeley, May 1974.

[C4] 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 RJ1487, San Jose, Calif., Dec. 1974.

[C5] FERNANDEZ, E. B.; SUMMERS, R. C.; AND COLEMAN, C. D. "An authorization model for a shared data base," *Proc. ACM-SIGMOD Conf.*, May 1975,* ACM, New York, 1975, pp 23–31.

[C6] CHAMBERLIN, D. D.; GRAY, J. N.; AND TRAIGER, I. L. "Views, authorization, and locking in a relational data base system," *Proc. AFIPS National Computer Conf.*, May 1975, Vol. 44, AFIPS Press, Montvale, N.J., 1975, pp 425–430.

[C7] WEE, R. S. S. "Problems in the dynamic sharing of data in a relational data base environment," IBM Scientific Centre Report UKSC 0067, Peterlee, England, August 1975.

[C8] GRAY, J. N.; LORIE, R. A.; AND PUTZOLU, G. R. "Granularity of locks in a large shared data base," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 428–451.

Integrity Control

[I1] FLORENTIN, J. J. "Consistency auditing of data bases," *Computer J.* 17, 1 (Feb. 1974), pp 52–58.

[I2] STONEBRAKER, M. "High level integrity assurance in relational data base management systems," Electronics Research Lab. Report ERL-M473, Univ. of Calif. at Berkeley, August 1974.

[I3] GRAVES, R. W. "Integrity control in a relational data description language," *Proc. ACM Pacific 75 Regional Conf.*, April 1975,** ACM, New York, 1975, pp 108–113.

[I4] STONEBRAKER, M. "Implementation of integrity constraints and views by query modification," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975, pp 65–78.

[I5] ESWARAN, K. P.; AND CHAMBERLIN, D. D. "Functional specifications of a subsystem for data base integrity," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 48–68.

[I6] HAMMER, M. M.; AND MCLEOD, D. J. "Semantic integrity in a relational data base system" *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 25–47.

Applications

[A1] SOOP, K.; SVENSSON, P.; AND WIKTORIN, L. "An experiment with a relational data base system in environmental research," *Proc. Fourth Internatl. Symposium on Computer and Information Sciences* (COINS IV), Dec. 1972 Plenum Press, New York, 1972.

[A2] KUNII, T. L.; AMANO, T.; ARISAWA, H.; AND OKADA, S. "An interactive fashion design system INFADS," abstract in *Proc. Conf. on Computer Graphics & Interactive Techniques*, July 1974; paper in *Computers & Graphics* 1, (1975), Pergamon Press, New York.

[A3] WILLIAMS, R. "On the application of relational data structures in computer graphics," *Information Processing 74, Proc. IFIP Congress*, August 1974, Vol. 4, North-Holland Publ. Co., Amsterdam, The Netherlands, pp 722–726.

[A4] KUNII, T. L.; WEYL, S.; AND TENENBAUM, J. M. "A relational data base schema for describing complex pictures with color and texture," *Proc. Second Jt. Conf. on Pattern Recognition*, August 1974, IEEE Cat. No. 74CH0885-4C, IEEE, New York, 1974.

[A5] VALLE, G. "Interactive handling of data base relations: experiments with the relational approach," Technical Report, Univ. of Bologna, Bologna, Italy, March 1975.

[A6] DEJONG, S. P.; AND ZLOOF, M. M. "The system for business automation (SBA): programming language," IBM Research Report RC5302, Yorktown Heights, N.Y., March 1975.

[A7] DEJONG, S. P.; AND ZLOOF, M. M. "Application design within the system for business automation", IBM Research Report RC5366, Yorktown Heights, N.Y., April 1975.

[A8] NAVATHE, S. B.; AND MERTEN, A. G. "Investigations into the application of the relational model to data translation," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975, pp 123–138.

[A9] BANDURSKI, A. E.; AND JEFFERSON, D. K. "Data description for computer-aided design," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975 pp 193–202.

[A10] WILLIAMS, R.; AND GIDDINGS, G. M. "A picture building system," *Proc. Conf. on Computer Graphics, Pattern Recognition, & Data Structure*, May 1975, IEEE Cat. No. 75CH0981-1C, IEEE, New York, 1975.

[A11] GO, A.; STONEBRAKER, M.; AND WILLIAMS, C. "An approach to implementing a geo-data system," *Proc. ACM-SIGDA-SIGMOD-SIGGRAPH Workshop on Data Bases for Interactive Design*, Sept. 1975, ACM, New York, 1975, pp. 67–77.

[A12] DONOVAN, J.; FESSEL, R.; GREENBERG, S.; AND GUTENTAG, L. "An experimental VM/370 based information system," *Proc. Internatl. Conf. on Very Large Data Bases*, Sept. 1975, ACM, New York, 1975, pp 549–553.

Deductive Inference and Approximate Reasoning

The references in this section represent a small sample of the publications in deductive inference. Many additional references will be found in [D1].

[D1] CHANG, C. L.; AND LEE, R. C. T. *Symbolic logic and mechanical theorem proving*, Academic Press, New York, 1973.

[D2] MINKER, J. "Performing inferences over relational data bases," *Proc. ACM-SIGMOD Conf.* May 1975,* ACM, New York, 1975 pp 79–91.

[D3] ZADEH, L. A. "Calculus of fuzzy restrictions," Report ERL-M502, Electronics Research Lab., Univ. of Calif., Berkeley, Calif., Feb. 1975.

Natural Language Support

[E1] SIMMONS, R. F. "Natural language question-answering systems: 1969," *Comm. ACM* **13**, 1 (Jan. 1970), 15–30.

[E2] SCHANK, R. C.; AND COLBY, K. M. (Eds.), *Computer models of thought and language* W. H. Freeman, San Francisco, 1973.

[E3] RUSTIN, RANDALL (Ed.), "Natural language processing," Courant Computer Science Symposia *8*, New York, Dec. 1971, Prentice-Hall, Englewood Cliffs, N.J., 1971.

[E4] THOMPSON, F. P.; LOCKEMANN, P. C.; DOSTERT, B. H.; AND DEVERILL, R. "REL: a rapidly extensible language system," *Proc. 24th ACM National Conf.*, New York, 1969, ACM, New York, 1969, pp 399–417.

[E5] KELLOGG, C. H.; BURGER, J.; DILLER, T.; AND FOGT, K. "The CONVERSE natural language data management system: current status and plans," *Proc. ACM Symposium on Information Storage and Retrieval*, 1971, ACM, New York, 1971, pp 33–46.

[E6] WINOGRAD, T. "Procedures as a representation for data in a computer program for understanding natural language," MIT Project MAC Report MAC TR-84, Cambridge, Mass., 1971.

[E7] MONTGOMERY, C. A. "Is natural language an unnatural query language?" *Proc. ACM National Conf.*, New York, 1972, ACM, New York, 1972, pp 1075–1078.

[E8] PETRICK, S. R. "Semantic interpretation in the REQUEST system," IBM Research Report RC4457, IBM Research Center, Yorktown Heights, New York, July 1973.

[E9] CODD, E. F. "Seven steps to RENDEZVOUS with the casual user," *Proc. IFIP TC-2 Working Conf. on Data Base Management Systems*, April 1974, North-Holland Publ. Co., Amsterdam, The Netherlands, 1974.

Sets and Relations (prior to 1969)

These references are included to enable the reader to trace work published prior to 1969 on computer support for (mathematical) sets and relations.

[Y1] CODASYL Development Committee, "An information algebra", *Comm. ACM* **5**, 4 (April 1962), 190–204.

[Y2] LEVIEN, R. E.; AND MARON, M. E. "A computer system for inference execution and data retrieval," *Comm. ACM* **10**, 11 (Nov. 1967), 715–721.

[Y3] CHILDS, D. L. "Feasibility of a set-theoretical data structure—a general structure based on a reconstituted definition of relation," *Proc. IFIP Congress 1968*, North-Holland Publ. Co., Amsterdam, The Netherlands, 1968, pp 162–172.

[Y4] CHILDS, D. L. "Description of a set-theoretic structure," *Proc. AFIPS 1968 Fall Jt. Computer Conf.*, Vol. 33, AFIPS Press, Montvale, N.J., 1968, pp 557–564.

[Y5] ASH, W. L.; AND SIBLEY, E. H. "TRAMP: A relational memory with deductive capabilities," *Proc. ACM 23rd National Conf.*, August 1968, Brandon/Systems Press, Princeton, N.J., 1968, pp 143–156.

[Y6] FELDMAN, J. A.; AND ROVNER, P. D. "An ALGOL-based associative language," *Comm. ACM* **12**, 8 (August 1969), 439–449.

[Y7] KUHNS, J. L. "Logical aspects of question answering by computer," *Proc. Third Internatl. Symposium on Computer and Information Sciences* (COINS IV), Dec. 1969, Academic Press, New York, 1969.

[Y8] KOCHEN, M. "Adaptive mechanisms in digital concept processing," in *Discrete Adaptive Processes—Symposium and Panel Discussion*, AIEE, New York, 1962 pp 50–58.