AD-A034 994 STANFORD UNIV CALIF DIGITAL SYSTEMS LAB F/G 9/2

A CLOSED SYSTEM OF REPRESENTATION FOR RELATIONALLY-ORGANIZED DA--ETC(U)

DEC 74 C R HOLLANDER N00014-67-A-0112-0044

UNCLASSIFIED TN-53 NL

1 OF 1

AD A034994

END

DATE FILMED

3-77

A CLOSED SYSTEM OF REPRESENTATION FOR RELATIONALLY-ORGANIZED DATA AND ITS DESCRIPTORS

Clifford R. Hollander

Technical Note No. 53

December 1974

Digital Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California

This work was supported in part by the Joint Services Electronics Programs, U.S. Army, U.S. Navy, and U.S. Air Force under Contract N-00014-67-A-0112-0044.

A CLOSED SYSTEM OF REPRESENTATION FOR RELATIONALLY-ORGANIZED DATA AND ITS DESCRIPTORS

Clifford R. Hollander

Technical Note. no 53

Digital Systems Laboratory
Stanford University
Stanford, California 94305

December 1974

Abstract

Recently we have witnessed a dramatic increase in both the study and use of data bases. These activities have in turn stimulated interest in data description facilities. The work reported here was motivated by the observation that descriptors for structured data are themselves, in fact, "data-like" in many respects. This paper introduces a simple language for relationally-organized data and a companion descriptor language, for the purpose of demonstrating a single system of representation for both data constructs and their descriptors. It is assumed that each data construct (i.e., relation) belongs to a previously defined relation class and that a descriptor is used to define the structure of the elements of each class. The system presented here is "closed" in the sense that it allows each descriptor to be represented as a relation. Its adoption would permit data description facilities to be implemented in terms of data manipulation facilities.

Key words and Phrases: Data description, relation, descriptor, relation class, closed descriptive system.

Introduction

Non-procedural data description languages (DDLs) have emerged in conjunction with more flexible schemes for organizing data. Some DDLs (e.g., [1,2]) were tied to a particular procedural host language (e.g., COBOL). Others (e.g., [3,4]) supported only specific classes of data structures. The most general DDLs (e.g., [5,6]) were host language independent and/or embraced a wide spectrum of data organizations (e.g., hierarchical, relational, network). In each case, however, the facilities for constructing data descriptors were kept separate from those for handling the data itself.

The work reported here is an adaptation of methods applied earlier to hierarchical data structures [7]. It was motivated by the observation that descriptors for structured data are themselves, in fact, "data-like" in many respects. We introduce a simple data language and a companion descriptor language, for the purpose of demonstrating a single closed system of representation for both data constructs (i.e., relations) and their descriptors. It is assumed that each relation belongs to a previously defined relation class and that a descriptor is used to define the structure of the elements of each class.

The system is "closed" in the sense that it allows each descriptor to be represented in terms of a relation whose structure is, in turn, specified by means of a second-level descriptor. Closure is achieved because a single relation class is found whose own structure is sufficiently general to accomodate the data representation of any descriptor, including its own. Transformations are defined, relating the descriptor and data representations for a descriptor.

An obvious benefit could be derived from a scheme such as the one proposed here. Its implementation would permit a single set of data manipulation facilities (e.g., [4,5]) to suffice both for building elements of existing relation classes and for defining new relation classes themselves.

A Relational Data-Descriptor Language Pair

We require a framework within which to discuss the relationship between data and the descriptors which serve to define its structure. Let us therefore introduce a language of relational data structures (i.e., relations) and a complementary descriptor language. It is important to remark at this point that this paper is deliberately incomplete in its treatment of relational data bases; we are focusing here upon certain aspects of the description of relationally organized data, whereas we ignore its manipulation. For a more thorough treatment of the theory and potential of the relational approach to data management, the reader is referred to [3,4,8].

We consider a data language consisting of <u>relations</u> defined over (not necessarily distinct) sets (called <u>domains</u>) $D_1, D_2, \ldots, D_n$. To be consistent with [9], we shall require that each D_k contain only simple, non-aggregate values; <u>integers</u> and <u>identifiers</u> would be examples of such domains. A relation R over $D_1, D_2, \ldots, D_{n_R}$ is a subset of the cartesian product $D_1 \times D_2 \times \ldots \times D_{n_R}$. The number of domains involved, n_R, is referred to as the degree of R. Stated another way, R is a set of tuples, each of the form $<d_1, d_2, \ldots, d_{n_R}>$ where $d_i \in D_i$. For convenience (and to distinguish between nondistinct domains), a unique role name, id_k, is associated with each domain, D_k, which underlies R. A role name is used for accessing the corresponding component of any tuple $r \in R$ (i.e., $id_k[r]$ selects the k-th component of r). We shall denote the current set of relations by $\mathfrak{R}$.

We shall define the structure of each relation by means of descriptor. A separate language is used for formulating these descriptors. This language must allow us to specify the domains and role names from which a relation is built. To define a relation $R \in \mathfrak{R}$ we use a descriptor ds(R) which takes the form of a tuple of named domains

$$\underline{ds}(R) = <id_1 : D_1, id_2 : D_2, \ldots, id_{n_R} : D_{n_R}> \quad (1)$$

where each D_k is a domain and the corresponding id_k is its role name relative to R. We let $\mathfrak{D}$ denote the current set of descriptors. It is important to distinguish between $\mathfrak{D}$ and the set of all possible descriptors; while the latter depends only upon the descriptor language, the former depends upon $\mathfrak{R}$ as well. We can express the assumed relationship between data (relations) and descriptors in terms of a mapping

$$\text{DESCRIBED\_BY} : \mathfrak{R} \rightarrow \mathfrak{D} \quad (2)$$

where as we have observed above $\mathfrak{R}$ and $\mathfrak{D}$ vary with time. Each descriptor D actually defines the structure of set of relations, called a relation class. Formally speaking, the relation class is the inverse image of D under DESCRIBED_BY, i.e.

$$\text{DESCRIBED\_BY}^{-1} : \mathfrak{D} \rightarrow \{\text{relation classes}\}. \qquad (3)$$

Fig. 1 displays an example relation, EMP, and the corresponding descriptor ds(EMP); EMP might be used to hold employee information. EMP is defined over four nondistinct domains: two instances each of a domain name and a domain integer. For convenience EMP is displayed using a tabular format with one column, labelled by a role name, for each domain (instance). It should be noted that the order of appearance of the rows (i.e., tuples) in the table is irrelevant.

A Data Representation for Descriptors

Within the framework established in the previous section, any extension to $\mathfrak{R}$ to define a new data relation class requires that a new descriptor D be created and included in $\mathfrak{D}$. We discuss here a transformation technique whereby any descriptor can be represented as a relation. The importance of this technique is that it facilitates extensions to $\mathfrak{R}$ by reducing the problem of creating a new descriptor to one of building an element of a previously defined relation class. We let rn(D) denote the relation by which we represent D.

For a relation to be used to represent descriptor information, its own structure must be specified by a descriptor. This would seem to lead, unfortunately, to a system requiring an infinite number of descriptor levels, (i.e., a relation R, described by a descriptor ds(R), represented as a relation rn(ds(R)), described by a descriptor ds(rn(ds(R))), ...). However, this situation can be avoided if a single relation class can be found to accomodate the data representation of any descriptor, including the one that defines the structure of (each element of) the relation class itself.

We recall from eq. (1) that a descriptor D is an n-tuple of identifier-domain pairs. We can represent each element of D by means of a 3-tuple of the form

$$\langle k,\ id_k,\ domain_k \rangle \qquad (4)$$

where k indexes a position within D. Having made this observation, let us now define a relation, denoted by rn(D) whose tuples are given by eq. (3) for k = 1,2,...,n. For example the result of applying this technique to the descriptor ds(EMP) in Fig. 1 is

rn(ds(EMP)) = (5)

INDEX	ROLE	DOM
1	NAME	name
2	SAL	integer
3	MGR	name
4	CODE	integer

We note that for any descriptor D, rn(D) has degree 3 and is defined over domains integer,† identifier,† and domain;* as in eq. (5), we shall use INDEX, ROLE, and DOM as their respective role names. The descriptor for rn(D), namely ds(rn(D)), is given by

<INDEX:integer, ROLE:identifier, DOM:domain> (6)

---

† Note that both of these domains serve as candidate keys [8] for such a relation.

* This use of a domain-valued domain (i.e., a domain whose elements are domain names) is very much akin to the use of mode-valued modes in programming languages such as ALGOL 68 [10].

EMP:

NAME	SAL	MGR	CODE
J. Smith	10	W. Harris	7
F. Mills	21	A. Kelly	13
L. Tan	11	F. Mills	9
W. Harris	19	A. Kelly	14
P. Jones	12	W. Harris	9
R. Simms	13	F. Mills	8

(a) Tabular Form of EMP

ds(EMP) = <NAME:name, SAL:integer, MGR:name, CODE:integer>

(b) Descriptor for EMP

Fig. 1. A Sample Relation, EMP

The important thing to notice about eq. (6) is that it does not depend upon D. So that regardless of the original descriptor D to which this representation scheme is applied, it always yields a relation in the relation class defined by the descriptor in eq. (6). For convenience, let us denote this "descriptor-descriptor" by D_{super}. To demonstrate that we have indeed achieved the desired closure with respect to descriptor levels, we apply the transformation rn(·) to D_{super} yielding

$\underline{rn}(D_{super})$ =

INDEX	ROLE	DOM
1	INDEX	integer
2	ROLE	identifier
3	DOM	domain

(7)

and we observe that D_{super} is a fixedpoint [11] of the composite transformation ds(rn(·)), i.e.,

$$\underline{ds}(\underline{rn}(D_{super})) = D_{super}. \qquad (8)$$

Clearly it is a straightforward matter to define an inverse transformation to rn for recovering the descriptor representation of a descriptor from its data representation. Fig. 2 diagrams the relationships that exist between the various constructs arising from a typical data relation (e.g., EMP).

Conclusions

We have examined a transformation technique by which descriptors can be represented as data relations. Although the technique is demonstrated only for a particular choice of data and descriptor languages, it is clear that

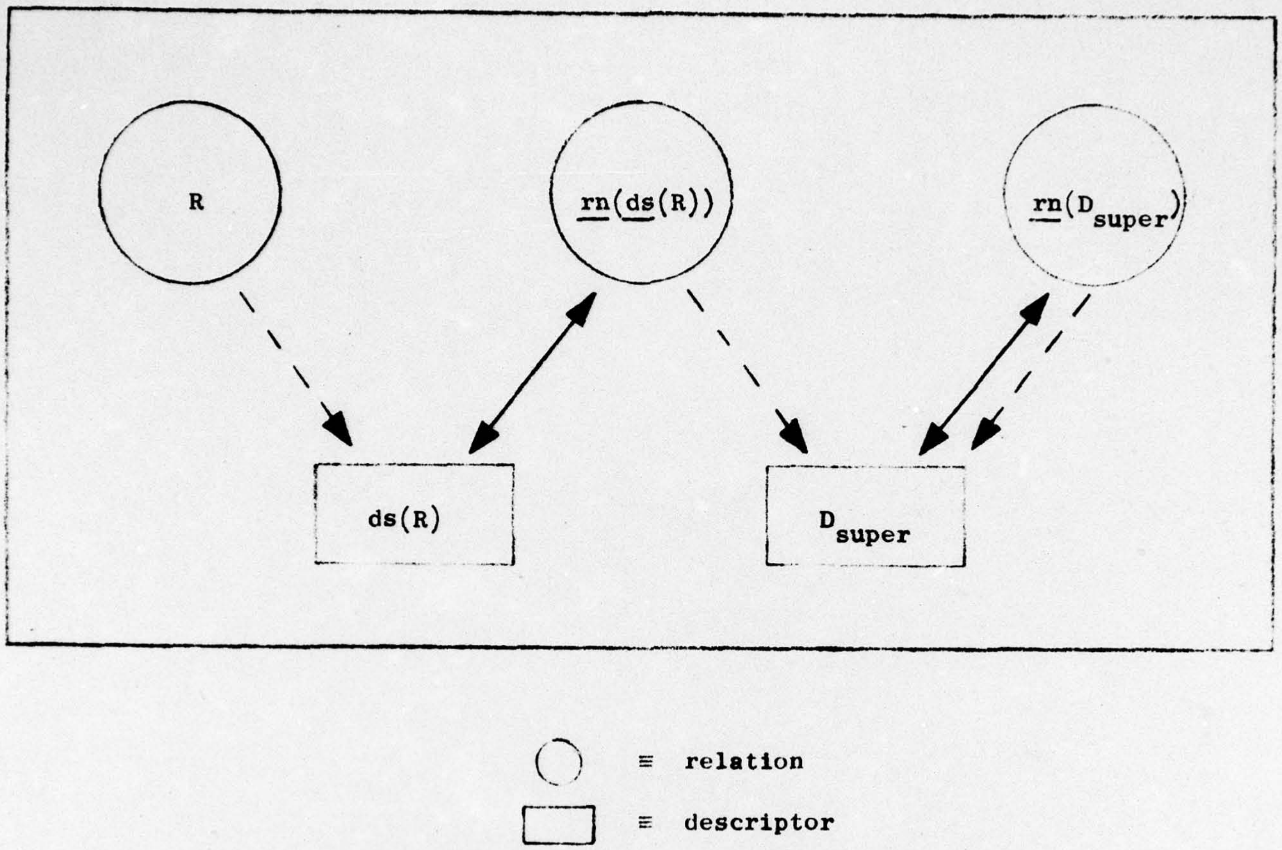

Fig. 2. Descriptive environment of a relation R.

it can be generalized to a broad class of data/descriptor language pairs. It is in fact applicable to any pair in which the data language constructs are sufficiently powerful to encode the variability of structure exhibited by the set of descriptors.

The result of employing such a technique is a single closed system of representation for both data and the descriptors which define the structure of that data. The benefit derived from adopting this kind of system is that it would allow a data description facility to be completely subsumed by (or defined in terms of) a data manipulation facility.

References

[1] J. W. Dempsey and J. K. Mullin, "Problems of building a hybrid data definition facility," Proceedings of the ACM-SICFIDET workshop on data description and access, November 1970, pp. 174-187.

[2] Max E. Ellis and Kenneth H. Nelson, "A data description language for hierarchical data files," Proceedings of the ACM-SICFIDET workshop on data description and access, November 1970, pp. 87-106.

[3] Edgar F. Codd, "A relational model of data for large shared data banks," Communications of the ACM, vol. 13, no. 6, June 1970, pp. 377-387.

[4] Edgar F. Codd, "A data base sublanguage founded on the relational calculus," Proceedings of the ACM-SICFIDET workshop on data description, access and control, November 1971, pp. 35-68.

[5] CODASYL, Data base task group report, New York, April 1971.

[6] Edgar H. Sibley and R. W. Taylor, "A data definition and mapping language," Communications of the ACM, vol. 16, no. 12, December 1973, pp. 750-759.

[7] Clifford R. Hollander, "A scheme for representing descriptors as data," Proc. IFIP Congress 1974, Intl. Fed. Info. Processing Societies, Stockholm, Sweden (August 1974).

[8] William Kent, "A primer for normal forms (in a relational data base)," IBM Technical Report TR02.600 (December 1973).

[9] Edgar F. Codd, "Further normalization of the data base relational model," Courant Computer Science Symposium 6, Data Base Systems, New York, May 1971, Prentice-Hall.

[10] A. van Wijngaarden, et al, "Report on the algorithmic language ALGOL 68," Numerische Mathematic, 14 (1969), pp. 79-128.

[11] Zohar Manna, et al, "Inductive methods for proving properties of programs," Communications of the ACM, 16, 8 (August 1973), pp. 491-502.

DISTRIBUTION LIST FOR ONR ELECTRONICS PROGRAM OFFICE

No. of Copies	
1	Director Advanced Res. Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209 Attn: Technical Library
1	Office of Naval Research Electronics Program Office (Code 427) 800 North Quincy Street Arlington, Virginia 22217
6	Office of Naval Research Code 105 800 North Quincy Street Arlington, Virginia 22217
6	Naval Research Laboratory Department of the Navy Washington, D.C. 20375 Attn: Code 2627
1	Office of the Director of Defense Res. and Engrg. Information Off. Lib. Branch The Pentagon Washington, D.C. 20301
1	U.S. Army Research Office Box CM, Duke Station Durham, North Carolina 27706
12	Defense Documentation Center Cameron Station Alexandria, Virginia 22314
1	Director National Bureau of Standards Washington, D.C. 20234 Attn: Technical Library
1	Commanding Officer Office of Naval Research Branch Office 536 South Clark Street Chicago, Illinois 60605

No. of Copies	
1	San Francisco Area Office Office of Naval Research 50 Fell Street San Francisco, Calif. 94102
1	Air Force Office of Scientific Research Department of the Air Force Washington, D.C. 20333
1	Commanding Officer Office of Naval Research Branch Office 1030 East Green Street Pasadena, California 91101
1	Commanding Officer Office of Naval Research Branch Office 495 Summer Street Boston, Massachusetts 02210
1	Director U.S. Army Engineering Research and Development Laboratories Fort Belvoir, Virginia 22060 Attn: Technical Documents Center
1	ODDR&E Advisory Group on Electron Devices 201 Varick Street New York, New York 10014
1	New York Area Office Office of Naval Research 207 West 24th Street New York, New York 10011
1	Air Force Weapons Laboratory Technical Library Kirtland Air Force Base Albuquerque, New Mexico 87117

No. of Copies	
1	Air Force Avionics Laboratory Air Force Systems Command Technical Library Wright-Patterson Air Force Base Dayton, Ohio 45433
1	Air Force Cambridge Research Laboratory L. G. Hanscom Field Technical Library Cambridge, Mass. 02138
1	Harry Diamond Laboratories Technical Library Connecticut Avenue at Van Ness, N. W. Washington, D.C. 20438
1	Naval Air Development Center Johnsville Warminster, Penn. 18974 Attn: Technical Library
1	Naval Weapons Center Technical Library (Code 753) China Lake, Calif. 93555
1	Naval Training Device Center Technical Library Orlando, Florida 22813
1	Naval Research Laboratory Underwater Sound Reference Division Technical Library P. O. Box 8337 Orlando, Flordia 32806
1	Navy Underwater Sound Lab. Technical Library Fort Trumbull New London, Conn. 06320
1	Commandant, Marine Corps Scientific Advisor (Code AX) Washington, D.C. 20380
1	Naval Ordnance Station Technical Library Indian Head, Maryland 20640

No. of Copies	
1	Naval Ship Engineering Center Philadelphia Division Technical Library Philadelphia, Penn. 19112
1	Naval Postgraduate School Technical Library (Code 0212) Monterey, California 93940
1	Naval Missile Center Technical Lib. (Code 5632.2) Point Mugu, Calif. 93010
1	Naval Ordnance Station Technical Library Louisville, Kentucky 40214
1	Naval Oceanographic Office Technical Library (Code 1640) Suitland, Maryland 20390
1	Naval Explosive Ordnance Disposal Facility Technical Library Indian Head, Maryland 20640
1	Naval Electronics Lab. Center Technical Library San Diego, California 92152
1	Naval Undersea Warfare Center Technical Library 3202 East Foothill Boulevard Pasadena, California 91107
1	Naval Weapons Laboratory Technical Library Dahlgren, Virginia 22448
1	Naval Ship Research and Development Center Central Lib. (Codes L42 and L43) Washington, D.C. 20007
1	Naval Ordnance Lab. White Oak Technical Library Silver Spring, Md. 20910
1	Naval Avionics Facility Technical Library Indianapolis, Indiana 46218

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
A Closed System of Representation for Rationally-Organized Data and Its Descriptors

5. TYPE OF REPORT & PERIOD COVERED
Technical Note

6. PERFORMING ORG. REPORT NUMBER
Technical Note no. 53

7. AUTHOR(s)
Clifford R. Hollander

8. CONTRACT OR GRANT NUMBER(s)
N-00014-67-A-0112-0044

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Digital Systems Laboratory
Stanford University
Stanford, California 94305

14 p.

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Project 6961

11. CONTROLLING OFFICE NAME AND ADDRESS
Sponsored Projects Office
Stanford University
Stanford, California 94305

TN-53

12. REPORT DATE
Dec[redacted]r [redacted]974

13. NO. OF PAGES
16

14. MONITORING AGENCY NAME & ADDRESS (if diff. from Controlling Office)
Stanford Electronics Labs
Stanford University
Stanford, California 94305

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
data description
relation
descriptor
relation class
closed descriptive system

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Recently we have witnessed a dramatic increase in both the study and use of data bases. These activities have in turn stimulated interest in data description facilities. The work reported here was motivated by the observation that descriptors for structured data are themselves, in fact, "data-like" in many respects. This paper introduces a simple language for relationally-organized data and a companion descriptor language, for the purpose of demonstrating a single system of representation for both data constructs and their descriptors. It is assumed that each data construct (i.e., relation) belongs to a previously defined relation class

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

408071

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. KEY WORDS (Continued)

20 ABSTRACT (Continued)

and that a descriptor is used to define the structure of the elements of each class. The system presented here is "closed" in the sense that it allows each descriptor to be represented as a relation. Its adoption would permit data description facilities to be implemented in terms of data manipulation facilities.

DD FORM 1 JAN 73 1473 (BACK)

EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)