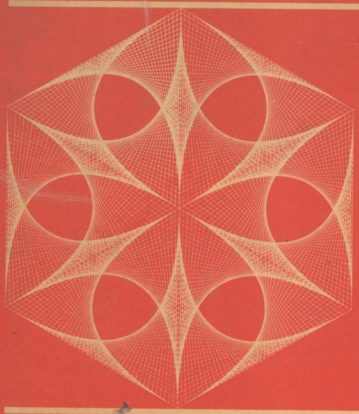


Applied Computer Science

Berichte zur praktischen Informatik

6

# Structured Database Programming

Von H. Wedekind

Hanser

The series "Applied Computer Science" wants to provide a fast dissemination of new, application-oriented developments in computer science. In it papers will be published which deal with methods for the solution of problems important for practical application; here belong also reports on application systems. It is intended for publication of:

Research Reports
Lecture and Seminar Notes
Proceedings of Symposia.

The principle areas of interest are:

1. Programming languages and compilers
2. Computer organization and operating systems
3. Data base and information systems
4. Computer Graphics
5. Software Engineering
6. Specific applications such as computer-aided instruction, process control, and computer-aided design.

Papers submitted for publication will be judged on the timeliness of the subject presented and the quality of the description. Manuscripts may be in German or English, and should be submitted to members of the editorial board. Authors will receive 25 reprints of their paper if published.

Manuscripts should have a length of at least 100 pages with a DIN A4 format. They will be photomechanically reproduced. The written area should not exceed 156 mm × 230 mm. Forms showing the proper layout area can be obtained from the publishers. Necessary corrections can be done as usual by pasting over or using correction fluid. Hand-written supplements (equations, special symbols, etc.) should be drawn with black Indian ink.

Die Reihe Applied Computer Science will der schnellen Verbreitung neuer, anwendungsorientierter Entwicklungen in der Informatik dienen. Dabei werden vor allem Arbeiten veröffentlicht, die Methoden zur Lösung für die Praxis relevanter Probleme enthalten; hierzu sind auch Berichte über Anwender-Systeme zu rechnen. Zur Veröffentlichung sind vorgesehen:

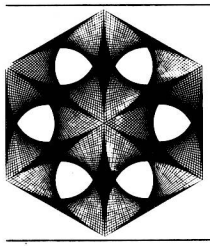
Forschungsberichte
Vorlesungs- und Seminarausarbeitungen
Tagungsberichte.

Es sollen insbesondere folgende Gebiete berücksichtigt werden:

1. Programmsprachen und Übersetzer
2. Rechnerorganisation und Betriebssysteme
3. Datenbank- und Informationssysteme
4. Graphische Datenverarbeitung
5. Software-Technologie
6. Datenverarbeitung für spezielle Anwendungen, wie z. B. Rechner-unterstützter Unterricht, Prozeßsteuerung und Rechner-unterstütztes Entwerfen und Konstruieren.

Entscheidend für die Veröffentlichung sind die Aktualität des behandelten Themas und die Qualität der Darstellung. Manuskripte in deutscher oder englischer Sprache nehmen die Mitglieder des Editorial Board entgegen. Autoren erhalten 25 Sonderdrucke ihres Beitrags.

Manuskripte im DIN-A4-Format sollen wenigstens 100 Seiten umfassen. Sie werden fotomechanisch vervielfältigt. Vordrucke mit eingezeichnetem Satzspiegel sind beim Verlag erhältlich. Die beschriebene Fläche sollte nicht größer als 156 mm × 230 mm sein. Notwendige Korrekturen können in üblicher Weise durch Überkleben oder Verwenden eines Korrekturlackes vorgenommen werden. Handschriftliche Ergänzungen (Formeln, Sonderzeichen usw.) werden am besten mit schwarzer Tusche eingezeichnet.



Applied Computer Science

Berichte zur praktischen Informatik

# Band 3
# Krönig, Praxis von Sprachen, Programmsystemen und Programmgeneratoren

insbesondere in der Prozeß-Datenverarbeitung, in technisch-wissenschaftlichen Anwendungen und beim Einsatz von Kleinrechnern. 1. Treffen 1976 des German Chapter of the ACM (Association for Computing Machinery). Herausgegeben von Dr. Dirk Krönig, Konstanz. 194 Seiten. 1976. Kartoniert.

# Band 4
# Noltemeier, Graphen, Algorithmen, Datenstrukturen

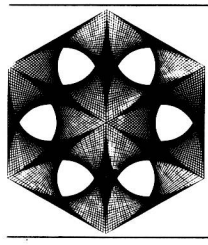
Ergebnisse des Workshops WG 76 (2. Fachtagung über graphen-theoretische Konzepte in der Informatik) vom 16. bis 18. Juni 1976 in Göttingen. Herausgegeben von Prof. Dr. Hartmut Noltemeier, Göttingen. 336 Seiten. 1977. Kart.

# Band 5
# Martin, Mikrocomputer in der Prozessdatenverarbeitung

Aufbau und Einsatz der Mikrocomputer zur Überwachung, Steuerung und Regelung. Von Prof. Dr.-Ing. Wolf Martin, Berlin. VIII, 2o5 Seiten, 85 Bilder, 13 Tabellen. 1977. Kartoniert.

Carl Hanser Verlag, Postfach 86 o4 2o, 8 München

结构数据库程序



Applied Computer Science
Berichte zur praktischen Informatik

# Editorial Board



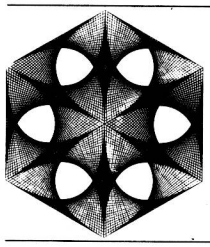
## Band 1
## Pape, Graphen–Sprachen und Algorithmen auf Graphen

Herausgegeben von Prof. Dr. Uwe Pape, Berlin. 1. Fachtagung über Graphentheoretische Konzepte der Informatik (Graphentheoretic Concepts in Computer Science) 2.-4. Juli 1975, Berlin. 3. Workshop und Treffen des German Chapter of the ACM über Graphen-Sprachen und Algorithmen auf Graphen. 236 Seiten. 1976. Kartoniert.

## Band 2
## Hoßfeld, Praxis der Realisierung von Informationssystemen

Herausgegeben von Dr. Friedel Hoßfeld, Jülich. 4. Workshop und Treffen des German Chapter of the ACM am 18. November 1975, Jülich. 224 Seiten. 1976. Kartoniert.

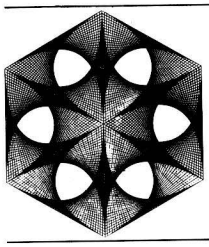
Carl Hanser Verlag, Postfach 86 04 02, 8 München



Applied Computer Science
Berichte zur praktischen Informatik

# 6 Structured Database Programming

If you see a giant
look where the sun stands
and make sure
that it is not the
shadow of a pygmy.
(Novalis, German poet and philosopher)



Applied Computer Science

Berichte zur praktischen Informatik

Carl Hanser Verlag München Wien

# Structured Database Programming

von Prof. Dr. H. Wedekind

Mit 19 Bildern



Carl Hanser Verlag München Wien 1977

Dr. Hartmut Wedekind is professor of business administration
and informatics at the Technical University in Darmstadt (Germany).

CIP-Kurztitelaufnahme der Deutschen Bibliothek

Structured database programming / hrsg. von H. Wedekind.
– 1. Aufl. – München, Wien: Hanser, 1977.
(Applied computer science: 6)
ISBN 3-446-12371-7

NE: Wedekind, Hartmut [Hrsg.]


Druck: Georg Appl, Wemding
Printed in Germany

## Preface

The monograph attempts to apply the ideas of Structured Programming to data processing tasks in a database environment. Programming in database systems differs from the conventional approach in the use of a data manipulation language (DML). This particular language for interrogating and maintaining databases may be embedded in a general purpose programming language, or it may stand alone, in which case it is called a query language.

With the rise of high-level and nonprocedural DML's the programming methodology was fundamentally affected. Data processing tasks with database interrogation were all of a sudden easy to program in comparison to the hard job to be accomplished with procedural languages. In nonprocedural languages a set of records may be selected by its content. The selection conditions are specified in static predicate statements in a way also found in program verification theory. In a broader context one can say that a nonprocedural program is a prescription for solving a problem without regard to details of how it is solved. What has to be solved is explicitly specified in terms of structures or abstractions which are relevant to the problem. Procedural languages on the other hand control the access of single records dynamically. The commands of procedural languages show that procedural programming is based to a great extent on the inherent sequential organization of conventional digital computers.

The stepwise refinement of a problem as well as the piecewise composition of a program yields a structural building. It is of vital importance that the target language supports the structuring process. Codasyl's DBTG and IMS of IBM are not considered because their DML's make it difficult to describe the genesis of a program. Although data independence was considered as the main reason for establishing databases, current data-

base management systems are not germane to the task. Most systems provide, in effect, a single data structure with many optional parts. It is up to the user to fit his structure to this framework.

The relational data model and its nonprocedural languages provides concepts and methods adaptable to the Structured Programming approach. The similarities in designing a relational database and a well-structured program are amazing, even in details. Globally speaking, both methodologies first try to master the logical complexity, before tackling problems of performance on access path and device levels. The essential part of every programming task with regard to formatted data can be structured by a powerful, nonprocedural DML like SEQUEL (Structured English Query Language), leaving only minor issues to the procedures of the host program. Structured Database Programming will radically change the common practice of commercial program design. In the future a well-structured program will be a program understandable in a static fashion, or as Dijkstra (11) puts it: Our intellectual powers are rather geared to master static relation and our power to visualize processes evolving in time are poorly developed. For that reason we should do our utmost to shorten the conceptual gap between the static program and the dynamic process, to make the correspondence between the program (spread out in text space) and the process (spread out in time) as trivial as possible.

The methodology outlined in this monograph was highly influenced by the outstanding book "Principles of Program Design" by M. Jackson. In particular the representation of data structures, the schematic logic and some examples are taken from him. In fact, the monograph could be regarded as an extension of Jackson's book to shared (global) data in a database. We are dealing only with the structural specification of programs and not with the nitty-gritty details of runnable programs.

The monograph, however, is not confined to nonshared (local) data accessed only by individual programs.

It is generally very difficult to keep up with a field that is economically profitable, and it is only natural to expect that some of the techniques described here will eventually be superseded by better ones. With respect to the database management system, we selected here as a vehicle the System R, a project of IBM Research in San Jose (Cal.). Since System R is still in a research environment, some unexpected changes may occur from this area. Nevertheless, it is the author's hope that further research will be stimulated particularly with regard to the assessment of various program structures.

Programming can only be taught by leaning heavily on examples. Among the practical problems the most important ones are the payroll example, the creation of purchase orders, the evaluation of system log-files and the bill of material explosion. The examples are more comprehensive programming tasks and not just stand-alone queries, by which concepts and semantics of DML's are generally explained. Without examples the methodology behind is simply not comprehensible. The examples, however, have to be looked upon only as carriers and not as the subject of the monograph.

The composition of this publication is in English and not in German, as one might have expected from the author's native tongue. The switching to a foreign language was required, because a lecture on the subject was first given in English at the University of Sao Paulo during a sabbatical stay in Brazil. I am very grateful to Mrs. Sara Upmeyer for preparing the manuscript technically.

Darmstadt, summer 1976 H. Wedekind

## Contents

# 1. Introduction

In the last decade structured programming was developed as a design methodology to control the process of constructing programs. In particular Dijkstra, Hoare (9) and Wirth (27) contributed the cornerstones to this scientific discipline, which is today considered as the only valid approach to tackle complex programming tasks in a systematic way. Until the books of Wirth (28) and Jackson (17) were published, structured programming was demonstrated with examples rather complex in the algorithmic part, but very simple in the data structures used. The eight queens problem (find all the positions of eight queens on a chessboard such that they cannot capture each other) is one of the most prominent examples of this type. Wirth (28) with regard to systems programming problems and Jackson (17) with respect to commercial applications showed how a program structure should be developed on the basis of a problem-oriented data structure.
The structured programming methodology was extended by a rule specifying that the program structure has to emerge out of a problem-oriented data structure. Hoare in particular originated the data-oriented methodology with his contribution "Notes on Data Structuring" (9, p. 83-174) continued by Wirth and Jackson. Program structures and data structures are looked upon as being so closely related that a one-to-one correspondence can be established between their structural elements (see Wirth (28, p. 321)):

| structural elements | program structure | data structure |
|---|---|---|
| 1) primitive | 1) assignment | 1) field (attribute) |
| 2) collection | 2) compound assignments | 2) record types |
| 3) choice | 3) selection | 3) disjunct groups of record types |
| 4) repetition | 4) iteration | 4) repeating groups of records, sequence of records (files) |
| 5) general graph | 5) goto-jump | 5) pointer |
| 6) recursion | 6) procedure | 6) recursive data structures (tree structures) |

Assuming that there are no clashes or disharmonies between the problem to be solved and the data structure, then the structured programming methodology according to Wirth and Jackson asserts that program structuring is in its essence a data structuring if the sources of complexity are not the algorithms but the data as in commercial environments.

One major issue for database systems on the other hand is the demand on data independence. Data independence requires the insulation of programs and data because a change in the data may corrupt the whole program. There are maintenance aspects aiming at a separation of data and program structures. The logical problem structure cast into data structures and translated into program structures in the structured programming fashion on the one hand and the claimed data independence in database systems on the other hand have an opposite tendency. One of the key points of this monograph is to make the trade-off between these tendencies easier by simplifying the data structure and thereby the program structure. We apply a multiple-record-at-a-time access logic (set logic) on simple files (relations). From a logical point of view groups of records are accessed determined by their contents. The single record access logic is then not in the programs to be developed but in the data access mechanism of the database