A USER-ORIENTED TRANSACTION DEFINITION FACILITY FOR A RELATIONAL DATABASE SYSTEM

by

C. STEVEN ROUSH

B. S., Kansas State University, Manhattan, Kansas, 1972

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements of the degree

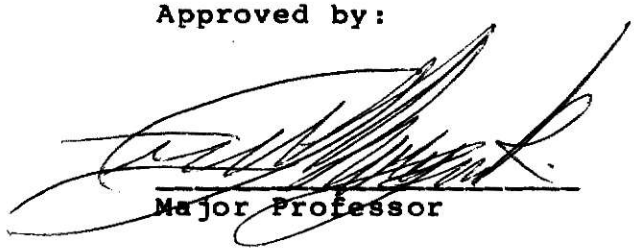
MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1979

Approved by:



Major Professor

ACKNOWLEDGEMENTS

I want to thank Dr. Fred Maryanski for keeping my work on track and NCR, Inc. for funding this project.

TABLE OF CONTENTS

LIST OF FIGURES

1. INRODUCTION

Software has become the major obstacle preventing users from reaching their automation goals. According to Boehm's projections, software now accounts for approximately 80% of every dollar invested in computing, or about 20 billion dollars annually[1]. Further, both Boehm[2] and Myers[3] quote projected software growth figures of 15 to 25% annually. It is not clear how these growth rates can be attained.

Myers states that productivity increases will only average 3 to 7% during the period from 1955 to 1985[4], and Boehm estimates that during the ten years before 1985 improved productivity and increased numbers of programmers together can only account for a software growth rate of 12 to 17%[5]. Mills, however makes these estimates seem optimistic. He notes that software maintenance already requires about 75% of the total software effort, and that "unless radical methods are found, maintenance will go even higher in its demands and will very nearly stifle development"[6].

There are other problems with current software production techniques which must also be noted. Software has a reputation for being delivered late and being unreliable when it is delivered[7]. Finally, because current programmer productivity appears to be between 600 and 6000 instructions per man-year[8], it is very difficult for software systems to keep pace with the changing needs of

the users.

For these reasons, a large effort is being made to help reduce the problems of software development. Structured design, structured programming, new language design, program proving, project management techniques and database management systems have all evolved with the intention of improving the quality of software. This report will discuss work done in the database management area that may eventually help reduce the effort required to create and maintain a body of software, thus helping to enhance the utility of the computer.

Specifically, the report discusses the design and implementation of a subsystem of a user-oriented database management system which is under development. The goal of this prototype system, which is named SNAP, is to allow transactions, both retrieval and maintenance, to be defined by people who are not computer professionals, and entire systems of transactions to be defined by these same people with professional assistance required only during the process of identifying the interrelationships between the data items in the system. Though far from complete, the prototype database system does currently allow a trained professional to define transactions which would probably satisfy many of the user requirements to retrieve (display) their data.

The subsystem described by this report is named the Transaction Definition Facility (TDF). Its goal is to

interact with the user to define his transactions in terms which do not require knowledge of the internal structure of the database (the schema), and to produce programs to actually perform those transactions.

SNAP's design has been based on two assumptions. The first is that users of a relational database in third normal form (3NF) require less knowledge of the database schema to successfully perform transactions than users of databases which have not been so rigorously defined. Instead, the database management system can rely on the axioms which are known to apply to relations in 3NF to supplement a user's commands. The second assumption is that a user's database transactions are heirarchical in nature. That is, that transactions are composed of 'lines' which represent various levels of detail or nesting, but those levels of detail never partially overlap. This is analagous to the concept of blocks in a block structured language. Figure 1. presents an example of a three level transaction.