The INGRES Papers: Anatomy of a Relational Database System

Michael Stonebraker, Editor

22/03

The INGRES Papers: Anatomy of a Relational Database System

Michael Stonebraker, Editor
University of California, Berkeley

ADDISON-WESLEY PUBLISHING COMPANY
Reading, Massachusetts • Menlo Park, California • Don Mills, Ontario
Wokingham, England • Amsterdam • Sydney • Singapore • Tokyo • Mexico City
Bogotá • Santiago • San Juan

This book is in the Addison-Wesley Series in Computer Science.

Consulting Editor: Michael A. Harrison

Library of Congress Cataloging in Publication Data
Main entry under title:

The INGRES papers.

Bibliography: p.
Includes index.
1. Data base management--Addresses, essays, lectures.
2. INGRES (Computer system)--Addresses, essays, lectures. I. Stonebraker, Michael.
II. Title: Relational database system.
QA76.9.D3I535 1986 001.64'2 84-24584
ISBN 0-201-07185-1

ABCDEFGHIJ-HA-898765

Preface

This book summarizes the work of the INGRES project at the University of California, Berkeley over the years 1973-1983. The project was initiated by Professor Eugene Wong and myself in late 1973 and has been in existence since that time. Professor Lawrence Rowe joined the project in 1976.

Initially, the project goal was to construct a working relational database system. In 1970, Ted Codd proposed the relational model as a better foundation on which to build a database system. His ideas (as reported in [CODD70, CODD71, CODD72]) were widely debated in the early 1970s. The critics insisted that a relational database system could not run efficiently. Moreover, even if one could be built, no REAL programmers could be taught to use the novel languages which were being proposed. On the other hand, Codd's supporters claimed that older technologies (e.g., network and hierarchical data models) were too complex, too difficult to program and offered insufficient insulation from changing data structures. The issues were hotly discussed at annual SIGFIDET (now SIGMOD) conferences, and formally debated at the 1974 conference. Eugene Wong and I had begun reading papers in database management in 1972, and it was apparent to us that the relational model offered simplicity and elegance. It seemed to us that this had to be the way of the future, and we embarked on an implementation project to prove that an efficient, easy-to-use system could be built.

The INGRES system first ran in 1975. During 1976 and 1977, we produced increasingly reliable and functional systems. Because our software worked in a UNIX environment, and there were few other database alternatives available, we were able to convince a substantial collection of outside users to try INGRES. By 1978 we had considerable user feedback and were aware of the benefits and shortcomings of our software. Fixing the drawbacks would involve a lot of work and did not seem like a reasonable research project. Hence, active development of the University of California INGRES prototype tapered off in 1979. Since then, several companies (including Amdahl, ELXSI, Computer Associates, Relational Technology, and

NBI) have produced commercial products based on the University of California code.

The first section of this book contains a collection of three papers that chronicle the development of the INGRES system. The papers were written in 1976, 1978, and 1984. The final paper in the trilogy describes the experience of one company, Relational Technology, in correcting the earlier drawbacks.

During the University of California development phase, we made many engineering tradeoffs in the design of INGRES. Many of these issues were concurrently investigated formally by members of the INGRES team. The second section of this book contains a representative collection of supporting studies on issues faced in building a database system. Two of the papers cover the traditional topics of concurrency control and query optimization. The third paper uses INGRES as a vehicle to investigate what database machine architectures might be feasible. The fourth paper uses INGRES to study compilation, microcode, a fast file system, and a special purpose operating system as performance enhancement techniques. The final paper in Section 2 contains a collection of complaints and frustrations concerning the services provided by contemporary operating systems.

In 1977, we enlarged our development sights to include managing data distributed across multiple computer systems in a computer network. Although fraught with implementation problems, Distributed INGRES worked in a two-site configuration over a 9600 baud RS232 interface in 1981 and worked with an arbitrary number of sites over an Ethernet in 1983. The third section of this book includes four papers that discuss this development project, and the lessons we learned from it. The first one indicates the general characteristics of our design, and some performance numbers, and a backward look at problems encountered. Then, two papers which are representative of our thoughts have been included on query processing in a distributed environment. The last paper discusses protocols necessary to correctly recover from crashes in a distributed environment.

During the development of INGRES, we have designed and/or implemented several user interfaces to database systems. Our first programming language interface, EQUEL, remains today as a reasonable mechanism for coupling an existing host language to a database system. Initial use of this interface convinced us that any host language coupling would be fraught with impossible dilemmas. To explore alternative solutions to these problems, we designed and implemented a new programming language, RIGEL, which performed both general purpose computation and database access in one environment. A processor for RIGEL was constructed between 1977 and 1979. More than a dozen external users tried RIGEL in 1979 and 1980. The second paper in Section 4 discusses constructs in this language. A third user interface which we constructed in 1981-83 was oriented toward forms. All real database applications spend an inordinate amount of code manipulating the screen, and we attempted to reduce this effort by an order of magnitude. The third paper in this section discusses the design of the resulting system, FADS. The last paper in this section proposes a new programming language interface, which may someday replace EQUEL. This interface was motivated by the special needs of application programs which allow users to browse through a collection of data making random changes.

Many researchers and practitioners agree that relational database systems are well suited to business data processing applications. However, there are many other kinds of users with database problems. For lack of a better word, we will term them "non-traditional" applications, and it is evident that their database needs are somewhat different than business ones. Many researchers in the database community are engaged in a "group grope" for ideas that address the issues of this community. Section 5 contains a set of four papers that propose various extensions to the relational model appropriate in nonbusiness environments. At the moment, the INGRES project is attempting to release a new version of the system (Version 8), which contains many of these ideas. We have always prototyped our ideas, and now seems an appropriate time to integrate the more valuable ones, so others can try them out. It is hoped that such research eventually results in a small set of powerful primitives that can address the needs of such clients, and do so with the same robustness that relational database systems have addressed the needs of business data processing applications.

One of the most difficult database problems faced by any real world application designers is to specify his "schema," and then make access path choices to generate good performance for his application. The twin problems of logical database design and physical database design must be overcome. Section 6 contains two papers which represent our insights into this important area.

The future directions of the project are primarily in the area of database support for nontraditional applications. We are pursuing ideas to support text processing, expert systems, and spatial data applications in a database context. We have decided not to attempt a release of Distributed INGRES because the code is not reliable enough to be useful to others.

Because the project is at a crossroads, it seems appropriate to summarize past successes and failures at this time. The papers in this book have been selected with this goal in mind, and were all written by students and professors at the University of California who were associated with the INGRES project.

The INGRES project has always been organized as a chief programmer supported by a team of four to six people. The chief programmer was the only full-time employee; all other people associated with the project have been students intent on passing courses and obtaining degrees. The chief programmers have been:

Dr. Gerald Held (1973-1975)
now Director of Strategic Planning at Tandem Computers, Inc.

Mr. Peter Kreps (1975-1977)
now Member of the Technical Staff at Relational Technology Inc.
Dr. Robert Epstein (1977-1980)
now Vice President at SYBASE
Mr. Eric Allman (1980-1982)
now Member of the Technical Staff at Britton-Lee, Inc.
Mr. Joseph Kalash (1982-1985)
now Program Manager at Unisoft

Whatever success the project has had rests largely on the contributions of these people. Each was the "keeper" of the INGRES knowledge (i.e., how the system really worked) and was the ultimate authority on how to fix bugs. Moreover, large portions of the system were written by them. It would be impossible to list all of the students who have been associated with the INGRES project but I feel that it is important to acknowledge the contributions that some of them have made to developing the various systems that we have built.

The following students were largely responsible for implementing the early versions of the INGRES system:

Mr. James Ford (1973-1975)
now Principal Member of the Technical Staff at CXC Corp.
Mr. William Zook (1973-1975)
Mr. Rick Berman (1973-1975)
now Section Manager at Tandem Computers, Inc.
Mr. Nick Whyte (1976-1978)
now Manager of Data Base Development at ELXSI Computers, Inc.
Mr. Peter Rubinstein (1975-1977)
Ms. Iris Schoenberg (1974-1975)
Ms. Angela Go (1974-1975)
Ms. Carol Joyce (1974-1976)
now Member of the Technical Staff at Relational Technology Inc.
Dr. Karel Youssefi (1973-1976)
now Member of the Technical Staff at Tandem Computers, Inc.
Dr. Nancy McDonald (1973-1975)
now Senior Scientist at GTE Data Services
Mr. Michael Ubell (1975-1977)
now Director of IDM Software at Britton-Lee, Inc.
Dr. Daniel Ries (1975-1978)
now Director, End User Product Development at Computer Corporation of America, Inc.
Dr. Paula Hawthorn (1976-1979)
now Director of Product Development at Britton-Lee, Inc.
Ms. Polly Siegal (1976-1978)
now Development Engineer at Hewlett-Packard

Mr. Marc Meyer (1978-1981)
now an independent consultant
Dr. Randy Katz (1977-1980)
now Professor of Computer Science at the University of California, Berkeley

Distributed INGRES was largely coded by the following students:

Mr. John Woodfill (1979-1983)
now a graduate student in Computer Science at Stanford University
Mr. Jeff Ranstrom (1979-1982)
now Member of the Technical Staff at Altos Computers, Inc.

In addition, the following students contributed various modules to later versions of the code:

Ms. Nadene Lynn (1981-1982)
now OEM Support Manager at Relational Technology Inc.
Mr. Robert McCord (1980-1982)
now Project Manager, Database Systems at Tolerant Systems, Inc.
Mr. Dennis Fogg (1981-1982)
now a graduate student in Computer Science at M.I.T.
Mr. James Ong (1981-1982)
now a graduate student in Computer Science at Yale
Ms. Heidi Stettner (1981-1982)
now Audio Systems Programmer at Lucas Films, Ltd.

RIGEL was primarily coded under the supervision of Larry Rowe by the following students:

Dr. Kurt Shoens (1979-1981)
now Member of the Technical Staff at IBM Research
Mr. Dan Brotsky (1978-1979)
now a graduate student at M.I.T.
Mr. Joseph Cortopassi (1978-1980)
now Manager of User Interfaces at Relational Technology Inc.
Mr. Doug Doucette (1978-1980)
now Member of the Technical Staff at Tolerant Systems, Inc.

FADS was primarily coded by Kurt Shoens as a portion of his Ph.D. dissertation with assistance from:

Mr. Mark Hanner
now a product marketing engineer for Relational Technology Inc.

The INGRES project has been supported primarily by research grants from various federal agencies. The support of Mr. John Machado of the Naval Electronics Systems Command, Captain William Price of the Air Force Office of Scientific Research, Dr. Jimmie Suttle of the Army Research Office, and Dr. Rick Adrion and Dr. Bernard Chern of the National Science Foundation are especially appreciated.

This book was edited using UNIX document processing facilities primarily by Beatrice Dryfoos and Beth Rabb to whom I am deeply indebted. Typesetting was expertly done by Beatrice Dryfoos.

Berkeley, California *M.S.*

Contents

SECTION

1

Design of Relational Systems

This section of the book contains a trilogy of papers describing the design and implementation of INGRES. The first paper describes INGRES as it existed in early 1976. This paper sketches the original design principles and the reasoning behind them. It was written when the original system was emerging and appeared in the September 1976 issue of *Transactions on Data Systems* (TODS).

The second part of the trilogy was written in late 1978 when initial performance tests on the code had been done and we discovered some major design flaws. At this point, it was clear that major problems existed with the code, and the INGRES project was having a "what's next" crisis. The original title of the paper (Chapter two here) was "Requiem for a Data Base System," but the TODS reviewers thought the title was too somber. It appeared as "Retrospection on a Database System" in the September 1980 issue of TODS.

The last paper in the trilogy was written in early 1984 and describes the changes to INGRES since 1978. Since the majority of the work on the code has been done at Relational Technology Inc., the paper is aptly named "The Commercial INGRES Epilogue."

Keep the following thoughts in mind as you read these papers. First, notice some of the naive comments in the first paper. There are remarks that the UNIX kernal will be used for buffer management and that a multiple process database system should execute as fast as a single process one. Such comments reflected the inexperience of the designers concerning the real operating system costs of various functions. The second paper discusses the extreme cost of such operations once INGRES was running and benchmark testing had been performed.

[redacted] Notice the discussion in the three papers concerning the tradeoff of space for speed. The earlier system was constrained to a small address space machine and all tradeoffs were forced to the "small space" alternative. The third paper indicates the ways in which the code was speeded up by trading space for time (e.g., caching the "state" of past commands).

Then, notice the willingness of the design team to rethink past decisions and to rewrite major pieces of the system, if appropriate. The adage that it is never too late to throw everything away and begin again with a clean slate should always be kept in mind.

Notice also the desire to use the operating system facilities intact for the purposes for which they were designed, regardless of the performance consequences. Notice that INGRES continues to use the operating system scheduler, messages, and file system. Although loud complaints appear in a later section about operating system services, it is clear that we had no desire to redo such services if we could avoid it.

In the first paper note the defense of single statement transactions and a single level of delegation in the protection system. Such remarks reflect our absence of maturity about these issues.

Finally, notice that the theme of rapid prototyping is prevalent throughout the three papers. The second paper describes an early INGRES user using the now popular technique for application design. Moreover, it is evident that the INGRES design team used this approach for all their development.

From the third paper, one can note the direction in which INGRES is moving. Tactics include generating a query plan by examining all (or most) of the possibilities and saving the plan for reuse if possible. Plans should include both merge-sort and tuple substitution as possible tactics. These decisions are ones that the System R designers also arrived at [SELI79] and are commonly believed correct in 1984.

One might ask, "what implementation issues are not well understood at the moment?" The rest of this introduction addresses this question. First, there is a trend by operating system designers to move transaction management inside the operating system. This topic is briefly addressed in the next section, and it appears that there are substantial implementation problems to be overcome [STON84]. Hence, the proper place of transaction management is still an open issue.

Secondly, most database management systems peak at 50 or fewer transactions per second in large transaction-processing applications. The reason is not insufficient CPU horsepower or I/O bandwidth but a variety of software bottlenecks. For example, while a lock is being set, the lock table is temporarily in an inconsistent state. As a result, a process to set a lock must get exclusive control of the lock table for the duration of its modification to the data structure. In applications with a very large number of terminals (say 1000), there may be queuing delays to access the lock table. Another problem is that standard database practice requires that a log record be written out to disk at the conclusion of a transaction. Many database systems have a single log file per database; hence, one must write 50 log records to a single file in one second in order to commit 50 transactions per second. This is difficult to do with most contemporary operating systems. How to overcome this myriad of software bottlenecks and achieve dramatic increases in transaction throughput (say 1000 per second) remains an open question.

A third issue is storage of complex objects in a database system. This topic is extensively discussed in Section 5.0 for engineering applications. However, one should note that data dictionaries for relational database systems are being enlarged to hold information on reports, forms, graphs, application programs, and so forth, as well as information on relations and storage structures. The information about a form includes the layout of its fields, the position of trim elements, which fields are protected, and so forth. Such information does not naturally fit the relational model. I feel that extending the semantics for the relational model to handle such complex objects is one of the most pressing open issues.

Nowadays, computers are being sold with increasing amounts of main memory, and buffer pools with a size of several megabytes are becoming commonplace. Clearly, database systems must change over the current decade to manage a database that is partly on disk and partly in main memory. This possibility suggests for example that hash-join algorithms may be a viable query processing tactic [DEWI84]. Re-engineering current systems to take advantage of massive amounts of main memory is a challenging open issue.

Lastly, I predict that optical disks will finally see the light of day and become commercially significant. A "write once" device with a very large capacity has considerable appeal. Clearly, one can put the database log on such a device. However, one might be able to build a data structure in which data records are never overwritten. This would allow all past values of all items in the database to be accessible and would allow one to ask questions of the database as of some particular point of time in the past. Building such a data structure on a write-once medium is a challenging open issue.

CHAPTER ONE

The Design and Implementation of INGRES

Michael Stonebraker / Peter Kreps / Eugene Wong / Gerald Held

1.0. INTRODUCTION

INGRES (Interactive Graphics and Retrieval System) is a relational database system which is implemented on top of the UNIX operating system developed at Bell Telephone Laboratories [RITC74a] for Digital Equipment Corporation PDP 11/40, 11/45, and 11/70 computer systems. The implementation of INGRES is primarily programmed in C, a high level language in which UNIX itself is written. Parsing is done with the assistance of YACC, a compiler-compiler available on UNIX [JOHN74].

The advantages of a relational model for database management systems have been extensively discussed in the literature [CODD70, CODD74, DATE74] and hardly require further elaboration. In choosing the relational model, we were particularly motivated by the high degree of data independence that such a model affords, and the possibility of providing a high level and entirely procedure-free facility for data definition, retrieval, update, access control, support of views, and integrity verification.

1.1. Aspects Described in This Paper

In this paper we describe the design decisions made in INGRES. In particular we stress the design and implementation of:

1) the system process structure (see Section 2.0 for a discussion of this UNIX notion);
2) the embedding of all INGRES commands in the general purpose programming language C;

M. Stonebraker, P. Kreps, E. Wong, and G. Held, *ACM Transactions on Database Systems*, vol. 1, no. 3, September 1976. Copyright 1976, Association for Computing Machinery, Inc.; reprinted by permission.