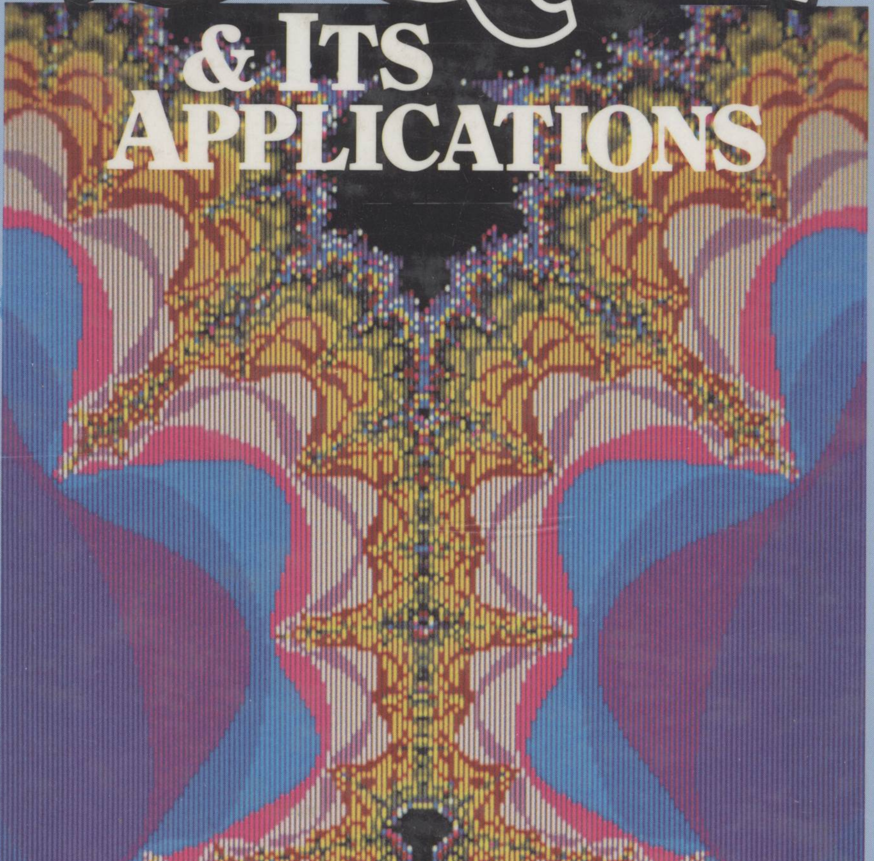
SQL & ITS APPLICATIONS



Raymond A. Lorie
Jean-Jacques Daudenarde

SQL and Its Applications

Raymond A. Lorie
Jean-Jacques Daudenarde

PRENTICE HALL, Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

```
Lorie, Raymond A.
    SQL and its applications / Raymond A. Lorie, Jean-Jacques
  Daudenarde.
       p.   cm.
     Includes bibliographical references and index.
     ISBN 0-13-837956-4
     1. SQL (Computer program language)  2. Data base management.
  I. Daudenarde, Jean-Jacques.   II. Title.
  QA76.73.S67L67  1991
  005.75'6--dc20                                            90-39784
                                                                CIP
```

Editorial/production supervision: *Mary P. Rottino*
Cover design: *Lundgren Graphics, Inc*
Manufacturing buyers: *Kelly Behr and Susan Brunke*
Cover photo: *Larry Keenan/The Image Bank*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-837956-4

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

To

Ileana, Miguel, Ignacio, and Francisco

Hélène, Sophie, and Eric

Preface

Many programs that run on data processing systems today rely heavily on data that are stored "permanently" on some medium such as magnetic disks. The collection of such preserved data is called a ***database***. Any system that helps the user in managing a database is a ***database management system*** or ***DBMS***. Some of the most important functions of such a system deal with the creation of a database, the insertion of information into the database, the retrieval of the stored information, its protection against unauthorized access, and its recovery from system malfunctioning.

The relational model for the management of data was introduced in 1970 in a paper by E. F. Codd, published in the *Communication of the ACM, 13:6* entitled "A Relational Model for Large Shared Data Banks." This original paper triggered an extensive research in the database field in the 1970's, both theoretical and practical. In the early 1980's, database management systems based on the relational model became available commercially. Starting modestly, it has now become a technology of major importance and several products are available from various companies such as IBM, Oracle, Ingres, and Teradata, to name a few. More and more data processing shops are installing these products, using them for developing applications, and/or querying their databases.

The data language used to interact with the database plays a fundamental role in a DBMS. Although several classes of languages have been proposed in the past, the ***Structured Query Language***, or ***SQL***, has emerged as the most commonly used relational language. Initially introduced in System R, a research prototype developed at IBM Research in San Jose, California, SQL is now used in a large number of products, and standardization efforts are under way.

Up to now, most users of relational systems were specialists, accustomed to learn new systems by reading long user's manuals or papers published in the technical literature, or by gaining hands-on experience on systems installed in large data processing accounts. But the situation is changing. First, more users are getting involved with relational systems because more applications are being developed based on the new technology. Second, relational systems are becoming available on much smaller processors, even personal computers, opening the field considerably. The new breed of users consists of application analysts and programmers or end users who are specialists in their fields, but not necessarily in data processing or database technology. These professionals know their information management problems and want to know if and how relational systems - and SQL in particular - can help. This book caters to their needs, as well as the needs of anybody who wants simply to learn about the relational approach to data management at a pragmatic level.

The book is divided into three parts. Part 1 is introductory in nature. Part 2 presents the different constructs of the language. Many short examples are given; they all refer to the same application, making it easy to remember the context and understand what each example is about. While Part 2 studies each SQL statement individually in a convenient order, Part 3 looks at various applications. Each chapter analyzes one of these applications, with emphasis on a certain generic problem. Each time, important questions are considered, such as how to organize the data, or can a certain logical operation be done with a single SQL statement? If not, can a sequence of statements do the job? When is it necessary to write some code in a host language? The same questions will eventually come up during the design of any application, and the case studies presented here should prepare the reader to recognize the multiple trade-offs, and make the right choices.

When developing Part 2 we realized that we had to face a certain dilemma. SQL is a language which is still young; and although a large portion of the syntax and semantics of the language is common to all implementations, there are differences. We could have chosen to cover one particular instance of SQL, or we could have chosen to discuss only what is common to all important implementations, or to concentrate on only those constructs that have practically been standardized. Instead, we chose to introduce all the concepts that exist in most SQL. When the syntax - or the semantics - differs, we say so and show one particular instance. This approach allows us to be complete in covering the concepts; we feel this is important. We do not claim to be exhaustive in giving all features of all implementations. Consider, for example, the functions that apply to data of type "character string": once the reader full exploitation of such relationships is not trivial; Chapter 14 discusses some interesting algorithms.

Finally, Chapter 15 goes through the design of an interactive, menu-driven application; it mainly illustrates the trade-offs between ***static*** and ***dynamic*** SQL and presents some details on how to use the dynamic approach. Chances are that you will never have to use dynamic SQL. But, just in case you have to, this chapter introduces the most important concepts.

Chapters 2 to 9 end with short series of exercises. All these refer to a common application context that is described in Chapter 2; the answers are given in Appendix.

Acknowledgments

We would like to thank our colleagues at IBM Research for keeping up an environment in which relational databases and SQL in particular have been - and still are - the base for interesting research; we have learned a lot from them. We are grateful to Charles Bontempo, Horacio Terrizzano, and a third anonymous reviewer for their many comments and suggestions. Some of these suggestions resulted in significant changes. We also acknowledge the help of Ignacio Terrizzano and David Torney, who read an early manuscript and whose comments have been particularly valuable. Finally, we are indebted to the IBM Corporation, and the management of the Research Division in particular, for providing some of the computer resources needed for the production of this book; we also thank the editors and staff at Prentice Hall for their constant support.

Raymond A. Lorie
IBM Corporation
Research Division
San Jose, California

Jean-Jacques Daudenarde
IBM Corporation
Storage Systems Products Division
San Jose, California

understands that there are such functions - for example to extract a substring - it is less important to give the exhaustive list of functions implemented in the various systems. Such a list is easily found in the reference manual specific to an implementation.

Again, the main goal of the book is to explain the concepts of SQL and show how these concepts can be helpful in solving a wide variety of information problems.

Chapter organization

The first chapter provides an introduction to the relational way of organizing data. It describes a particular application and shows how its data can be couched in relational terms. Then, SQL is briefly introduced, showing examples of data definition, manipulation, and query capabilities. We believe it is important to know, from the start, how an SQL system handles the requests; for that reason, Chapter 1 covers the embedding of SQL in programs, as well as the issues of ***dynamic*** versus ***static*** SQL.

Chapter 2 considers another application and shows how the concept of ***normalization*** helps the designer to come up with an appropriate data organization.

Part 2 covers the language itself.

Chapter 3 introduces the basics of ***data definition***. This is normally the first set of SQL facilities that a user would need when getting familiar with an SQL system in a hands-on session. It also discusses how data can be inserted into the database. At this point, the reader can create a table and populate it with data.

Chapter 4 discusses the basic ***query*** capabilities of SQL but only those that apply to a single table. Many examples are given; they are all based on the application introduced in Chapter 1.

Chapter 5 covers all other query features; in particular it introduces the notion of ***join*** and ***subquery***.

Chapter 6 revisits the ***data manipulation*** facilities. They comprise all features that change the contents of the database, such as insertions, deletions, and updates. Since set-oriented data manipulation uses query capabilities to specify the set involved, we had to postpone their coverage until this point.

Chapter 7 studies some ***performance*** issues. At first, this chapter title may be somewhat surprising. After all, the scope of the book is the language, and many performance issues are very much linked to the implementation. But there are operations that are intrinsically complex and we want the user to be able to recognize them. Also, some techniques that are used by practically all implementations are somewhat visible at the language level (indexing, locking); they need to be covered here.

Chapter 8 deals with ***views***. A view allows the user to see the data in a way that is different from the way the data are actually stored.

Chapter 9 studies the ***protection of data***. Data are an important asset and must be protected from any corruption or loss, as well as from unauthorized access. Although systems generally try to protect the data without putting any burden on the user, this cannot be done completely without some semantic knowledge that only the user has. This chapter discusses the SQL constructs used to communicate such knowledge to the system.

Part 3 of the book is dedicated to case studies.

Chapter 10 discusses a classical data processing example, with emphasis on simple ***transactions*** and ***batch*** processing. Chapter 11, on the contrary, focuses on ***complex queries***. These two case studies, together with the examples of Part 2 should enable most users to develop straightforward applications.

The next chapters look at more advanced applications which all exhibit some kind of specific characteristics. Together, they enable the reader to build an arsenal of techniques that will be extremely useful in broad areas of application.

Chapters 12 and 13 look at problems which deal with interesting types of data. Chapter 12 is concerned with the management of ***textual information***; Chapter 13 addresses the management of ***object-oriented data***. Object data play a very important role in engineering applications, graphics, and office automation; the chapter looks at how to group the information into highly structured clusters. Although, in these chapters, the techniques are presented in specific application contexts, they are quite general in nature. The most likely situation is that some of these problems will come up when you design real-life applications, even if the context is apparently quite different.

Chapter 14 is concerned with a class of ***graph*** problems where a relationship is defined between instances of the same entity. Such a relationship can be represented by arcs between nodes in a graph. For example, a city A is connected to another city B, and B is connected to C, which is itself connected to D, or A, or B; or a part uses other parts in a ***bill of material*** application. The

Contents

Part 1. Introduction

Chapter 1

Relational Data Management

Any human organization, from the simplest to the most complex, relies on information. In fact, any activity generates information about what happened, where, when, how, and so on. Any piece of information may be expressed as a sentence in a natural language. This representation may be satisfactory for some applications; for others it is not. In fact, for most applications, a better organization consists of representing the information as a collection of well-structured data. Additionally, since data generated today may be important for future activities and decisions, they must be recorded permanently, or at least until explicitly destroyed. The science - or art - of organizing permanent data is called ***data management***.

It is fairly easy to save data on a permanent medium such as magnetic tape or disk. In general, the programming languages such as Cobol, C, and Fortran provide commands to save the contents of program variables on permanent storage and to read them back into memory. However, these commands generally allow the user to save the data in a sequential manner, one data element after another or one record (a collection of data elements) after another. Sometimes, more sophisticated commands allow a program to store data in a way that supports direct access to a record, given the record number. Nevertheless, although such data organizations are extremely useful, they provide only low-level functions, leaving most of the data management problems to the user.

Fortunately, the state of the art in data management has improved tremendously in the last decade. Years of research and development in theoretical as well as practical aspects of data management have yielded impressive results: today, sophisticated languages and systems exist, which simplify the user's job by making it possible to think about data at a much more conceptual level without worrying about a myriad of details.

The most successful of these systems today are based on what is called the ***relational model of data*** and rely on SQL (Structured Query Language). Their success is in part due to the simplicity of the model and the power of the operations available to store, manipulate, retrieve, and control the data. Another reason for their success is the internal sophistication of the systems that support the relational model and SQL in particular; it is this sophistication that enables high-level functions to be executed very efficiently at an affordable cost.

This chapter will guide the reader unfamiliar with data management through four sections. The first section uses an example to illustrate the most important concepts of data management without referring to any computer implementation. The second section introduces the relational model and describes the data of an application in relational terms. The third section goes through some SQL scenarios for data manipulation and queries. Finally, the last section briefly discusses the facilities generally provided by a data management system and by most SQL systems in particular.

1.1 Introducing data management concepts

Let us consider an organization and analyze the data needed to ensure its functioning. The organization is a company which rents tools. The company purchases tools and rents them to customers.

Actions generate the data that need to be kept in order to do business. When a tool is purchased, its existence must be recorded. This is done by maintaining a file of cards, each card recording the information associated with a tool. Since it is important to be able to refer to a particular tool in an unambiguous way, each tool must have a unique name. A simple way of naming a new tool is to give it a sequential number; the first tool receives the number 1, the next tool, 2, and so on. Each tool card with its tool number is stored in the file - let us call it the TOOL file - in order of increasing tool number. (By the way, this makes it very easy to find the next available number since one can just add 1 to the number on the last card in the file.) In addition to the tool number, the tool card will also contain data associated with the tool, such as