SQL
Structured Query Language

SQL
Structured Query Language

2nd Edition

Dr. Carolyn J. Hursch
Dr. Jack L. Hursch

WINDCREST®

Notices

ACCELL/SQL	Unify Corp.
dBASE™ dBASE IV™	Ashton-Tate Corp.
Empress™	Empress Software Inc.
IBM™ DB2™ IMS™ SQL/DS™	International Business Machines Corp.
Informix	*Informix Software, Inc.*
Ingres™	Relational Technology Inc.
Microsoft™	Microsoft Corp.
Novell™ XQL™	Novell Inc.
Oracle™ SQL*Plus™	Oracle Corp.
PARADOX™ Paradox SQL Link™	Borland International
Rdb™ VAX™ SQL VAX™	Digital Equipment Corp.
R:BASE™	Microrim, Inc.
SQLBASE™	Gupta Technologies
XDB	Software Systems, Inc.

SECOND EDITION
FIRST PRINTING

Library of Congress Cataloging-in Publication Data

Hursch, Carolyn J.
Sql, structured query language / by Carolyn J. Hursch and Jack L. Hursch.—2nd ed.
p. cm.
Rev. ed. of: SQL, the structured query language. 1st ed. c1988.
Includes bibliographical references and index.
ISBN 0-8306-8803-X ISBN 0-8306-3803-2 (pbk.)
1. SQL (Computer program language) 2. Data base management. I. Hursch, Jack L. II. Hursch, Carolyn J. SQL, the structured query language. III. Title.
QA76.73.S67H87 1991
005.75'6—dc20 91-2669
CIP

TAB Books offers software for sale. For information and a catalog, please contact TAB Software Department, Blue Ridge Summit, PA 17294-0850.

Book Editor: Patti McCarty
Production: Katherine G. Brown
Book Design: Jaclyn J. Boone
Electronic Pre-press: Terry Hite WP1

Introduction

The widespread implementation of relational databases has brought with it the need for a database language that is user-friendly for the casual user, yet convenient for the programmer and applications builder. The Structured Query Language, SQL (commonly pronounced SEQUEL), after a rapid period of evolution, fills that need. It is easily learned and understood by the end user, and also can be embedded by the programmer in procedural languages such as C, Cobol, or PL/1.

The SQL language provides a much needed common avenue of discourse between the end-user and the programmer. This act alone provides substantial benefit in smoothing out the transition from paper files to computerized database systems, and in the development of applications for existing database systems.

This text sets forth the structure and syntax of SQL, in sufficient detail for those who are not computer professionals to quickly learn and make use of it to create, maintain, and query a relational database of any size. All of the commonly used SQL features are exhibited and worked through step-by-step with examples. Exercises at the end of most chapters make the text appropriate for classroom use.

Because our purpose is to present the complete picture of SQL, its mathematical structure is traced from its basis in first-order logic to its present-day role as a structured query language. The chapters that work through this process are set apart from the detailed description of the SQL language because they will be of interest mainly to computer science professionals.

CHAPTER 1 traces the history of the development of SQL from E. F. Codd's articles in the early 1970s, setting forth the principles of a relational database, through the Chamberlin articles of the late 1970s, to the latest publications of the American National Standards Institute (ANSI) in their efforts to develop a standard SQL language.

CHAPTER 2 sets forth the notation and defines the terms used throughout the book.

CHAPTER 3 contains an overview of all of the components of conventional SQL language, the SQL commands, key words, datatypes and value expressions supported by SQL. The three main types of statements—Data Definition, Data Manipulation, and Data Control—are listed and the syntax for each is shown, as well as the syntax for the various expressions, connectives, predicates, and functions that make up the SQL language.

CHAPTER 4 contains explanations and examples of the use of SQL table expressions and predicates in preparation for using them to set up a database in the next chapter.

CHAPTER 5 illustrates the use of the Data Definition statements CREATE, ALTER, and DROP, and gives examples of these statements as they are used to define and modify tables in an illustrative database system.

CHAPTER 6 illustrates the use of the Data Manipulation commands INSERT, UPDATE, DELETE, and SELECT. Because creating views (unlike creating tables) involves data manipulation, it is included in this chapter.

CHAPTER 7 explains the use of the Data Control statements, by the persons who must administer the database, and discusses the security and integrity constraints that may be invoked using SQL.

CHAPTER 8 shows the use of aggregate functions, logical operators and subqueries as they are employed in SQL, and includes examples of the use of NOT with each of these.

CHAPTER 9 is devoted exclusively to the topic of joins because joins can add a great deal to the efficient use of a relational database. Equijoins and non-equijoins are explained, as well as the Cartesian product, natural joins, and outer joins.

CHAPTER 10 is a detailed discussion of views. With a well-designed database, the end user will be using views most of the time rather than base tables. At the same time, not all operations possible with tables are possible with views. Therefore, SQL operations on views are explained and the view-update problem is discussed. In CHAPTER 10 the use of indices is discussed. This leads to a discussion of the optimal formation of queries to speed up the retrieval process.

CHAPTER 11 discusses the elements of relational algebra and relates them to SQL. SQL has been said to "resemble" the tuple rela-

tional calculus. However, it contains properties taken from both the relational algebra and the relational calculus.

CHAPTER 12 develops the elements of the first-order logic needed for the tuple relational calculus as discussed by E. F. Codd in an early article. Interpretations as developed in the propositional calculus and the predicate calculus with quantifiers are explained, and their extension in the predicate calculus to a database scheme is exhibited. This database scheme is then shown to be an interpretation determining a form of the predicate calculus known as the tuple relational calculus. The extension of the idea of an interpretation is shown to determine what is retrieved by means of a query formulated in the tuple relational calculus. Finally, the examples and exercises show how the tuple relational calculus queries are converted to SQL queries. Thus, chapter 12 demonstrates the theme by which SQL developed from the need for a query language through first-order logic to a user-friendly relational database interface.

CHAPTER 13 presents "embedded SQL," a form of SQL that can be embedded in computer programs and converted to host language code with a precompiler. Cursors (implementor-defined work areas for holding relational tables obtained from queries) are discussed, as well as the embedded SQL commands for manipulating cursors. The goal in chapter 13 is to present embedded SQL as it now exists under the current ANSI Standard and most current implementations without limiting the presentation to any one implementation. Suggestions are made for modifying the embedding process to make the programmers task less onerous and more likely to produce bug-free programs. These suggestions are demonstrated by examples of SQL embedded in short C language modules, which programmers can modify to fit their own implementation and requirements.

CHAPTER 14 describes the use of SQL in some of the principal commercial relational databases and SQL servers now on the market.

CHAPTER 15 describes the requirements that must be met by a commercial database in order to claim conformance with the ANSI standard.

A Glossary defines all words relevant to SQL, as well as all relational database terms that the reader might need to know in connection with this topic.

A Bibliography provides the interested reader with additional sources of information on all of the topics covered.

Contents

Introduction *ix*

1 How SQL got here *1*

Roots 2
Fully relational databases 3
Current status 5
Summary 6

2 Notation and definitions *7*

Notation 7
Definitions 8
- Relations 8
- Relation schemes 9
- Databases 9
- Tables 10
- Virtual tables (views) 10
- Columns 11
- Rows 11
- Data values 12
- Target list 12
- NULL values 12
- The NULL Value function 13
- Reserved words 14
- Database objects 14
- Result tables 14

Summary 14

3 The components of SQL 15

The catalog 15
SQL commands 16
Reserved words 16
Datatypes 17
Value expressions 18
Logical connectives 18
Predicates 19
The Data Definition Language (DDL) 19
- Creating tables 19
- Creating an index 20
- Altering tables 20
- Dropping tables 20
- Dropping indices 21

The Data Manipulation Language (DML) 21
- Inserting 21
- Updating 22
- Deleting 22
- Retrieving (using SELECT) 23
- Creating views 23

The Data Control Language (DCL) 24
Table expressions (clauses) 24
Aggregate functions 25
Subqueries 25
Summary 26

4 Table expressions and predicates 27

Table expressions (clauses) 27
- FROM clause 28
- WHERE clause 28
- GROUP BY clause 29
- HAVING clause 30
- ORDER BY clause 31

Predicates 32
- Comparison predicate 32
- BETWEEN predicate 33
- IN (or NOT IN) predicate 34
- LIKE (or NOT LIKE) predicate 36
- NULL predicate 37
- Quantified predicates ALL, SOME and ANY 38
- EXISTS predicate 39

Summary 40
Chapter 4 exercises 40
- Answers to chapter 4 exercises 40

5 Using data definition statements 43

CREATE DATABASE 43
CREATE SCHEMA 44
CREATE STORAGE AREA 44
CREATE TABLE 45
CREATE DOMAIN 50
CREATE SYNONYM 50
CREATE VIEW 51
CREATE INDEX 51
ALTER TABLE 51
COPY 53
DROP DATABASE 53
DROP SCHEMA 53
DROP STORAGE AREA 54
DROP TABLE 54
DROP DOMAIN 54
DROP SYNONYM 54
DROP VIEW 55
DROP INDEX 55
Summary 55
Chapter 5 exercises 56
Answers to chapter 5 exercises 56

6 Using data manipulation statements 59

INSERT 59
Inserting part or all of a single row 60
Inserting multiple rows, or parts of multiple rows 62
UPDATE 63
DELETE 64
SELECT 65
CREATE VIEW 66
Summary 67
Chapter 6 exercises 67
Answers to chapter 6 exercises 68

7 Using data control statements 71

Access control 71
GRANT 71
REVOKE 74
Views as security devices 75
Integrity control 75
COMMIT 75
BEGIN, START, and SET TRANSACTION 76
ROLLBACK 77
Summary 78
Chapter 7 exercises 78
Answers to chapter 7 exercises 78

8 *Logical connectives, aggregate functions, and subqueries* 81

Logical connectives 81
AND (INTERSECTION) 81
OR (UNION) 82
Using AND and OR in the same query 83
MINUS (Difference) 84
Substituting IN and NOT IN for AND, OR or MINUS 84
Aggregate functions 86
AVERAGE (AVG) 88
COUNT 89
COUNT(*) 89
SUM 89
MAX and MIN 90
Aliases (correlation names) 90
Column aliases 91
Table aliases 91
Subqueries (SubSELECTs or nested SELECTS) 92
Subqueries that select multiple columns 93
Subqueries using EXISTS 94
Subqueries using ANY or ALL 94
Summary 95
Chapter 8 exercises 95
Answers to chapter 8 exercises 96

9 *Joins* 99

Equijoins 100
The cartesian product 102
The natural join 103
Join on specified columns only 103
Non-equijoins 105
Additional conditions in join queries 106
Joining more than two tables 106
Joining tables to views 107
Creating a view from a join 107
Joining views to views 107
Joining a table with itself 108
Outer joins 108
Summary 109
Chapter 9 exercises 110
Answers to chapter 9 exercises 111

10 *Views, indices, and queries* 113

Views 113
Creating a view 114

Views on multiple tables 116
Joining a view to another view or table 117
Expressions and functions in views 117
Aggregate functions in views 117
Updating rows in views 118
Inserting rows into views 119
Deleting rows from views 119
Using views to restrict table access 119
Using indices to optimize performance 120
Indices and keys 120
Unique indices 121
Indices on multiple columns 121
Optimizing queries 122
Summary 122
Chapter 10 exercises 122
Answers to chapter 10 exercises 123

11 Relational algebra and SQL *125*

Relational definitions 125
Boolean operators 126
Projection operator 127
Select operator 128
Join operator 128
Division of relations, lossy joins 129
Remarks 131
Summary 131
Chapter 11 exercises 131
Answers to chapter 11 exercises 132

12 Logic and SQL *135*

Relational definitions 136
Formal theory 136
Interpretations 137
Propositional calculus 137
Predicates and quantifiers 140
Free and bound variables 141
Interpretations for predicate calculus 142
Tuple relational calculus 143
Simple alpha expressions 145
Alpha expressions 145
Converting alpha expressions to SQL expressions 147
Summary 148
Chapter 12 exercises 148
Answers to chapter 12 exercises 150

13 Embedded SQL *155*

Recognizing embedded SQL statements 157
A first look at embedded SQL 157
Host variables in embedded SQL 157
Variable declaration 158
EXEC SQL INCLUDE 159
SQL commands in embedded SQL 159
SELECT 159
UPDATE 160
DELETE 160
INSERT 160
Other SQL commands 160
Cursors in embedded SQL 161
SQLCODE 163
Using cursors to update and delete 169
SQLCODE revisited 170
Summary 170

14 SQL at work *171*

System R 171
Information Management System (IMS) 172
SQL/Data System (SQL/DS) 172
Database 2 (DB2) 173
ACCELL/SQL 173
dBASE IV 173
Empress 174
Informix 174
Ingres 174
ORACLE 175
R:BASE 175
Rdb 176
SQLBASE 176
XDB 176
XQL 177
SQL servers 177
Paradox SQL Link 177
SQL as a knowledge-base query language 178
Summary 178

15 ANSI conformance requirements *179*

Glossary *183*

Bibliography *191*

Index *197*

1
How SQL got here

Since the first edition of this book was published, the popularity of Structured Query Language, SQL (commonly pronounced "sequel"), has continued to grow at a rapid rate. Even more impressive is the variety of innovations in the SQL language brought forth by the vast array of relational database management systems now on the market.

The original standards set forth by the American National Standards Institute (ANSI) have now been augmented by new publications (ANSI X3.135-1989 and ANSI X3.168-1989), and the ANSI staff and associates continue to work toward further refinements of terms and concepts. Important as this effort is, there actually exist two standards today: (1) the barebones, carefully crafted ANSI Standard SQL and (2) the looser, yet broader industry standard based on IBM's mainframe database system DB2. Most SQL database systems on the market today are based in whole or in part on DB2, which includes all features of the ANSI standard.

Therefore, this edition will set forth the basic ANSI SQL as it exists at this time, augmented by the most commonly used enhancements found in commercial relational database systems. Because different vendors may use different names for the same thing, we will note the corresponding term used by individual companies from a list of popular vendors wherever there is a large difference in terms for a similar operation or concept. The list of vendors can never be conclusive. At this writing we are aware of several now in development which we cannot yet include—and there will always be more. Rather, our treatment will be representative of what is now available rather than exhaustive, and in that sense will offer a contrast in the different approaches being used to make databases accessible.

The diversity of additions to the SQL language points up the fact that different vendors have taken different routes to the solution of problems arising out of the use of early versions of SQL. Certain problems, such as the use of null values, unsolved in the original ANSI version, and still under discussion and evaluation by that organization, have been considered, approached, circumvented, allowed for, and perhaps solved by software manufacturers intent on supplying users with a complete method for managing their data. Depending on the purposes of the user, one or another of these approaches might be sufficient until such time as the perfect solution is reached and promulgated.

Therefore, while the immediate purpose of this revised edition is to update the ANSI version of SQL that we presented earlier, a broader purpose is to show the many ways in which the ingenuity of various vendors has enlarged and expanded its usage.

To make the reader's life easier, a consistent set of symbols will be used throughout this text, regardless of those used by the vendors. In some cases, our symbols and terms will agree with those of a specific vendor; sometimes not. (For example, some database systems refer to their restricted list of words as "key words"; some call this list "reserved words." We will call them all "reserved words" here.)

Our definitions then, will indicate the ANSI Standard meaning of terms, but other meanings and other terms not in ANSI although in common use in industry, will be included.

ANSI has now set up "conformance standards" by which a database can be judged. These will form a checklist as to whether or not the database is in accord with ANSI Standard SQL. In response to this, vendors now make claims in their advertising about whether their version conforms to this standard and at what level. ANSI conformance requirements are discussed in chapter 15.

Roots

Here is a brief history of the development of the SQL data sublanguage. When E. F. Codd introduced the concept of a relational database in 1970, he suggested that "the adoption of a relational model of data . . . permits the development of a universal data sublanguage based on an applied predicate calculus." Although he indicated the requirements and the advantages of such a language, he did not attempt at that time to devise one. In a later article, he discussed the concept of "relational completeness" (a term which he coined, and which is now widely used) of a database sublanguage.

Acceptance of the relational idea was relatively slow (but only in

comparison with the usual speed of technical advances in the computer field). Therefore, it was not until 1974 that Chamberlin and Boyce published an article suggesting the form of a structured query language which at that time, was called SEQUEL. The following year, Boyce, Chamberlin, King and Hammer published an article setting forth the sublanguage SQUARE which was much like SEQUEL except that SQUARE used mathematical expressions rather than the English terms of SEQUEL. Both languages are shown by the authors to be "relationally complete" in the sense outlined by Codd in his 1970 and 1972 articles. "Relationally complete" in that context means "at least as powerful as the tuple relational calculus."

The SQUARE article was followed by another article by Chamberlin and others in 1976 when the name was changed to SEQUEL 2, and it was used as the query language for the IBM research database System R.

By the time Chamberlin wrote in 1980, summarizing user experience with the language, the name had been changed to its present form: SQL, denoting a "Structured Query Language." A test of SQL by a broad group of users resulted in several changes to the language including the addition of "outer joins" to SQL's capabilities, which Codd had already suggested in his 1979 article. Further development reported in other articles resulted in present-day SQL. (See the Bibliography for references for all publications mentioned.)

During the last decade, relational databases have emerged and become more popular than the hierarchical and network databases that preceded them. This trend appears to be accelerating.

Fully relational databases

To inject some order into the rapidly increasing literature on relational databases, Codd in 1985 laid down 12 principles, at least six of which must be satisfied in order for a database to call itself "fully relational." These were preceded by one overall general rule, called "Rule Zero" as follows:

Rule 0 Relational database management. For any system that is advertised as, or claimed to be, a relational database management system, that system must be able to manage databases entirely through its relational capabilities.

The essence of the 12 specific rules is as follows:

1. *Representation of information.* All information in a relational database is represented explicitly at the logical level and in exactly one way—by values in tables.

2. *Guaranteed logical accessibility.* Each and every datum (atomic value) in a relational database is guaranteed to be logically accessible by resorting to a combination of table name, primary key value and column name.
3. *Systematic representation of missing information.* Null values (distinct from the empty character string or a string of blank characters and distinct from zero or any other number) are supported in a fully relational DBMS for representing missing information and inapplicable information in a systematic way independent of datatype.
4. *Dynamic online catalog.* The database description is represented at the local level in the same way as ordinary data, so that authorized users can query it in the same relational language that they use in working with the regular data.
5. *Comprehensive data sublanguage.* A relational system may support several languages and various modes of terminal use (for example, the fill-in-the-blanks mode). However, there must be at least one language whose statements are expressible, in some well-defined syntax, as character strings. Also, it must be comprehensive in supporting all of the following items:

   - Data definition
   - View definition
   - Data manipulation (interactive and by program)
   - Integrity constraints
   - Authorization
   - Transaction boundaries (begin, commit and rollback)

6. *Updatable views.* All views that are theoretically updatable are also updatable by the database system.
7. *High-level insert, update, and delete.* The capability of handling a base relation or a derived relation as a single operand applies not only to the retrieval of data but also to the insertion, update and deletion of data.
8. *Physical data independence.* Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representations or access methods.
9. *Logical data independence.* Application programs and terminal activities remain logically unimpaired when information preserving appropriate changes of any kind are made to the base tables.
10. *Integrity independence.* Integrity constraints specific to a particular database must be definable in the relational data

sublanguage and storable in the catalog, not in the application program.

11. *Distribution independence.* Whether or not a system supports database distribution, it must have a data sublanguage that can support distributed databases without impairing the application programs or terminal activities.
12. *Nonsubversion.* If a relational system has a low-level (single-record-at-a-time) language, that low-level language cannot be used to subvert or bypass the integrity rules and constraints expressed in the higher level relational language (multiple-records-at-a time).

A proof that SQL is relationally complete has been outlined by Date (p. 276 of Date's *An Introduction to Database Systems*, Vol. 1).

Chapter 12 of this book gives a discussion of the tuple relational calculus beginning with classical logic. In this development we demonstrate the conversion of logic queries to SQL.

The fact that the developers knew ahead of time what SQL should be and what it would be required to do, gave it a strong theoretical foundation. This is probably a first in computer language development because most computer languages in use today are the result of a basic idea supplemented by a great deal of ad hoc patching to meet problems as they arise. This fact—of specifying the need for SQL before developing the mechanics of it—gave rise to an elegantly parsimonious language consisting of relatively few commands that can be used to satisfy most of the needs of a very complex database.

Its simplicity makes SQL convenient for the casual user as well as the sophisticated developer. It can be used for ad hoc queries, and, it also can be embedded in a host-language program.

At this point, there are several structured languages in existence that are being used for querying relational databases. However, SQL appears to be the one most widely adopted for commercial use.

Current status

The American National Standards Institute (ANSI) has published a standard, called Database Language SQL, setting forth the minimal syntax and semantics for SQL. (The ANSI Standard uses the Backus-Nauer Form (BNF) of syntactic notation which, for the reader's convenience, is not used here. See Nauer, P. in the Bibliography.)

In its most recent publication (X3.135-1989), ANSI added an "integrity enhancement feature" to its original SQL standard. This feature demands that a database have some means for specifying