

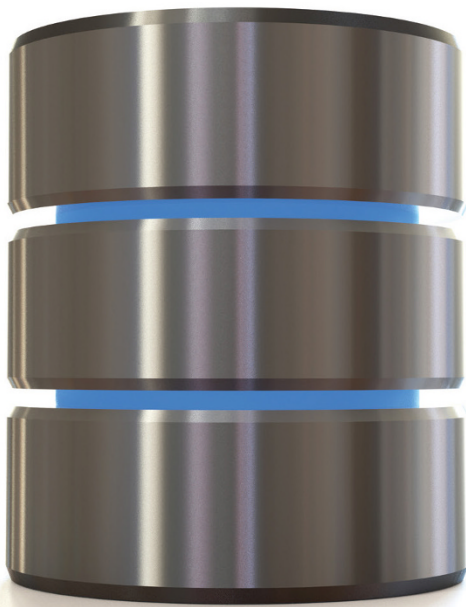
LEARNING MADE EASY



9th Edition

# SQL

for  
**dummies**<sup>®</sup>  
A Wiley Brand



Learn all the features  
in the latest version of SQL

Build a relational database and  
create a management system

Secure and protect your  
database from corruption

**Allen G. Taylor**

Bestselling author of all previous  
editions of *SQL For Dummies*





# SQL

9th Edition

**by Allen G. Taylor**

Author of SQL All-in-One For Dummies

for  
**dummies**<sup>®</sup>  
A Wiley Brand

## SQL For Dummies® 9th Edition

Published by: **John Wiley & Sons, Inc.**, 111 River Street, Hoboken, NJ 07030-5774, [www.wiley.com](http://www.wiley.com)

Copyright © 2019 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Trademarks:** Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit <https://hub.wiley.com/community/support/dummies>.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit [www.wiley.com](http://www.wiley.com).

Library of Congress Control Number: 2018960776

ISBN: 978-1-119-52707-7 (pbk); ISBN: 978-1-119-52708-4 (ePDF); ISBN: 978-1-119-52709-1 (ePub)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

# Contents at a Glance

---

<b>Introduction</b> .....	1
<b>Part 1: Getting Started with SQL</b> .....	5
CHAPTER 1: Relational Database Fundamentals .....	7
CHAPTER 2: SQL Fundamentals .....	23
CHAPTER 3: The Components of SQL .....	55
<b>Part 2: Using SQL to Build Databases</b> .....	83
CHAPTER 4: Building and Maintaining a Simple Database Structure .....	85
CHAPTER 5: Building a Multi-table Relational Database .....	109
<b>Part 3: Storing and Retrieving Data</b> .....	141
CHAPTER 6: Manipulating Database Data .....	143
CHAPTER 7: Handling Temporal Data .....	163
CHAPTER 8: Specifying Values .....	179
CHAPTER 9: Using Advanced SQL Value Expressions .....	209
CHAPTER 10: Zeroing In on the Data You Want .....	223
CHAPTER 11: Using Relational Operators .....	259
CHAPTER 12: Delving Deep with Nested Queries .....	283
CHAPTER 13: Recursive Queries .....	303
<b>Part 4: Controlling Operations</b> .....	313
CHAPTER 14: Providing Database Security .....	315
CHAPTER 15: Protecting Data .....	331
CHAPTER 16: Using SQL within Applications .....	351
<b>Part 5: Taking SQL to the Real World</b> .....	365
CHAPTER 17: Accessing Data with ODBC and JDBC .....	367
CHAPTER 18: Operating on XML Data with SQL .....	377
CHAPTER 19: SQL and JSON .....	399

<b>Part 6: Advanced Topics</b> .....	413
CHAPTER 20: Stepping through a Dataset with Cursors .....	415
CHAPTER 21: Adding Procedural Capabilities with Persistent Stored Modules . . . .	427
CHAPTER 22: Handling Errors .....	445
CHAPTER 23: Triggers .....	457
<b>Part 7: The Parts of Tens</b> .....	463
CHAPTER 24: Ten Common Mistakes .....	465
CHAPTER 25: Ten Retrieval Tips .....	469
<b>Appendix: ISO/IEC SQL: 2016 Reserved Words</b> .....	473
<b>Index</b> .....	479

# Table of Contents

---

<b>INTRODUCTION</b> .....	1
About This Book .....	1
Foolish Assumptions .....	2
Icons Used in This Book .....	2
Beyond the Book .....	3
Where to Go from Here .....	3
<b>PART 1: GETTING STARTED WITH SQL</b> .....	5
<b>CHAPTER 1: Relational Database Fundamentals</b> .....	7
Keeping Track of Things .....	8
What Is a Database? .....	9
Database Size and Complexity .....	10
What Is a Database Management System? .....	10
Flat Files .....	12
Database Models .....	13
Relational model .....	13
Components of a relational database .....	14
Dealing with your relations .....	14
Enjoy the view .....	16
Schemas, domains, and constraints .....	18
The object model challenged the relational model .....	19
The object-relational model .....	20
Database Design Considerations .....	20
<b>CHAPTER 2: SQL Fundamentals</b> .....	23
What SQL Is and Isn't .....	23
A (Very) Little History .....	25
SQL Statements .....	26
Reserved Words .....	28
Data Types .....	28
Exact numerics .....	29
Approximate numerics .....	31
Character strings .....	33
Binary strings .....	35
Booleans .....	36
Datetimes .....	36
Intervals .....	38
XML type .....	38
ROW types .....	41
Collection types .....	42

REF types . . . . .	44
User-defined types . . . . .	44
Data type summary . . . . .	48
Null Values . . . . .	49
Constraints . . . . .	50
Using SQL in a Client/Server System . . . . .	50
The server . . . . .	51
The client . . . . .	52
Using SQL on the Internet or an Intranet . . . . .	52
<b>CHAPTER 3: The Components of SQL . . . . .</b>	<b>55</b>
Data Definition Language . . . . .	56
When “just do it!” is not good advice . . . . .	56
Creating tables . . . . .	57
A room with a view . . . . .	59
Collecting tables into schemas . . . . .	64
Ordering by catalog . . . . .	65
Getting familiar with DDL statements . . . . .	66
Data Manipulation Language . . . . .	68
Value expressions . . . . .	68
Predicates . . . . .	72
Logical connectives . . . . .	73
Set functions . . . . .	73
Subqueries . . . . .	76
Data Control Language . . . . .	76
Transactions . . . . .	76
Users and privileges . . . . .	77
Referential integrity constraints can jeopardize your data . . . . .	80
Delegating responsibility for security . . . . .	82
<b>PART 2: USING SQL TO BUILD DATABASES . . . . .</b>	<b>83</b>
<b>CHAPTER 4: Building and Maintaining a Simple Database Structure . . . . .</b>	<b>85</b>
Using a RAD Tool to Build a Simple Database . . . . .	86
Deciding what to track . . . . .	86
Creating a database table . . . . .	87
Altering the table structure . . . . .	93
Creating an index . . . . .	95
Deleting a table . . . . .	97
Building POWER with SQL’s DDL . . . . .	98
Using SQL with Microsoft Access . . . . .	99
Creating a table . . . . .	101
Creating an index . . . . .	105



	Altering the table structure .....	105
	Deleting a table .....	106
	Deleting an index .....	106
	Portability Considerations .....	107
<b>CHAPTER 5:</b>	<b>Building a Multi-table Relational Database .....</b>	<b>109</b>
	Designing a Database .....	110
	Step 1: Defining objects .....	110
	Step 2: Identifying tables and columns .....	110
	Step 3: Defining tables .....	111
	Domains, character sets, collations, and translations .....	115
	Getting into your database fast with keys .....	116
	Working with Indexes .....	119
	What's an index, anyway? .....	119
	Why you should want an index .....	121
	Maintaining an index .....	121
	Maintaining Data Integrity .....	122
	Entity integrity .....	122
	Domain integrity .....	124
	Referential integrity .....	124
	Just when you thought it was safe .....	127
	Potential problem areas .....	128
	Constraints .....	130
	Normalizing the Database .....	134
	Modification anomalies and normal forms .....	134
	First normal form .....	136
	Second normal form .....	137
	Third normal form .....	138
	Domain-key normal form (DK/NF) .....	139
	Abnormal form .....	140
	<b>PART 3: STORING AND RETRIEVING DATA .....</b>	<b>141</b>
<b>CHAPTER 6:</b>	<b>Manipulating Database Data .....</b>	<b>143</b>
	Retrieving Data .....	144
	Creating Views .....	145
	From tables .....	146
	With a selection condition .....	147
	With a modified attribute .....	148
	Updating Views .....	149
	Adding New Data .....	150
	Adding data one row at a time .....	151
	Adding data only to selected columns .....	152
	Adding a block of rows to a table .....	152

	Updating Existing Data . . . . .	155
	Transferring Data . . . . .	158
	Deleting Obsolete Data . . . . .	161
<b>CHAPTER 7:</b>	<b>Handling Temporal Data . . . . .</b>	<b>163</b>
	Understanding Times and Periods . . . . .	164
	Working with Application-Time Period Tables . . . . .	165
	Designating primary keys in application-time period tables. . . . .	168
	Applying referential integrity constraints to application-time period tables. . . . .	169
	Querying application-time period tables . . . . .	170
	Working with System-Versioned Tables . . . . .	171
	Designating primary keys in system-versioned tables. . . . .	173
	Applying referential integrity constraints to system-versioned tables. . . . .	174
	Querying system-versioned tables . . . . .	174
	Tracking Even More Time Data with Bitemporal Tables . . . . .	175
	Formatting and Parsing Dates and Times . . . . .	176
<b>CHAPTER 8:</b>	<b>Specifying Values . . . . .</b>	<b>179</b>
	Values . . . . .	179
	Row values . . . . .	180
	Literal values . . . . .	180
	Variables . . . . .	182
	Special variables . . . . .	184
	Column references . . . . .	185
	Value Expressions . . . . .	186
	String value expressions . . . . .	186
	Numeric value expressions . . . . .	187
	Datetime value expressions . . . . .	187
	Interval value expressions . . . . .	188
	Conditional value expressions . . . . .	189
	Functions . . . . .	189
	Set functions . . . . .	189
	Value functions . . . . .	193
	Table functions . . . . .	208
<b>CHAPTER 9:</b>	<b>Using Advanced SQL Value Expressions . . . . .</b>	<b>209</b>
	CASE Conditional Expressions . . . . .	210
	Using CASE with search conditions . . . . .	211
	Using CASE with values . . . . .	212
	A special CASE — NULLIF . . . . .	215
	Another special CASE — COALESCE . . . . .	216

CAST Data-Type Conversions . . . . .	217
Using CAST within SQL . . . . .	219
Using CAST between SQL and the host language . . . . .	220
Row Value Expressions . . . . .	221
<b>CHAPTER 10: Zeroing In on the Data You Want . . . . .</b>	<b>223</b>
Modifying Clauses . . . . .	224
FROM Clauses . . . . .	225
WHERE Clauses . . . . .	226
Comparison predicates . . . . .	227
BETWEEN . . . . .	228
IN and NOT IN . . . . .	229
LIKE and NOT LIKE . . . . .	231
SIMILAR . . . . .	232
NULL . . . . .	232
ALL, SOME, ANY . . . . .	234
EXISTS . . . . .	236
UNIQUE . . . . .	237
DISTINCT . . . . .	238
OVERLAPS . . . . .	238
MATCH . . . . .	239
Referential integrity rules and the MATCH predicate . . . . .	240
Logical Connectives . . . . .	243
AND . . . . .	243
OR . . . . .	244
NOT . . . . .	244
GROUP BY Clauses . . . . .	245
HAVING Clauses . . . . .	247
ORDER BY Clauses . . . . .	248
Limited FETCH . . . . .	250
Peering through a Window to Create a Result Set . . . . .	251
Partitioning a window into buckets with NTILE . . . . .	252
Navigating within a window . . . . .	253
Nesting window functions . . . . .	255
Evaluating groups of rows . . . . .	256
Row pattern recognition . . . . .	257
<b>CHAPTER 11: Using Relational Operators . . . . .</b>	<b>259</b>
UNION . . . . .	259
The UNION ALL operation . . . . .	261
The CORRESPONDING operation . . . . .	262
INTERSECT . . . . .	262
EXCEPT . . . . .	264

Join Operators . . . . .	265
Basic join . . . . .	265
Equi-join . . . . .	267
Cross join . . . . .	269
Natural join . . . . .	270
Condition join . . . . .	270
Column-name join . . . . .	271
Inner join . . . . .	272
Outer join . . . . .	272
Union join . . . . .	276
ON versus WHERE . . . . .	282
<b>CHAPTER 12: Delving Deep with Nested Queries . . . . .</b>	<b>283</b>
What Subqueries Do . . . . .	285
Nested queries that return sets of rows . . . . .	285
Nested queries that return a single value . . . . .	289
The ALL, SOME, and ANY quantifiers . . . . .	292
Nested queries that are an existence test . . . . .	293
Other correlated subqueries . . . . .	295
UPDATE, DELETE, and INSERT . . . . .	299
Retrieving changes with pipelined DML . . . . .	301
<b>CHAPTER 13: Recursive Queries . . . . .</b>	<b>303</b>
What Is Recursion? . . . . .	303
Houston, we have a problem. . . . .	305
Failure is not an option. . . . .	305
What Is a Recursive Query? . . . . .	306
Where Might You Use a Recursive Query? . . . . .	306
Querying the hard way . . . . .	308
Saving time with a recursive query . . . . .	309
Where Else Might You Use a Recursive Query? . . . . .	311
<b>PART 4: CONTROLLING OPERATIONS . . . . .</b>	<b>313</b>
<b>CHAPTER 14: Providing Database Security . . . . .</b>	<b>315</b>
The SQL Data Control Language . . . . .	316
User Access Levels . . . . .	316
The database administrator . . . . .	317
Database object owners . . . . .	317
The public . . . . .	318
Granting Privileges to Users . . . . .	318
Roles . . . . .	320
Inserting data . . . . .	320
Looking at data . . . . .	321

Modifying table data . . . . .	321
Deleting obsolete rows from a table . . . . .	322
Referencing related tables . . . . .	322
Using domains . . . . .	323
Causing SQL statements to be executed . . . . .	325
Granting Privileges across Levels . . . . .	325
Granting the Power to Grant Privileges . . . . .	327
Taking Privileges Away . . . . .	328
Using GRANT and REVOKE Together to Save Time and Effort . . . . .	329
<b>CHAPTER 15: Protecting Data . . . . .</b>	<b>331</b>
Threats to Data Integrity . . . . .	332
Platform instability . . . . .	332
Equipment failure . . . . .	332
Concurrent access . . . . .	333
Reducing Vulnerability to Data Corruption . . . . .	336
Using SQL transactions . . . . .	336
The default transaction . . . . .	338
Isolation levels . . . . .	338
The implicit transaction-starting statement . . . . .	341
SET TRANSACTION . . . . .	341
COMMIT . . . . .	342
ROLLBACK . . . . .	342
Locking database objects . . . . .	343
Backing up your data . . . . .	343
Savepoints and subtransactions . . . . .	344
Constraints Within Transactions . . . . .	345
Avoiding SQL Injection Attacks . . . . .	350
<b>CHAPTER 16: Using SQL within Applications . . . . .</b>	<b>351</b>
SQL in an Application . . . . .	352
Keeping an eye out for the asterisk . . . . .	352
SQL strengths and weaknesses . . . . .	353
Procedural languages' strengths and weaknesses . . . . .	353
Problems in combining SQL with a procedural language . . . . .	353
Hooking SQL into Procedural Languages . . . . .	354
Embedded SQL . . . . .	355
Module language . . . . .	358
Object-oriented RAD tools . . . . .	360
Using SQL with Microsoft Access . . . . .	361

<b>PART 5: TAKING SQL TO THE REAL WORLD</b> .....	365
<b>CHAPTER 17: Accessing Data with ODBC and JDBC</b> .....	367
ODBC .....	368
The ODBC interface .....	368
Components of ODBC .....	369
ODBC in a Client/Server Environment .....	370
ODBC and the Internet .....	370
Server extensions .....	371
Client extensions .....	372
ODBC and an Intranet .....	373
JDBC .....	373
<b>CHAPTER 18: Operating on XML Data with SQL</b> .....	377
How XML Relates to SQL .....	377
The XML Data Type .....	378
When to use the XML type .....	379
When not to use the XML type .....	380
Mapping SQL to XML and XML to SQL .....	380
Mapping character sets .....	381
Mapping identifiers .....	381
Mapping data types .....	382
Mapping tables .....	382
Handling null values .....	383
Generating the XML Schema .....	384
SQL Functions That Operate on XML Data .....	385
XMLDOCUMENT .....	385
XMLELEMENT .....	385
XMLFOREST .....	386
XMLCONCAT .....	386
XMLAGG .....	387
XMLCOMMENT .....	388
XMLPARSE .....	388
XMLPI .....	388
XMLQUERY .....	389
XMLCAST .....	389
Predicates .....	390
DOCUMENT .....	390
CONTENT .....	390
XMLEXISTS .....	390
VALID .....	391

Transforming XML Data into SQL Tables . . . . .	392
Mapping Non-Predefined Data Types to XML . . . . .	393
Domain . . . . .	393
Distinct UDT . . . . .	394
Row . . . . .	395
Array . . . . .	396
Multiset . . . . .	397
The Marriage of SQL and XML . . . . .	398
<b>CHAPTER 19: SQL and JSON . . . . .</b>	<b>399</b>
Using JSON with SQL . . . . .	400
Ingesting and storing JSON data into a relational database . . . . .	400
Generating JSON data from relational data . . . . .	400
Querying JSON data stored in relational tables . . . . .	400
The SQL/JSON Data Model . . . . .	401
SQL/JSON items . . . . .	401
SQL/JSON sequences . . . . .	402
Parsing JSON . . . . .	402
Serializing JSON . . . . .	402
SQL/JSON Functions . . . . .	403
JSON API common syntax . . . . .	403
Query functions . . . . .	404
Constructor functions . . . . .	408
IS JSON predicate . . . . .	411
JSON nulls and SQL nulls . . . . .	411
SQL/JSON Path Language . . . . .	411
There's More . . . . .	412
<b>PART 6: ADVANCED TOPICS . . . . .</b>	<b>413</b>
<b>CHAPTER 20: Stepping through a Dataset with Cursors . . . . .</b>	<b>415</b>
Declaring a Cursor . . . . .	416
Query expression . . . . .	417
ORDER BY clause . . . . .	417
Updatability clause . . . . .	419
Sensitivity . . . . .	419
Scrollability . . . . .	420
Opening a Cursor . . . . .	421
Fetching Data from a Single Row . . . . .	422
Syntax . . . . .	423
Orientation of a scrollable cursor . . . . .	424
Positioned DELETE and UPDATE statements . . . . .	424
Closing a Cursor . . . . .	425

<b>CHAPTER 21: Adding Procedural Capabilities with Persistent Stored Modules</b> .....	427
Compound Statements .....	428
Atomicity.....	429
Variables.....	430
Cursors .....	430
Conditions.....	431
Handling conditions .....	431
Conditions that aren't handled .....	434
Assignment.....	434
Flow of Control Statements .....	435
IF...THEN...ELSE...END IF .....	435
CASE...END CASE .....	435
LOOP...ENDLOOP .....	437
LEAVE.....	437
WHILE...DO...END WHILE .....	438
REPEAT...UNTIL...END REPEAT .....	438
FOR...DO...END FOR.....	439
ITERATE .....	439
Stored Procedures.....	440
Stored Functions .....	442
Privileges.....	442
Stored Modules .....	443
<b>CHAPTER 22: Handling Errors</b> .....	445
SQLSTATE .....	445
WHENEVER Clause.....	447
Diagnostics Areas .....	448
Diagnostics header area.....	449
Diagnostics detail area .....	450
Constraint violation example.....	452
Adding constraints to an existing table .....	453
Interpreting the information returned by SQLSTATE.....	454
Handling Exceptions .....	455
<b>CHAPTER 23: Triggers</b> .....	457
Examining Some Applications of Triggers .....	457
Creating a Trigger .....	458
Statement and row triggers.....	459
When a trigger fires.....	459
The triggered SQL statement.....	459
An example trigger definition .....	460
Firing a Succession of Triggers .....	460
Referencing Old Values and New Values .....	461
Firing Multiple Triggers on a Single Table .....	462



<b>PART 7: THE PARTS OF TENS</b> .....	463
<b>CHAPTER 24: Ten Common Mistakes</b> .....	465
Assuming That Your Clients Know What They Need .....	465
Ignoring Project Scope .....	466
Considering Only Technical Factors .....	466
Not Asking for Client Feedback .....	466
Always Using Your Favorite Development Environment .....	467
Using Your Favorite System Architecture Exclusively .....	467
Designing Database Tables in Isolation .....	467
Neglecting Design Reviews .....	468
Skipping Beta Testing .....	468
Not Documenting Your Process .....	468
<b>CHAPTER 25: Ten Retrieval Tips</b> .....	469
Verify the Database Structure .....	470
Try Queries on a Test Database .....	470
Double-Check Queries That Include Joins .....	470
Triple-Check Queries with Subselects .....	470
Summarize Data with GROUP BY .....	471
Watch GROUP BY Clause Restrictions .....	471
Use Parentheses with AND, OR, and NOT .....	471
Control Retrieval Privileges .....	472
Back Up Your Databases Regularly .....	472
Handle Error Conditions Gracefully .....	472
<b>APPENDIX: ISO/IEC SQL: 2016 RESERVED WORDS</b> .....	473
<b>INDEX</b> .....	479



# Introduction

---

Welcome to database development using SQL, the industry-standard database query language. Many database management system (DBMS) tools run on a variety of hardware platforms. The differences among the tools can be great, but all serious products have one thing in common: They support SQL data access and manipulation. If you know SQL, you can build relational databases and get useful information out of them.

## About This Book

---

Relational database management systems are vital to many organizations. People often think that creating and maintaining these systems must be extremely complex activities — the domain of database gurus who possess enlightenment beyond that of mere mortals. This book sweeps away the database mystique. In this book, you

- » Get to the roots of databases.
- » Find out how a DBMS is structured.
- » Discover the major functional components of SQL.
- » Build a database.
- » Protect a database from harm.
- » Operate on database data.
- » Determine how to get the information you want out of a database.

The purpose of this book is to help you build relational databases and get valuable information out of them by using SQL. SQL is the international standard language used to create and maintain relational databases. This edition covers the latest version of the standard, SQL:2016.

This book doesn't tell you how to design a database (I do that in *Database Development For Dummies*, also published by Wiley). Here I assume that you or somebody else has already created a valid design. I then illustrate how you implement that

design by using SQL. If you suspect that you don't have a good database design, then by all means fix your design before you try to build the database. The earlier you detect and correct problems in a development project, the cheaper the corrections will be.

## Foolish Assumptions

If you need to store or retrieve data from a DBMS, you can do a much better job with a working knowledge of SQL. You don't need to be a programmer to use SQL, and you don't need to know programming languages, such as Java, C, or BASIC. SQL's syntax is like that of English. If you *are* a programmer, you can incorporate SQL into your programs. SQL adds powerful data manipulation and retrieval capabilities to conventional languages. This book tells you what you need to know to use SQL's rich assortment of tools and features inside your programs.

## Icons Used in This Book

When something in this book is particularly valuable, we go out of our way to make sure that it stands out. We use these cool icons to mark text that (for one reason or another) *really* needs your attention. Here's a quick preview of the ones waiting for you in this book and what they mean.



TIP

Tips save you a lot of time and keep you out of trouble.



REMEMBER

Pay attention to the information marked by this icon — you may need it later.



WARNING

Heeding the advice that this icon points to can save you from major grief. Ignore it at your peril.



TECHNICAL  
STUFF

This icon alerts you to the presence of technical details that are interesting but not absolutely essential to understanding the topic being discussed.

# Beyond the Book

---

In addition to the content in this book, you'll find some extra content available at the `www.dummies.com` website:

- » **For the Cheat Sheet for this book, visit** `www.dummies.com/` and search for SQL For Dummies 9E cheat sheet.
- » **For updates to this book, if any, visit the** `www.dummies.com` store and search for SQL For Dummies 9E.

# Where to Go from Here

---

Now for the fun part! Databases are the best tools ever invented for keeping track of the things you care about. After you understand databases and can use SQL to make them do your bidding, you wield tremendous power. Co-workers come to you when they need critical information. Managers seek your advice. Youngsters ask for your autograph. But most importantly, you know, at a very deep level, how your organization really works.



# 1

## Getting Started with SQL

**IN THIS PART . . .**

The essentials of relational databases

Basic SQL concepts

Fundamental database tools



## IN THIS CHAPTER

- » Organizing information
- » Defining “database” in digital terms
- » Deciphering DBMS
- » Looking at the evolution of database models
- » Defining “*relational* database” (can you relate?)
- » Considering the challenges of database design

# Chapter 1

# Relational Database Fundamentals

**S**QL (pronounced *ess-que-ell*, not *see'qwl*, though database geeks still argue about that) is a language specifically designed with databases in mind. SQL enables people to create databases, add new data to them, maintain the data in them, and retrieve selected parts of the data. Developed in the 1970s at IBM, SQL has grown and advanced over the years to become the industry standard. It is governed by a formal standard maintained by the International Standards Organization (ISO).

Various kinds of databases exist, each adhering to a different model of how the data in the database is organized.

SQL was originally developed to operate on data in databases that follow the *relational model*. Recently, the international SQL standard has incorporated part of the *object model*, resulting in hybrid structures called object-relational databases. In this chapter, I discuss data storage, devote a section to how the relational model

compares with other major models, and provide a look at the important features of relational databases.

Before I talk about SQL, however, I want to nail down what I mean by the term *database*. Its meaning has changed, just as computers have changed the way people record and maintain information.

## Keeping Track of Things

Today people use computers to perform many tasks formerly done with other tools. Computers have replaced typewriters for creating and modifying documents. They've surpassed calculators as the best way to do math. They've also replaced millions of pieces of paper, file folders, and file cabinets as the principal storage medium for important information. Compared with those old tools, of course, computers do much more, much faster — and with greater accuracy. These increased benefits do come at a cost, however: Computer users no longer have direct physical access to their data.

When computers occasionally fail, office workers may wonder whether computerization really improved anything at all. In the old days, a manila file folder “crashed” only if you dropped it — then you merely knelt down, picked up the papers, and put them back in the folder. Barring earthquakes or other major disasters, file cabinets never “went down,” and they never gave you an error message. A hard-drive crash is another matter entirely: You can't “pick up” lost bits and bytes. Mechanical, electrical, and human failures can make your data go away into the Great Beyond, never to return. Backing up your data frequently is one thing you can do to enhance your peace of mind. Another thing you can do is store your data in the cloud and let your cloud provider do the backing up.

Taking the necessary precautions to protect yourself from accidental data loss allows you to start cashing in on the greater speed and accuracy that computers provide.

If you're storing important data, you have four main concerns:

- » Storing data must be quick and easy because you're likely to do it often.
- » The storage medium must be reliable. You don't want to come back later and find some (or all) of your data missing.

- » Data retrieval must be quick and easy, regardless of how many items you store.
- » You need an easy way to separate the exact information you want *now* from the tons of data that you *don't* want right now.

State-of-the-art computer databases satisfy these four criteria. If you store more than a dozen or so data items, you probably want to store those items in a database.

## What Is a Database?

The term *database* has fallen into loose use lately, losing much of its original meaning. To some people, a database is any collection of data items (phone books, laundry lists, parchment scrolls . . . whatever). Other people define the term more strictly.

In this book, I define a *database* as a self-describing collection of integrated records. And yes, that does imply computer technology, complete with programming languages such as SQL.



REMEMBER

A *record* is a representation of some physical or conceptual object. Say, for example, that you want to keep track of a business's customers. You assign a record for each customer. Each record has multiple *attributes*, such as name, address, and telephone number. Individual names, addresses, and so on are the *data*.

A database consists of both data and *metadata*. Metadata is the data that describes the data's structure within a database. If you know how your data is arranged, then you can retrieve it. Because the database contains a description of its own structure, it's *self-describing*. The database is *integrated* because it includes not only data items but also the relationships among data items.

The database stores metadata in an area called the *data dictionary*, which describes the tables, columns, indexes, constraints, and other items that make up the database.

Because a flat-file system (described later in this chapter) has no metadata, applications written to work with flat files must contain the equivalent of the metadata as part of the application program.

# Database Size and Complexity

Databases come in all sizes, from simple collections of a few records to mammoth systems holding millions of records. Most databases fall into one of three categories, which are based on the size of the database itself, the size of the equipment it runs on, and the size of the organization that is maintaining it:

- » A **personal database** is designed for use by a single person on a single computer. Such a database usually has a rather simple structure and a relatively small size.
- » A **departmental or workgroup database** is used by the members of a single department or workgroup within an organization. This type of database is generally larger than a personal database and is necessarily more complex; such a database must handle multiple users trying to access the same data at the same time.
- » An **enterprise database** can be huge. Enterprise databases may model the critical information flow of entire large organizations.

## What Is a Database Management System?

Glad you asked. A *database management system* (DBMS) is a set of programs used to define, administer, and process databases and their associated applications. The database being managed is, in essence, a structure that you build to hold valuable data. A DBMS is the tool you use to build that structure and operate on the data contained within the database.

You can find many DBMS programs on the market today. Some run on large and powerful machines, and some on personal computers, notebooks, and tablets. Some even run on smartphones. A strong trend, however, is for such products to work on multiple platforms or on networks that contain different classes of machines. An even stronger trend is to store data in data centers or even to store it out in the *cloud*, which could be a public cloud run by a large company such as Amazon, Google, or Microsoft, via the Internet, or it could be a private cloud operated by the same organization that is storing the data on its own intranet.

These days, *cloud* is a buzzword that is bandied about incessantly in techie circles. Like the puffy white things up in the sky, it has indistinct edges and seems to float somewhere out there. In reality, it is a collection of computing resources that is accessible via a browser, either over the Internet or on a private intranet. The thing that distinguishes the computing resources in the cloud from similar

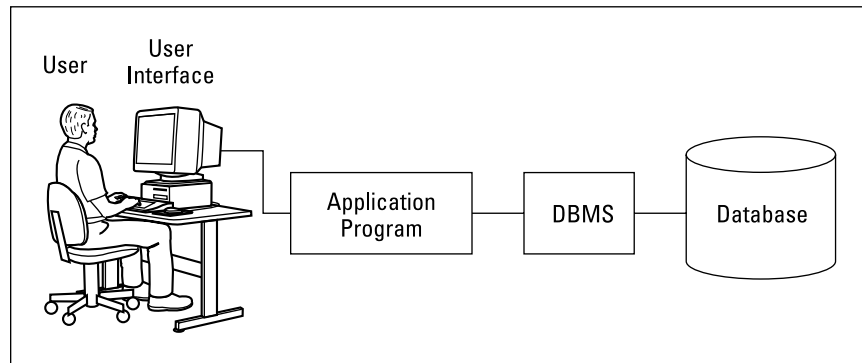
computing resources in a physical data center is the fact that the resources are accessible via a browser rather than an application program that directly accesses those resources.



REMEMBER

A DBMS that runs on platforms of multiple classes, large and small, is called *scalable*.

Whatever the size of the computer that hosts the database — and regardless of whether the machine is connected to a network — the flow of information between database and user is always the same. Figure 1-1 shows that the user communicates with the database through the DBMS. The DBMS masks the physical details of the database storage so that the application need only concern itself with the logical characteristics of the data, not with how the data is stored.



**FIGURE 1-1:**  
A block diagram  
of a DBMS-based  
information  
system.

## THE VALUE IS NOT IN THE DATA, BUT IN THE STRUCTURE

Years ago, some clever person calculated that if you reduce human beings to their components of carbon, hydrogen, oxygen, and nitrogen atoms (plus traces of others), they would be worth only 97 cents. However, droll this assessment, it's misleading. People aren't composed of mere isolated collections of atoms. Our atoms combine into enzymes, proteins, hormones, and many other substances that would cost millions of dollars per ounce on the pharmaceutical market. The precise structure of these combinations of atoms is what gives them greater value. By analogy, database structure makes possible the interpretation of seemingly meaningless data. The structure brings to the surface patterns, trends, and tendencies in the data. Unstructured data — like uncombined atoms — has little or no value.

# Flat Files

Where structured data is concerned, the flat file is as simple as it gets. No, a flat file isn't a folder that's been squashed under a stack of books. *Flat files* are so called because they have minimal structure. If they were buildings, they'd barely stick up from the ground. A flat file is simply a collection of data records, one after another, in a specified format — the data, the whole data, and nothing but the data — in effect, a list. In computer terms, a flat file is simple. Because the file doesn't store structural information (metadata), its overhead (stuff in the file that is not data but takes up storage space) is minimal.

Say that you want to keep track of the names and addresses of your company's customers in a flat file system. The system may have a structure something like this:

Harold Percival	26262 S. Howards Mill Rd	Westminster	CA92683
Jerry Appel	32323 S. River Lane Rd	Santa Ana	CA92705
Adrian Hansen	232 Glenwood Court	Anaheim	CA92640
John Baker	2222 Lafayette St	Garden Grove	CA92643
Michael Pens	77730 S. New Era Rd	Irvine	CA92715
Bob Michimoto	25252 S. Kelmsley Dr	Stanton	CA92610
Linda Smith	444 S.E. Seventh St	Costa Mesa	CA92635
Robert Funnell	2424 Sheri Court	Anaheim	CA92640
Bill Checkal	9595 Curry Dr	Stanton	CA92610
Jed Style	3535 Randall St	Santa Ana	CA92705

As you can see, the file contains nothing but data. Each field has a fixed length (the Name field, for example, is always exactly 15 characters long), and no structure separates one field from another. The person who created the database assigned field positions and lengths. Any program using this file must “know” how each field was assigned, because that information is not contained in the database itself.

Such low overhead means that operating on flat files can be very fast. On the minus side, however, application programs must include logic that manipulates the file's data at a very detailed level. The application must know exactly where and how the file stores its data. Thus, for small systems, flat files work fine. The larger a system is, however, the more cumbersome a flat-file system becomes.



TIP

Using a database instead of a flat-file system eliminates duplication of effort. Although database files themselves may have more overhead, the applications can be more portable across various hardware platforms and operating systems. A database also makes writing application programs easier because the programmer doesn't need to know the physical details of where and how the data is stored.