

# C O N T E N T S

## Introduction

xv

## I Names and Data Elements

1

### 1.1 Names

2

1.1.1 Watch the Length of Names

2

1.1.2 Avoid All Special Characters in Names

3

1.1.3 Avoid Quoted Identifiers

4

1.1.4 Enforce Capitalization Rules to Avoid  
Case-Sensitivity Problems

6

### 1.2 Follow the ISO-11179 Standards Naming Conventions

7

1.2.1 ISO-11179 for SQL

8

1.2.2 Levels of Abstraction

9

1.2.3 Avoid Descriptive Prefixes

10

1.2.4 Develop Standardized Postfixes

12

1.2.5 Table and View Names Should Be Industry Standards,  
Collective, Class, or Plural Nouns

14

1.2.6 Correlation Names Follow the Same Rules as Other  
Names . . . Almost

15

1.2.7 Relationship Table Names Should Be Common  
Descriptive Terms

17

1.2.8 Metadata Schema Access Objects Can Have Names That  
Include Structure Information

18

### 1.3 Problems in Naming Data Elements

18

1.3.1 Avoid Vague Names

18

1.3.2 Avoid Names That Change from Place to Place

19

1.3.3 Do Not Use Proprietary Exposed Physical Locators

21

## 2 Fonts, Punctuation, and Spacing

23

### 2.1 Typography and Code

23

2.1.1 Use Only Upper- and Lowercase Letters, Digits, and  
Underscores for Names

25

2.1.2 Lowercase Scalars Such as Column Names, Parameters,  
and Variables

25





	2.1.3	Capitalize Schema Object Names	26
	2.1.4	Uppercase the Reserved Words	26
	2.1.5	Avoid the Use of CamelCase	29
2.2		Word Spacing	30
2.3		Follow Normal Punctuation Rules	31
2.4		Use Full Reserved Words	33
2.5		Avoid Proprietary Reserved Words if a Standard Keyword Is Available in Your SQL Product	33
2.6		Avoid Proprietary Statements if a Standard Statement Is Available	34
2.7		Rivers and Vertical Spacing	37
2.8		Indentation	38
2.9		Use Line Spacing to Group Statements	39
<b>3</b>		<b>Data Declaration Language</b>	<b>41</b>
	3.1	Put the Default in the Right Place	41
	3.2	The Default Value Should Be the Same Data Type as the Column	42
	3.3	Do Not Use Proprietary Data Types	42
	3.4	Place the PRIMARY KEY Declaration at the Start of the CREATE TABLE Statement	44
	3.5	Order the Columns in a Logical Sequence and Cluster Them in Logical Groups	44
	3.6	Indent Referential Constraints and Actions under the Data Type	45
	3.7	Give Constraints Names in the Production Code	46
	3.8	Put CHECK() Constraint Near what they Check	46
	3.8.1	Consider Range Constraints for Numeric Values	47
	3.8.2	Consider LIKE and SIMILAR TO Constraints for Character Values	47
	3.8.3	Remember That Temporal Values Have Duration	48
	3.8.4	REAL and FLOAT Data Types Should Be Avoided	48
	3.9	Put Multiple Column Constraints as Near to Both Columns as Possible	48
	3.10	Put Table-Level CHECK() Constraints at the End of the Table Declaration	49
	3.11	Use CREATE ASSERTION for Multi-table Constraints	49



3.12	Keep CHECK() Constraints Single Purposed	50
3.13	Every Table Must Have a Key to Be a Table	51
3.13.1	Auto-Numbers Are Not Relational Keys	52
3.13.2	Files Are Not Tables	53
3.13.3	Look for the Properties of a Good Key	54
3.14	Do Not Split Attributes	62
3.14.1	Split into Tables	63
3.14.2	Split into Columns	63
3.14.3	Split into Rows	65
3.15	Do Not Use Object-Oriented Design for an RDBMS	66
3.15.1	A Table Is Not an Object Instance	66
3.15.2	Do Not Use EAV Design for an RDBMS	68
<b>4</b>	<b>Scales and Measurements</b>	<b>69</b>
4.1	Measurement Theory	69
4.1.1	Range and Granularity	71
4.1.2	Range	72
4.1.3	Granularity, Accuracy, and Precision	72
4.2	Types of Scales	73
4.2.1	Nominal Scales	73
4.2.2	Categorical Scales	73
4.2.3	Absolute Scales	74
4.2.4	Ordinal Scales	74
4.2.5	Rank Scales	75
4.2.6	Interval Scales	76
4.2.7	Ratio Scales	76
4.3	Using Scales	77
4.4	Scale Conversion	77
4.5	Derived Units	79
4.6	Punctuation and Standard Units	80
4.7	General Guidelines for Using Scales in a Database	81
<b>5</b>	<b>Data Encoding Schemes</b>	<b>83</b>
5.1	Bad Encoding Schemes	84
5.2	Encoding Scheme Types	86



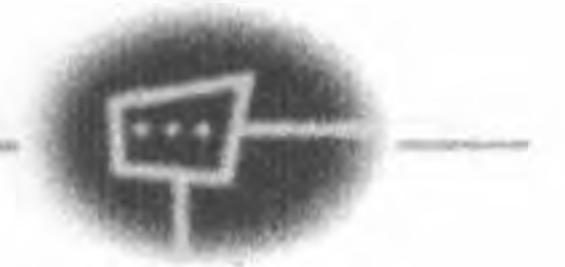


	5.2.1	Enumeration Encoding	86
	5.2.2	Measurement Encoding	87
	5.2.3	Abbreviation Encoding	87
	5.2.4	Algorithmic Encoding	88
	5.2.5	Hierarchical Encoding Schemes	89
	5.2.6	Vector Encoding	90
	5.2.7	Concatenation Encoding	91
	5.3	General Guidelines for Designing Encoding Schemes	92
	5.3.1	Existing Encoding Standards	92
	5.3.2	Allow for Expansion	92
	5.3.3	Use Explicit Missing Values to Avoid NULLs	92
	5.3.4	Translate Codes for the End User	93
	5.3.5	Keep the Codes in the Database	96
	5.4	Multiple Character Sets	97
<b>6</b>		<b>Coding Choices</b>	<b>99</b>
	6.1	Pick Standard Constructions over Proprietary Constructions	100
	6.1.1	Use Standard OUTER JOIN Syntax	101
	6.1.2	Infix INNER JOIN and CROSS JOIN Syntax Is Optional, but Nice	105
	6.1.3	Use ISO Temporal Syntax	107
	6.1.4	Use Standard and Portable Functions	108
	6.2	Pick Compact Constructions over Longer Equivalents	109
	6.2.1	Avoid Extra Parentheses	109
	6.2.2	Use CASE Family Expressions	110
	6.2.3	Avoid Redundant Expressions	113
	6.2.4	Seek a Compact Form	114
	6.3	Use Comments	118
	6.3.1	Stored Procedures	119
	6.3.2	Control Statement Comments	119
	6.3.3	Comments on Clause	119
	6.4	Avoid Optimizer Hints	120
	6.5	Avoid Triggers in Favor of DRI Actions	120
	6.6	Use SQL Stored Procedures	122



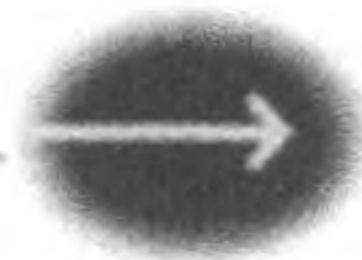
6.7	Avoid User-Defined Functions and Extensions inside the Database	123
6.7.1	Multiple Language Problems	124
6.7.2	Portability Problems	124
6.7.3	Optimization Problems	124
6.8	Avoid Excessive Secondary Indexes	124
6.9	Avoid Correlated Subqueries	125
6.10	Avoid UNIONS	127
6.11	Testing SQL	130
6.11.1	Test All Possible Combinations of NULLs	130
6.11.2	Inspect and Test All CHECK() Constraints	130
6.11.3	Beware of Character Columns	131
6.11.4	Test for Size	131
<b>7</b>	<b>How to Use VIEWS</b>	<b>133</b>
7.1	VIEW Naming Conventions Are the Same as Tables	135
7.1.1	Always Specify Column Names	136
7.2	VIEWS Provide Row- and Column-Level Security	136
7.3	VIEWS Ensure Efficient Access Paths	138
7.4	VIEWS Mask Complexity from the User	138
7.5	VIEWS Ensure Proper Data Derivation	139
7.6	VIEWS Rename Tables and/or Columns	140
7.7	VIEWS Enforce Complicated Integrity Constraints	140
7.8	Updatable VIEWS	143
7.8.1	WITH CHECK OPTION clause	143
7.8.2	INSTEAD OF Triggers	144
7.9	Have a Reason for Each VIEW	144
7.10	Avoid VIEW Proliferation	145
7.11	Synchronize VIEWS with Base Tables	145
7.12	Improper Use of VIEWS	146
7.12.1	VIEWS for Domain Support	146
7.12.2	Single-Solution VIEWS	147
7.12.3	Do Not Create One VIEW Per Base Table	148
7.13	Learn about Materialized VIEWS	149





<b>8</b>	<b>How to Write Stored Procedures</b>	<b>151</b>
8.1	Most SQL 4GLs Are Not for Applications	152
8.2	Basic Software Engineering	153
8.2.1	Cohesion	153
8.2.2	Coupling	155
8.3	Use Classic Structured Programming	156
8.3.1	Cyclomatic Complexity	157
8.4	Avoid Portability Problems	158
8.4.1	Avoid Creating Temporary Tables	158
8.4.2	Avoid Using Cursors	159
8.4.3	Prefer Set-Oriented Constructs to Procedural Code	161
8.5	Scalar versus Structured Parameters	167
8.6	Avoid Dynamic SQL	168
8.6.1	Performance	169
8.6.2	SQL Injection	169
<b>9</b>	<b>Heuristics</b>	<b>171</b>
9.1	Put the Specification into a Clear Statement	172
9.2	Add the Words “Set of All...” in Front of the Nouns	173
9.3	Remove Active Verbs from the Problem Statement	174
9.4	You Can Still Use Stubs	174
9.5	Do Not Worry about Displaying the Data	176
9.6	Your First Attempts Need Special Handling	177
9.6.1	Do Not Be Afraid to Throw Away Your First Attempts at DDL	177
9.6.2	Save Your First Attempts at DML	178
9.7	Do Not Think with Boxes and Arrows	179
9.8	Draw Circles and Set Diagrams	179
9.9	Learn Your Dialect	180
9.10	Imagine That Your WHERE Clause Is “Super Ameba”	180
9.11	Use the Newsgroups and Internet	181
<b>10</b>	<b>Thinking in SQL</b>	<b>183</b>
10.1	Bad Programming in SQL and Procedural Languages	184





10.2	Thinking of Columns as Fields	189
10.3	Thinking in Processes, Not Declarations	191
10.4	Thinking the Schema Should Look Like the Input Forms	194
<b>Resources</b>		<b>197</b>
	Military Standards	197
	Metadata Standards	197
	ANSI and ISO Standards	198
	U.S. Government Codes	199
	Retail Industry	199
	Code Formatting and Naming Conventions	200
<b>Bibliography</b>		<b>203</b>
	Reading Psychology	203
	Programming Considerations	204
<b>Index</b>		<b>207</b>
<b>About the Author</b>		<b>217</b>

trying to teach you to program in SQL in this book. You might want to read that again. If that is what you wanted, there are better books. This ought to be the second book you buy, not the first.

I assume that you already write SQL at some level and want to get better at it. If you want to learn SQL programming tricks, get a copy of my other book, *SQL for Smarties* (3rd edition, 2005). I am trying to teach the reader how to work in logical and declarative terms, instead of in a procedural or OO manner—“Query Eye for the Database Guy,” if you will forgive a horrible contemporary pun.

Few, if any, SQL programmers came to SQL before learning and writing for years in a procedural or object-oriented language. They then got one particular SQL product and were told to learn it on their own or with a book that has a title like “SQL for Brain-Dead Morons,” “Learn SQL in Ten Easy Lessons or Five Hard Ones,” or worse.

This is absurd! It takes at least five years to learn to be a master carpenter or chef. Why would you believe people could become SQL gurus in a weekend? What they become is bad SQL programmers, who speak SQL in dialect from the local SQL product with a strong accent from their previous languages. You might want to read “Teach Yourself Programming in Ten Years” by Peter Norvig ([www.norvig.com/21-days.html](http://www.norvig.com/21-days.html)) or “No Silver Bullets” by Fred Brooks, *Computer*, 20(4):10-19, April 1987) to get a reality check.