

Head First SQL

Wouldn't it be dreamy if there was a book that could teach me SQL without making me want to relocate to a remote island in the Pacific where there are no databases? It's probably nothing but a fantasy...



Lynn Beighley

O'REILLY®

Beijing • Cambridge • Köln • Sebastopol • Taipei • Tokyo

Head First SQL

by Lynn Beighley

Copyright © 2007 O'Reilly Media, Inc. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Series Creators:	Kathy Sierra, Bert Bates
Series Editor:	Brett D. McLaughlin
Editor:	Catherine Nolan
Design Editor:	Louise Barr
Cover Designers:	Louise Barr, Karen Montgomery
Production Editor:	Sanders Kleinfeld
Indexer:	Julie Hawks
Page Viewer:	Andrew Fader

Printing History:

August 2007: First Edition.

He's incredibly patient.



The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. The *Head First* series designations, *Head First SQL*, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc., was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and the authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

No clowns, doughnuts, or Girl Sprouts were harmed in the making of this book. Just my car, but it's been fixed.



This book uses RepKover™, a durable and flexible lay-flat binding.

ISBN: 978-0-596-52684-9

[M]

[6/09]

Author of Head First SQL



↑
Lynn Beighley

Lynn is a fiction writer stuck in a technical book writer's body. Upon discovering that technical book writing actually paid real money, she learned to accept and enjoy it.

After going back to school to get a Masters in computer science, she worked for the acronyms NRL and LANL. Then she discovered Flash, and wrote her first bestseller.

A victim of bad timing, she moved to Silicon Valley just before the great crash. She spent several years working for Yahoo! and writing other books and training courses. Finally giving in to her creative writing bent, she moved to the New York area to get an MFA in creative writing.

Her Head First–style thesis was delivered to a packed room of professors and fellow students. It was extremely well received, and she finished her degree, finished *Head First SQL*, and can't wait to begin her next book.

Lynn loves traveling, cooking, and making up elaborate background stories about complete strangers. She's a little scared of clowns.



↑
The view from
Lynn's window.

Table of Contents (Summary)

	Intro	xxv
1	Data and Tables: <i>A place for everything</i>	1
2	The SELECT Statement: <i>Gifted data retrieval</i>	53
3	DELETE and UPDATE: <i>A change will do you good</i>	119
4	Smart Table Design: <i>Why be normal?</i>	159
5	ALTER: <i>Rewriting the past</i>	197
6	Advanced SELECT: <i>Seeing your data with new eyes</i>	235
7	Multi-table Database Design: <i>Outgrowing your table</i>	281
8	Joins and Multi-table Operations: <i>Can't we all just get along?</i>	343
9	Subqueries: <i>Queries Within Queries</i>	379
10	Outer Joins, Self Joins, and Unions: <i>New maneuvers</i>	417
11	Constraints, Views, and Transactions: <i>Too many cooks spoil the database</i>	455
12	Security: <i>Protecting your assets</i>	493

Table of Contents (the real thing)

Intro

Your brain on SQL. Here you are trying to *learn* something, while here your *brain* is doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing SQL?

Who is this book for?	xxvi
We know what you're thinking	xxvii
Metacognition	xxix
Bend your brain into submission	xxxi
Read me	xxxii
The technical review team	xxxiv
Acknowledgments	xxxv

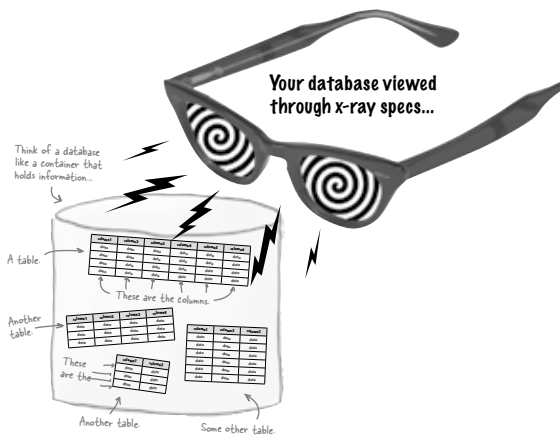
data and tables

1

A place for everything

Don't you just hate losing things? Whether it's your car keys, that 25% off coupon for Urban Outfitters, or your application's data, there's nothing worse than not being able to **keep up with what you need...** when you need it. And when it comes to your applications, there's no better place to store your important information than in a **table**. So turn the page, come on in, and take a walk through the world of **relational databases**.

Defining your data	2
Look at your data in categories	7
What's in a database?	8
Your database viewed through x-ray specs...	10
Databases contain connected data	12
Tables Up Close	13
Take command!	17
Setting the table: the CREATE TABLE statement	19
Creating a more complicated table	20
Look how easy it is to write SQL	21
Create the my_contacts table, finally	22
Your table is ready	23
Take a meeting with some data types	24
Your table, DESCRibed	28
You can't recreate an existing table or database!	30
Out with the old table, in with the new	32
To add data to your table, you'll use the INSERT statement	34
Create the INSERT statement	37
Variations on an INSERT statement	41
Columns without values	42
Peek at your table with the SELECT statement	43
SQL Exposed: Confessions of a NULL	44
Controlling your inner NULL	45
NOT NULL appears in DESC	47
Fill in the blanks with DEFAULT	48
Your SQL Toolbox	50



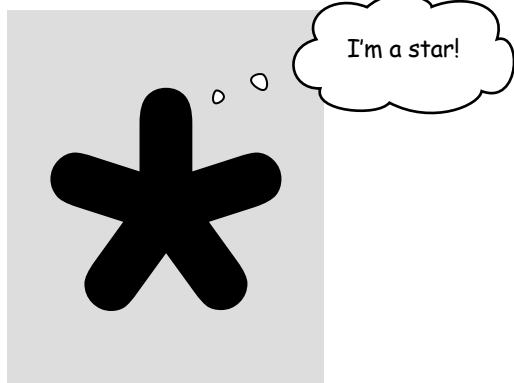
the **SELECT** statement

Gifted data retrieval

2

Is it really better to give than retrieve? When it comes to databases, chances are you'll need to **retrieve your data** as often than you'll need to insert it. That's where this chapter comes in: you'll meet the powerful **SELECT** statement and learn how to **gain access to that important information** you've been putting in your tables. You'll even learn how to use **WHERE**, **AND**, and **OR** to selectively get to your data and even avoid displaying the data that you *don't* need.

Date or no date?	54
A better SELECT	57
What the * is that?	58
How to query your data types	64
More punctuation problems	65
Unmatched single quotes	66
Single quotes are special characters	67
INSERT data with single quotes in it	68
SELECT specific columns to limit results	73
SELECT specific columns for faster results	73
Combining your queries	80
Finding numeric values	83
Smooth Comparison Operators	86
Finding numeric data with Comparison Operators	88
Text data roping with Comparison Operators	91
To be OR not to be	93
The difference between AND and OR	96
Use IS NULL to find NULLs	99
Saving time with a single keyword: LIKE	101
The call of the Wild(card)	101
Selecting ranges using AND and comparison operators	105
Just BETWEEN us... there's a better way	106
After the dates, you are either IN...	109
... or you are NOT IN	110
More NOT	111
Your SQL Toolbox	116



DELETE and UPDATE

A change will do you good

3

Keep changing your mind? Now it's OK! With the commands you're about to learn—**DELETE** and **UPDATE**—you're no longer stuck with a decision you made six months ago, when you first inserted that data about mullets coming back into style soon. With **UPDATE**, you **can change data**, and **DELETE** lets you **get rid of data** that you don't need anymore. But we're not just giving you the tools; in this chapter, you'll learn how to be selective with your new powers and avoid dumping data that you really do need.

Clowns are scary	120
Clown tracking	121
The clowns are on the move	122
How our clown data gets entered	126
Bonzo, we've got a problem	128
Getting rid of a record with DELETE	129
Using our new DELETE statement	131
DELETE rules	132
The INSERT-DELETE two step	135
Be careful with your DELETE	140
The trouble with imprecise DELETE	144
Change your data with UPDATE	146
UPDATE rules	147
UPDATE is the new INSERT-DELETE	148
UPDATE in action	149
Updating the clowns' movements	152
UPDATE your prices	154
All we need is one UPDATE	156
Your SQL Toolbox	158



Do we scare you?

smart table design

Why be normal?

4

You've been creating tables without giving much thought to them. And that's fine, they work. You can `SELECT`, `INSERT`, `DELETE`, and `UPDATE` with them. But as you **get more data**, you start seeing **things you wish you'd done** to make your `WHERE` clauses simpler. What you need is to make your tables more *normal*.

Two fishy tables	160
A table is all about relationships	164
Atomic data	168
Atomic data and your tables	170
Atomic data rules	171
Reasons to be normal	174
The benefits of normal tables	175
Clowns aren't normal	176
Halfway to 1NF	177
PRIMARY KEY rules	178
Getting to NORMAL	181
Fixing Greg's table	182
The <code>CREATE TABLE</code> we wrote	183
Show me the table money	184
Time-saving command	185
The <code>CREATE TABLE</code> with a PRIMARY KEY	186
1, 2, 3... auto incrementally	188
Adding a PRIMARY KEY to an existing table	192
<code>ALTER TABLE</code> and add a PRIMARY KEY	193
Your SQL Toolbox	194

Wait a second. I already have a table full of data. You can't seriously expect me to use the `DROP TABLE` command like I did in chapter 1 and type in all that data again, just to create a primary key for each record...



ALTER

5

Rewriting the Past

Ever wished you could correct the mistakes of your past?

Well, now is your chance. By using the **ALTER command**, you can apply all the lessons you've been learning to tables you designed days, months, even years ago. Even better, you can do it without affecting your data. By the time you're through here, you'll know what **normal** really means, and you'll be able to apply it to all your tables, past and present.

We need to make some changes	198
Table altering	203
Extreme table makeover	204
Renaming the table	205
We need to make some plans	207
Retooling our columns	208
Structural changes	209
ALTER and CHANGE	210
Change two columns with one SQL statement	211
Quick! DROP that column	215
A closer look at the non-atomic location column	222
Look for patterns	223
A few handy string functions	224
Use a current column to fill a new column	229
How our UPDATE and SET combo works	230
Your SQL Toolbox	232



It's time to turn your tired old hooptie table into a date magnet and take it to a level of table pimpification you never knew existed.



advanced *SELECT*

Seeing your data with new eyes

6

It's time to add a little finesse to your toolbox. You already know how to *SELECT* data and use *WHERE* clauses. But sometimes you need more **precision** than *SELECT* and *WHERE* provide. In this chapter, you'll learn about how to **order and group** your data, as well as how to **perform math operations** on your results.

Dataville Video is reorganizing	236
Problems with our current table	237
Matching up existing data	238
Populating the new column	239
<i>UPDATE</i> with a <i>CASE</i> expression	242
Looks like we have a problem	244
Tables can get messy	249
We need a way to organize the data we <i>SELECT</i>	250
Try a little <i>ORDER BY</i>	253
<i>ORDER</i> a single column	254
<i>ORDER</i> with two columns	257
<i>ORDER</i> with multiple columns	258
An orderly <i>movie_table</i>	259
Reverse the <i>ORDER</i> with <i>DESC</i>	261
The Girl Sprout® cookie sales leader problem	263
<i>SUM</i> can add them for us	265
<i>SUM</i> all of them at once with <i>GROUP BY</i>	266
<i>AVG</i> with <i>GROUP BY</i>	267
<i>MIN</i> and <i>MAX</i>	268
<i>COUNT</i> the days	269
<i>SELECT DISTINCT</i> values	271
<i>LIMIT</i> the number of results	274
<i>LIMIT</i> to just second place	275
Your <i>SQL</i> Toolbox	278

**DATAVILLE
VIDEO**



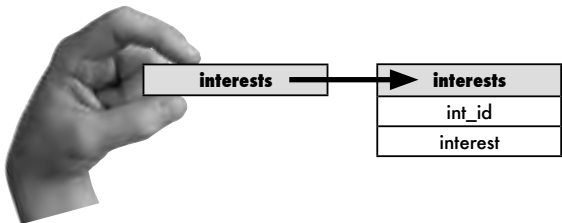
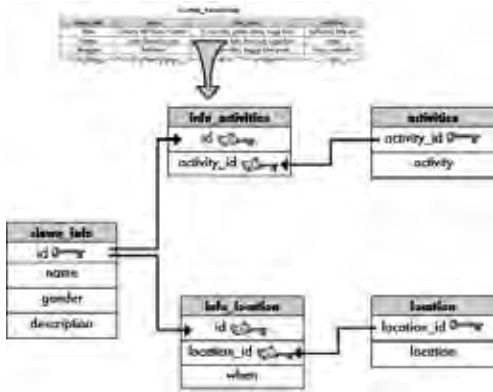
multi-table database design

7 Outgrowing your table

Sometimes your single table isn't big enough anymore.

Your data has become more complex, and that **one table** you've been using just **isn't cutting it**. Your single table is full of redundant data, wasting space and slowing down your queries. You've gone as far as you can go with a single table. It's a big world out there, and sometimes you need **more than one table** to contain your data, control it, and ultimately, be the master of your own database.

Finding Nigel a date	282
All is lost... But wait	293
Think outside of the single table	294
The multi-table clown tracking database	295
The clowntracking database schema	296
How to go from one table to two	298
Connecting your tables	303
Constraining your foreign key	305
Why bother with foreign keys?	306
CREATE a table with a FOREIGN KEY	307
Relationships between tables	309
Patterns of data: one-to-one	309
Patterns of data: when to use one-to-one tables	310
Patterns of data: one-to-many	311
Patterns of data: getting to many-to-many	312
Patterns of data: we need a junction table	315
Patterns of data: many-to-many	316
Finally in 1NF	321
Composite keys use multiple columns	322
Shorthand notations	324
Partial functional dependency	325
Transitive functional dependency	326
Second normal form	330
Third normal form (at last)	336
And so, Regis (and gregslis) lived happily ever after	339
Your SQL Toolbox	340



joins and multi-table operations

Can't we all just get along?

8

Welcome to a multi-table world. It's great to have **more than one table** in your database, but you'll need to learn some **new tools and techniques** to work with them. With multiple tables comes confusion, so you'll need **aliases** to keep your tables straight. And **joins** help you connect your tables, so that you can get at all the data you've spread out. Get ready, it's time to **take control of your database** again.

Still repeating ourselves, still repeating...	344
Prepopulate your tables	345
We got the "table ain't easy to normalize" blues	347
The special interests (column)	348
Keeping interested	349
UPDATE all your interests	350
Getting all the interests	351
Many paths to one place	352
CREATE, SELECT and INSERT at (nearly) the same time	352
CREATE, SELECT and INSERT at the same time	353
What's up with that AS?	354
Column aliases	355
Table aliases, who needs 'em?	356
Everything you wanted to know about inner joins	357
Cartesian join	358
Releasing your inner join	363
The inner join in action: the equijoin	364
The inner join in action: the non-equijoin	367
The last inner join: the natural join	368
Joined-up queries?	375
Table and Column Aliases Exposed: What are you hiding from?	376
Your SQL Toolbox	377

...and that's where
little result tables
really come from.



subqueries

9

Queries within queries

Yes, Jack, I'd like a two-part question, please. Joins are great, but sometimes you need to *ask your database more than one question*. Or *take the result of one query and use it as the input to another query*. That's where **subqueries** come in. They'll help you **avoid duplicate data, make your queries more dynamic**, and even get you in to all those high-end concert afterparties. (Well, not really, but two out of three ain't bad!)

Greg gets into the job recruiting business	380
Greg's list gets more tables	381
Greg uses an inner join	382
But he wants to try some other queries	384
Subqueries	386
We combine the two into a query with a subquery	387
As if one query wasn't enough: meet the subquery	388
A subquery in action	389
Subquery rules	391
A subquery construction walkthrough	394
A subquery as a SELECT column	397
Another example: Subquery with a natural join	398
A noncorrelated subquery	399
SQL Exposed: Choosing the best way to query	400
A noncorrelated subquery with multiple values: IN, NOT IN	403
Correlated subqueries	408
A (useful) correlated subquery with NOT EXISTS	409
EXISTS and NOT EXISTS	410
Greg's Recruiting Service is open for business	412
On the way to the party	413
Your SQL Toolbox	414

OUTER query

INNER query

Outer query

```
SELECT some_column, another_column
FROM table
WHERE column = (SELECT column FROM table);
```

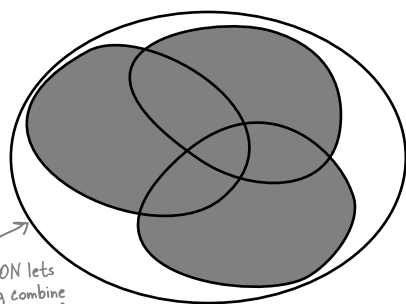
Inner query

outer joins, self-joins, and unions

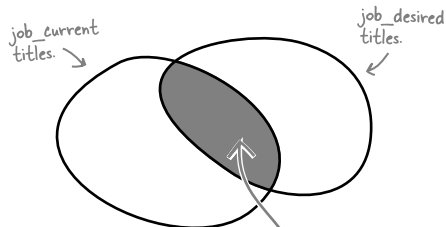
New maneuvers

10

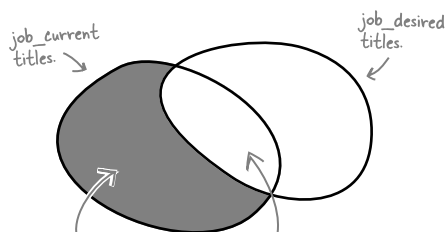
You only know half of the story about joins. You've seen cross joins that return every possible row, and inner joins that return rows from both tables where there is a match. But what you haven't seen are **outer joins** that give you back rows that *don't have matching counterparts in the other table*, **self-joins** which (strangely enough) *join a single table to itself*, and **unions** that *combine the results of queries*. Once you learn these tricks, you'll be able to get at all your data exactly the way you need to. (And we haven't forgotten about exposing the truth about subqueries, either!)



UNION lets Greg combine the results from these three separate queries into one table of results.



Titles must be in both tables to show up



Only titles that are NOT in the table specified by the EXCEPT show up.

Any titles that are in both tables will be excluded from the results.

Cleaning up old data	418
It's about left and right	419
Here's a left outer join	420
Outer joins and multiple matches	425
The right outer join	426
While you were outer joining...	429
We could create a new table	430
How the new table fits in	431
A self-referencing foreign key	432
Join the same table to itself	433
We need a self-join	435
Another way to get multi-table information	436
You can use a UNION	437
UNION is limited	438
UNION rules in action	439
UNION ALL	440
Create a table from your union	441
INTERSECT and EXCEPT	442
We're done with joins, time to move on to...	443
Subqueries and joins compared	443
Turning a subquery into a join	444
A self-join as a subquery	449
Greg's company is growing	450
Your SQL Toolbox	452

constraints, views, and transactions

11

Too many cooks spoil the database

Your database has grown and other people need to use it.

The problem is that some of them won't be as skilled at SQL as you are. You need ways to **keep them from entering the wrong data**, techniques for allowing them to **only see part of the data**, and ways to **stop them from stepping on each other when they try entering data at the same time**. In this chapter we begin protecting our data from the mistakes of others. Welcome to Defensive Databases, Part 1.

Greg's hired some help	456
Jim's first day: Inserting a new client	457
Jim avoids a NULL	458
Flash forward three months	459
CHECK, please: Adding a CHECK CONSTRAINT	460
CHECKing the gender	461
Frank's job gets tedious	463
Creating a view	465
Viewing your views	466
What your view is actually doing	467
What a view is	468
Inserting, updating, and deleting with views	471
The secret is to pretend a view is a real table	472
View with CHECK OPTION	475
Your view may be updatable if...	476
When you're finished with your view	477
When bad things happen to good databases	478
What happened inside the ATM	479
More trouble at the ATM	480
It's not a dream, it's a transaction	482
The classic ACID test	483
SQL helps you manage your transactions	484
What should have happened inside the ATM	485
How to make transactions work with MySQL	486
Now try it yourself	487
Your SQL Toolbox	490



12

security

Protecting your assets

You've put an enormous amount of time and energy into creating your database. And you'd be devastated if anything happened to it. You've also had to give other people **access to your data**, and you're worried that they might insert or update something incorrectly, or even worse, **delete the wrong data**. You're about to learn how databases and the objects in them can be made more **secure**, and how you can have complete control over **who can do what with your data**.

User problems	494
Avoiding errors in the clown tracking database	495
Protect the root user account	497
Add a new user	498
Decide exactly what the user needs	499
A simple GRANT statement	500
GRANT variations	503
REVOKE privileges	504
REVOKING a used GRANT OPTION	505
REVOKING with precision	506
The problem with shared accounts	510
Using your role	512
Role dropping	512
Using your role WITH ADMIN OPTION	514
Combining CREATE USER and GRANT	519
Greg's List has gone global!	520
Your SQL Toolbox	522
How about a Greg's List in your city?	524
Use SQL on your own projects and you too could be like Greg!	524



root



bashful



doc



dopey



grumpy



happy



sleepy



sneezy

leftovers

The Top Ten Topics (we didn't cover)



Even after all that, there's a bit more. There are just a few more things we think you need to know. We wouldn't feel right about ignoring them, even though they only need a brief mention. So before you put the book down, take a read through these **short but important SQL tidbits**.

Besides, once you're done here, all that's left is another appendix... and the index... and maybe some ads... and then you're really done. We promise!

#1. Get a GUI for your RDBMS	526
#2. Reserved Words and Special Characters	528
#3. ALL, ANY, and SOME	530
#4. More on Data Types	532
#5. Temporary tables	534
#6. Cast your data	535
#7. Who are you? What time is it?	536
#8. Useful numeric functions	537
#9. Indexing to speed things up	539
#10. 2-minute PHP/MySQL	540

A	ABSOLUTE ACTION ADD ADMIN AFTER AGGREGATE ALIAS ALL ALLOCATE ALTER AND ANY ARE ARRAY AS ASC ASSERTION AT AUTHORIZATION
B	BEFORE BEGIN BINARY BIT BLOB BOOLEAN BOTH BREADTH BY
C	CALL CASCADE CASCADED CASE CAST CATALOG CHAR CHARACTER CHECK CLASS CLOB CLOSE COLLATE COLLATION COLUMN COMMIT COMPLETION CONNECT CONNECTION CONSTRAINT CONSTRAINTS CONSTRUCTOR CONTINUE CORRESPONDING CREATE CROSS CUBE CURRENT CURRENT_DATE CURRENT_PATH CURRENT_ROLE CURRENT_TIME CURRENT_TIMESTAMP CURRENT_USER CURSOR CYCLE
D	DATA DATE DAY DEALLOCATE DEC DECIMAL DECLARE DEFAULT DEFERRABLE DEFERRED DELETE DEPTH DEREF DESC DESCRIBE DESCRIPTOR DESTROY DESTRUCTOR DETERMINISTIC DICTIONARY DIAGNOSTICS DISCONNECT DISTINCT DOMAIN DOUBLE DROP DYNAMIC
E	EACH ELSE END END_EXEC EQUALS ESCAPE EVERY EXCEPT EXCEPTION EXEC EXECUTE EXTERNAL
F	FALSE FETCH FIRST FLOAT FOR FOREIGN FOUND FROM FREE FULL FUNCTION
G	GENERAL GET GLOBAL GO GOTO GRANT GROUP GROUPING
H	HAVING HOST HOUR
I	IDENTITY IGNORE IMMEDIATE IN INDICATOR INITIALIZE INITIALLY INNER INOUT INPUT INSERT INT INTEGER INTERSECT INTERVAL INTO IS ISOLATION ITERATE
J	JOIN
K	KEY
L	LANGUAGE LARGE LAST LATERAL LEADING LEFT LESS LEVEL LIKE LIMIT LOCAL LOCALTIME LOCALTIMESTAMP LOCATOR
M	MAP MATCH MINUTE MODIFIES MODIFY MODULE MONTH
N	NAMES NATIONAL NATURAL NCHAR NCOL NDB NEXT NO NONE NOT NULL NUMERIC
O	OBJECT OF OFF OLD ON ONLY OPEN OPERATION OPTION OR ORDER ORDINALITY OUT OUTER OUTPUT
P	PAD PARAMETER PARAMETERS PARTIAL PATH POSTFIX PRECISION PREFIX PREORDER PREPARE PRESERVE PRIMARY PRIOR PRIVILEGES PROCEDURE PUBLIC
Q	
R	READ READS REAL RECURSIVE REF REFERENCES REFERENCING RELATIVE RESTRICT RESULT RETURN RETURNS REVOKE RIGHT ROLE ROLLBACK ROLLUP ROUTINE ROW ROWS
S	SAVEPOINT SCHEMA SCROLL SCOPE SEARCH SECOND SECTION SELECT SEQUENCE SESSION SESSION_USER SET SETS SIZE SMALLINT SOME SPACE SPECIFIC SPECIFICTYPE SQL SQLEXCEPTION SQLEXTYPE SQLNAMING START STATE STATEMENT STATIC STRUCTURE SYSTEM USER
T	TABLE TEMPORARY TERMINATE THAN THEN TIME TIMESTAMP TIMEZONE_HOUR TIMEZONE_MINUTE TO TRAILING TRANSACTION TRANSLATION TREAT TRIGGER TRUE
U	UNDER UNION UNIQUE UNKNOWN UNNEST UPDATE USAGE USER USING
V	VALUE VALUES VARCHAR VARIABLE VARYING VIEW
W	WHEN WHENEVER WHERE WITH WITHOUT WORK WRITE
X	
Y	YEAR
Z	ZONE

```
File Edit Window Help
> SELECT CURRENT_DATE;
+-----+
| CURRENT_DATE |
+-----+
| 2007-07-26   |
+-----+
1 row in set (0.00 sec)
```

```
File Edit Window Help
> SELECT CURRENT_TIME;
+-----+
| CURRENT_TIME |
+-----+
| 11:26:48     |
+-----+
1 row in set (0.00 sec)
```

```
File Edit Window Help
SELECT CURRENT_USER;
+-----+
| CURRENT_USER |
+-----+
| root@localhost |
+-----+
1 row in set (0.00 sec)
```

MySQL installation

Try it out for yourself



All your new SQL skills won't do you much good without a place to apply them. This appendix contains

instructions for getting your very own MySQL RDBMS for you to work with.

Get started, fast!	544
Instructions and Troubleshooting	544
Steps to Install MySQL on Windows	545
Steps to Install MySQL on Mac OS X	548



tools roundup

All your new SQL tools



Here are all your SQL tools in one place for the first time, for one night only (kidding)! This is a roundup of all the SQL tools we've covered. Take a moment to survey the list and feel great—you learned them all!



Symbols	552
A–B	552
C–D	553
E–I	554
L–N	555
O–S	556
T–X	557