THE EXPERT'S VOICE® IN MOBILE PROGRAMMING

Introducing SQLite for Mobile Developers

Enabling Database Functionality for Android and iPhone

Jesse Feiler

Apress®

Introducing SQLite for Mobile Developers

Jesse Feiler

Apress®

Introducing SQLite for Mobile Developers

ISBN-13 (pbk): 978-1-4842-1765-8

ISBN-13 (electronic): 978-1-4842-1766-5

Managing Director: Welmoed Spahr
Lead Editor: Jeffrey Pepper
Technical Reviewers: Aaron Crabtree and Cliff Wootton
Editorial Board: Steve Anglin, Pramila Balan, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Mark Powers
Copy Editor: Lori Jacobs
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc. is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com/9781484217658. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

Contents at a Glance

Contents

About the Author

Jesse Feiler is a developer, consultant, and author specializing in database technologies and location-based apps. He has worked with databases and data management on computers from mainframes to iPhone, iPad, and Apple TV using data management tools from DB2 (IBM) and DMSII (Burroughs) to Enterprise Objects Framework and Core Data, MySQL, Oracle, and, of course, SQLite.

In the early days of the web, he built the page caching mechanism for the Prodigy web browser for Mac using a relational database library similar in some ways to SQLite.

He is the creator of Minutes Machine the meeting management app, as well as Saranac River Trail app a guide to the Trail that includes location-based updates as well as social media tools. His apps are available in the App Store and are published by Champlain Arts Corp (`champlainarts.com`). As a consultant, he has worked with small businesses and nonprofits on projects such as production control, publishing, marketing and project management usually involving FileMaker and other databases.

His books include:

iOS Programming with Swift for Dummies (Wiley, 2015)

Swift for Dummies (Wiley, 2015)

iOS App Development for Dummies (Wiley, 2014)

iWork for Dummies (Wiley, 2012)

Videos include:

Mixed Language App Development with Objective-C and Swift (O'Reilly, 2015)

iOS Developer's Guide to Views and View Controller (O'Reilly, 2015)

Learning Objective-C Programing (O'Reilly, 2015)

He is heard regularly on WAMC Public Radio for the Northeast's The Roundtable, and is the founder of Friends of Saranac River Trail. A native of Washington DC, he has lived in New York City and currently lives in Plattsburgh NY.

He can be reached at `northcountryconsulting.com` (consulting) and `champlainarts.com` (app development).

About the Technical Reviewers

Trained in computer engineering, multimedia, and graphic design, **Aaron Crabtree** has spent the last 15 years working in the industry. Experience has taught Aaron that, in most cases, programmatically creating each piece of an application is necessary for greater flexibility and control. Drop him a line on Twitter: `@aaron_crabtree`

Cliff Wootton is a former Interactive TV systems architect at BBC News. The "News Loops" service developed there was nominated for a BAFTA and won a Royal Television Society Award for Technical Innovation. An invited speaker on pre-processing for video compression at the Apple WWDC conference. Taught post graduate MA students about real-world computing, multimedia, video compression, metadata and researching the deployment of next generation interactive TV systems based on open standards.

Currently working on R&D projects investigating new Interactive TV technologies, involved with MPEG standards working groups, writing more books on the topic and speaking at conferences when not lecturing on Multimedia at the University of the Arts in London.

Acknowledgments

What I love most about working with databases is putting the pieces of the puzzle together – what a friend of mine used to call "making order out of chaos." When it comes to thinking about SQLite, there are many pieces to put together – everything from the first work on relational databases back in the 1960s to the idea of lightweight code libraries that can fit appropriately onto heavy-duty computers, as well as small battery-powered mobile phones and autonomous satellites designed to explore other worlds far from the relative safety of their home planet.

This book has been made possible, first and foremost, by D. Richard Hipp who began (and continues) the SQLite project starting in 2000. The many people who have added and tested code are too many to thank, but you know who you are.

Closer to home, Jeff Pepper at Apress has been great to work with (again), as has Mark Powers. Together with Carole Jelen at Waterside Productions, they have all made it possible to provide this basic introduction to SQLite.

Introduction

I have worked with databases of many types on many platforms. Despite the fact that almost all of them have been based on the relational model. I've had some excursions into ancient database structures such as IMS (a hierarchical database) and, more recently into unstructured data, but in every case it has been a matter of a basic principle: finding the simplest way to organize data so that it makes sense. (As I often say to clients, "The data doesn't lie." When you lay data out this way, that way, and every which way, I've found that gradually its inherent structure becomes clear – if there is one. If there isn't an inherent structure, what may become clear is some structure no one has ever thought of for the data. And sometimes, the data stubbornly refuses to reveal a structure that we can understand.

This may sound very highfalutin and esoteric, but I do think that our job as database designers ultimately becomes a job of finding patterns. If we cannot find a logically inherent and inevitable pattern, the best choice is to find the best pattern that is usable for the task at hand.

We can't produce these data models without knowing something (a lot, in fact) about the ways to organize data, and, today that comes down to the relational model and the tools of SQL. This process is iterative (woe betide the design team that adopts a partial database model too early), and it relies on flexible tools as well as imagination and constant questioning of everyone working on a project.

SQLite has become a critical component of many data modeling projects because it is so lightweight and flexible. You can put up a SQLite database with basic functionality very quickly. That's where this book comes in. If you have the data and the mission to come up with something that uses the data – website, mobile app, or standalone software product – SQLite can be just what you want.

SQLite runs on many operating systems. With SQLite, you're working with a database that you can move around from one platform to another and one development environment to another. Your prototyping, experimentation, and development can proceed without committing yourself to anything besides the relational model and SQLite. You may move on to other relational tools, and your little SQLite app may turn into a production app for a multitude of devices or for a single many-multi-threaded corporate data center.

This flexibility and very low barrier to entry are two of the features that many people including me find attractive in SQLite.

I hope that you'll find this book useful. There's more information on SQLite at `sqlite.org`, and there's more information (and downloadable examples from the book) on my website at `northcountryconsulting.com`.

—Jesse Feiler

CHAPTER 1

Getting Up to Speed with Databases and SQLite

Ask people to tell you words they associate with *database* and you'll probably get *big* among the responses. Databases handle large amounts of data, and everyone knows that. How big? It depends whether you count the number of items in a database or the size of the items. The Library of Congress has 160 million items on 858 miles of shelving as of this writing. The catalog, which is based on a database, is available online at `www.loc.gov/about/fascinating-facts/`. There's no question among people who are interested in databases that the Library of Congress catalog is far from the largest database in the world. (In all likelihood, the largest databases are not visible to the public because they contain classified corporate and governmental information.)

This chapter provides an introduction or reminder about databases today and how they are used. If you've used databases in the past, much has changed, and if you haven't used them much in the past, this chapter will give you a quick overview.

Moving Beyond *Big*

Beyond *big*, you should start thinking about databases as being *structured* and *organized*. In fact, as a mobile developer, the structured and organized aspects of databases are much more important than their size. From the earliest days of the Web in the early 1990s, web browsers have used databases to store and organize their data. What data, you may wonder? How do you think your browser is able to store your passwords for the web sites you visit? How do you think browsers store web pages in a cache so that they can be retrieved without a new network access when possible? And what about your preferences for default font sizes? Or the download folder for files you can change when you feel like it, and even, on many browsers, restrictions on certain users or types of users? These are just simple examples, but the principles apply even to the largest (and smallest) databases today.

Databases Are Structured and Organized

All those examples can be implemented with a database because all those items need to be structured and organized. Some of the items can turn into large amounts of data (particularly a browser history if you don't choose an option to clear out the old information occasionally—another option that can be stored in a preference database), but some of the preferences listed are very small amounts of data: even with a fully modified file name for your download folder, 128 characters is enough to store that data element. Furthermore, most browsers store the name of your preferred download folder, but they don't store the history of download folders (last week's preferred download folder, the folder you used last month, etc.).

In thinking about databases, "big" is often relevant, but very frequently it's misleading. It is structure and organization that matter most.

And they matter a great deal to a mobile developer.

Precisely because mobile devices need to function in a world of constrained resources (e.g., limited storage space and battery power), structuring and organizing the data that is stored is particularly important. Built into a data management system are various optimizations that matter to databases large and small. For example, many databases can store character or string data. Such data is typically quite variable. Even if a database designer specifies that a text field can contain 50, 500, or 50,000 characters, behind the scenes trailing blanks are often discarded. The user (and even the developer) may never know this, but it makes the entire database function more efficiently. (Features like this function behind the scenes and they can be accessed and turned on or off by the database designer in many cases.)

Databases Are Smart

Being able to optimize storage to work with the presence or absence of characters in a text field is a powerful behind-the-scenes tool, but databases have intelligence built into them.

In most database engines today, the designer can specify many attributes of a field such as the following:

- *Name*. A name that is used internally is often quite different from the name that appears in the user interface.
- *Type*. You can specify that a certain database field must be an integer or a string or any other type that the database supports.
- *Optionality*. Some fields are optional, and you can specify that when you set up a database. (Many people have a car with a license number; many other people do not own cars.)
- *Default values*. A database can be set to provide a default value—either a simple value or a calculated value based on other data in the database.