The Minimum You Need to Know

About Qt and Databases

By Roland Hughes

Logikal Solutions

ISBN-13 978-0-9823580-5-4

This book was published by Logikal Solutions for the author. Neither Logikal Solutions nor the author shall be held responsible for any damage, claim or expense incurred by a user of this book as a result of its use or reliance upon.

Acknowledgments

I'd like to thank my loyal base of readers and the incredible core team of Qt developers slaving away on an Open Source library that stands full torso above the commercial C++ cross platform libraries of old.

Source Code License

Any person owning a copy of this book may use the source code from this book freely when developing software for their personal use, their company's use, or their client's use. Such persons may include the source code either modified or unmodified provided that the source delivered makes reference to the original author and is part of a fully functional application. It is expressly forbidden for anyone to post this source code on any bulletin board system, Internet Web site, or other electronic distribution medium other than SourceForge.net without the express written permission of the author. It is also expressly forbidden to sell this source as part of a library or shareware distribution of source. All files for all projects presented in this book will be available from http://www.theminimumyouneedtoknow.com

Users of the source code contained within this book agree to hold harmless both the author and the publisher for any errors, omissions, losses, or other financial consequences resulting from the use of said source. The source code is supplied “as is” with no warranty of any kind expressed or implied.

Table of Contents

Chapter 1

Introduction

1.1 Where We've Been and Where We Are Going

Hopefully you've been reading along in this book series and are quickly progressing as a developer or at least gaining a significant appreciation for what developers do. While you do not need to have read the rest of the series, it might help. The source code for the PostgreSQL expense tracking example in this book has been uploaded to SourceForge.net and can be found in the xpnsqt project. All source files will be zipped up and placed on http://www.theminimumyouneedtoknow.com for download as well. Unlike most books in this series, there will not be an accompanying CD-ROM.

While I did not write the books in exactly this order, the correct order of reading would have been:

The Minimum You Need to Know About Logic to Work in IT

ISBN-13 978-0-9770866-2-7

The Minimum You Need to Know to Be an OpenVMS Application Developer

ISBN-13 978-09770866-3-4

The Minimum You Need to Know About Java on OpenVMS

ISBN-13 978-0-9770866-1-0

The Minimum You Need to Know About Service Oriented Architecture

ISBN-13 978-0-9770866-7-2

The Minimum You Need to Know About Java and xBaseJ

ISBN-13 978-0-9823580-3-0

Infinite Exposure

ISBN-13 978-0-9770866-9-6

Originally I wrote the OpenVMS Application Developer book to fill a need. Lots of places run OpenVMS and new installations are happening all of the time, yet there is no source for new OpenVMS developers. It wasn't until after that book was published that I started getting feedback about recent college graduates not understanding logic. At that point I decided to make this a series and began writing the logic book.

With the logic and traditional application development books out in the field there was a natural progression. First we covered Java on OpenVMS using the same application we had developed in COBOL, BASIC, FORTRAN, C and C++ in the application book. It was necessary to cover accessing of RMS files and RDB databases via Java along with using FMS as your screen management tool. Seasoned developers wanted to learn some of the new technology and recent college grads only understood new technology without even the faintest clue about how to access the data which actually ran the company.

Next we stepped forward with Java on the Ubuntu platform accessing services presented by OpenVMS on an internal company network. Some of you might be shocked to learn that “The Minimum You Need to Know About Service Oriented Architecture” won a 2008 Best Book Award in the category Business: Computers/Technology/Internet from USA Book News and was a 2009 Eric Hoffer Finalist. There are a lot of books about Service Oriented Architecture on the market, but damned few actually cover exposing the heritage data silos. They usually try to sell you a chunk of middleware and another never ending support contract along with shiny new security holes.

The next book in the series didn't follow a logical progression per se, it came about based upon need. I was in the process of designing and writing a fuel surcharge application which would eventually be released as a Open Source project and should be usable by someone who isn't any smarter than a Microsoft product. That last requirement meant the application had to use its own indexed file system rather than mandate a user install, configure, and become a DBA for some database product.

The XBASE family of file formats are the most widely used throughout the history of PC computing, but the Java language doesn't directly include anything to support it, nor does it include any other usable indexed file system. (Yes, there is now a massive development effort called Java DB, but not at the time I needed it.) That meant I had to learn a library. I decided that if I had to take the time to learn a new library with scarce documentation that I would write a tutorial and make life easier for the next guy.

Unlike the previous books in this series, the xBaseJ book is available in electronic form only. You can find the PDF with the files for the project on SourceForge. You can also find it listed here:

http://www.free-ebooks.net/ebook/The-Minimum-You-Need-to-Know-About-Java-and-xBaseJ

From an educational perspective, I need to provide you with a book which uses a popular cross platform C++ GUI tool to do some classic standalone development against a database server. Many in the world of IT will call this "Client-Server" development even though both the client and the server are on the same machine. Historically, this type of development occurred prior to anyone even thinking about inventing the Internet, but modern education wise, it is best to cover this topic *after* covering Service Oriented Architecture.

Why?

Kids today know very little about application development, but they know a lot about surfing the Web and buying things on-line. It is easier for them to grasp a process which involves some kind of Web page sending data to some back end system (Service Oriented Architecture.) Only a few of them will have had to work with Quicken or some other personal finance/accounting application hosted on their own computer. A student who has actually used an application on a desktop computer which stores data on a remote database without going through the Internet will be a rare find indeed.

"Infinite Exposure" is quite simply a novel which was written in response to an interview question I got for the OpenVMS Application book. While it has gotten some great reviews, it is not part of my technical book writing.

Some of the previous books in this series used the Mega Zillionare application developed over and over again so you could learn the differences and trade-offs associated with each tool choice. I could bore the living Hell out of you doing that again, but we don't have different tools, only different databases. I'm going to have to deviate from the norm and develop a new application for multiple databases to show you the main issues. Yes, Qt does quite a good job of hiding the database *if you happen to be using a relational database*, but in most cases, you cannot simply change the driver name-recompile-and-go. There are quite a few differences between supported databases.

Don't worry, I know that most of you will be GUI only developers at best and cannot begin to function without the continual hand holding of an IDE (Integrated Development Environment.) While I developed the PostgreSQL version of our application with nothing more than a text editor and the Qt Designer for form layout, our later examples will include using multiple IDEs and provide quite a few project creation screen shots. You will be exposed to QtCreator, QDevelop, Monkey Studio, and Eclipse with the Qt plugin.

Our application will be an expense tracking system which initially uses the PostgreSQL database. We will then redevelop the application using Firebird (Borland Interbase) and SQLite. I will spend one chapter covering the use of stored procedures in PostgreSQL and Firebird. Our final technical chapter will cover problems with long queries and threading. It is too bad Oracle hasn't created a Qt plugin to access RDB or I could show you how to use a modern desktop with the world's highest availability database.

We won't cover MySQL. I know the database is popular. I know many of you will want to use it, but it's quite the quagmire right now. By default, it is not a relational database. MyISAM is the default storage engine and is now the only storage engine for the embedded version. InnoDB was purchased by Oracle some time ago and Oracle has started to squeeze that orange.

http://digitizor.com/2010/11/05/innodb-dropped-from-oracle-mysql-classic-edition/

While InnoDB is still available in the “community” edition, the fact it has been dropped from the Classic edition and InnoDB now has its own embedded/embeddable version seems to indicate the “community” version of MySQL is going to be trapped with a stagnant InnoDB release.

MyISAM creates a lot of data storage and application design problems. It doesn't currently have a boolean type completely implemented and there is no support for foreign key constraints. While the Boolean problem could be considered a nuisance, the lack of foreign key constraints really torpedoes this project. What good are reference tables containing categories and expenses if the database isn't going to enforce referential integrity? I went through this exact same problem eons ago when I used XBASE files for each table. Without an engine, I had to enforce the integrity from the application level. Not such a good thing.

1.2 Not a GUI Tutorial

This book will not be a GUI tutorial nor will it spend much time covering areas of the Qt library which do not pertain to our sample application's database portions. While it is true that many developers could use this as a Qt tutorial along with the manuals Trolltech provides on-line, that is not the goal of this book.

If you need additional information on the graphical portions of the library along with the philosophy behind it all I can recommend is a pair of excellent books.

“Foundations of Qt Development” ISBN-13: 978-1-59059-831-3

“C++ GUI Programming with Qt 4 Second Edition” ISBN-13: 978-0-13-235416-5

In truth, this book is meant to cover the one area where those books really fell down. It's sad because it is the one area most developers actually need to understand. Very few of us get paid to write painting and drawing applications. The bulk of business development revolves around getting data into a database and getting reports out of the database.

1.3 Client-Server Full Circle.

The Web doesn't live up to its hype for most applications. In many cases you want control, security, and limited deployment. In some cases you simply want a standalone application or an application that you control access via controlling where it gets installed. You don't want information about your new super-secret billion dollar R&D effort some place it could easily be poached. You want that application secured on its own internal network or machine without any connection to the Internet at all.

Standalone programs on standalone personal computers make sharing data a bit tough. Without a network and exposing some portion of your disk storage to outsiders, the options for sharing data are limited to physical media transfers along with dial-and-deliver options.

Most of your successful standalone development happened with midrange and mainframe computers where hundreds to thousands of terminals were scattered about a company, but they all connected into a single computer. This made exposing the data as simple as granting a user access to the files or databases.

During the late 1980s through the mid-1990s Client-Server was both the rage, and the bane of computing. Nobody was ever able to get it right, but everybody kept trying. The real problem with client-server was application deployment. We didn't have any automated methods of “pushing” updates out to users and most users didn't have the technical ability to perform an installation on their own.

When corporate desktops all ran some form of non-GUI DOS, file servers seemed to be the ultimate solution. Products like Netware provided security, the ability to address much more disk storage than most versions of DOS, and even provided an indexed file system which handled multi-user access. As long as the desktop computer trying to run the application didn't have too much of that precious 640K already consumed, they could run the executables a user could access.

Users wanted more and more sophisticated applications. Developers wanted to use different tools which did more and more of the programming work for them. These tools had various licensing systems which invariably involved installing some kind of encrypted license key file along with a run-time library (RTL) of some kind. While the file server could still be a great place to host the installation files and the actual data created by the application, the days of having a single executable file which everybody could run were pretty much over.

Another drastic thing happened near the end of the 1990s: corporate desktops started to diversify. While a great many people will be ignorant enough to say that Microsoft rules the corporate desktop, they would be wrong. Apple started making some serious inroads due to things the platform could do which Microsoft simply could not. After the disastrous release of Windows Vista, many corporations started loading various flavors of Linux on their company desktops as well as their servers.

Currently, the PC desktop industry is in a state of turmoil. This turmoil will exist for roughly another decade, but it will be good for the industry in general. All but the most hardened supporters of Microsoft have realized the company has a legacy platform which is in the mode of continually shrinking market share. Apple has a knack when it comes to getting people to pay a premium for their stuff, and recently, per the stock market, became bigger than Microsoft on a total_shares * share_price basis. Another player gaining a lot of momentum and respect are the various Linux distributions. After Windows Vista, anything looked stable, so companies gave it a try. Then the economy tanked world-wide. Suddenly company heads started looking at the annual licensing and support costs of Microsoft products and running the numbers against the Linux and Open Source alternatives. When the savings started being $20-$120 per employee per year the 10,000+ employee companies started taking note and issuing marching orders.

The turmoil is what lead to the SOA (Service Oriented Architecture) stampede. Some companies needed to get custom applications working on two to three versions of Windows as well as the current Apple OS and three to four different Linux distributions. Rather than fight the cross platform battle they lunged forward at the idea of providing Web-based applications which used Java and other Open Source tools that

ran in/on the major browsers for each operating system.

A great deal of effort was put forth by the W3C and many others to make SOA work. A method of deploying services for discovery via WSDL (Web Services Description Language) was created so any application anywhere could find what it needed and simply pluginto the service. The problem was much of this design centered around openness, not privacy and security.

It never ceases to amaze me when young MBAs refuse to write a new application on OpenVMS or even an IBM mainframe. PCs are all they have ever seen, therefore, that must be all there is to use. They are so dead set against creating a “green screen” application which would be able to restrict access to only those who should really see, that they bring us full circle, back to the mid-1980s and cross platform development.

1.4 Today's Cross Platform Challenge.

In today's world, cross platform has a slightly different definition. It used to just mean different operating systems on the desktop, but now it extends to handheld devices like netbooks, PDAs (Personal Digital Assistant), and smart phones. In truth, the distinction between cellular phone, PDA, and netbook is becoming blurred by the continual addition of features to smart phones and the emergence of the iPad. Companies are making wide use of VPNs (Virtual Private Networks) on handheld devices so personnel in the field can run applications which access databases secured behind the corporate firewall. They have a need to maintain one application yet have it run on every desktop, netbook, and PDA deployed by the company. Let us not forget you now have to deal with virtual keyboards and mice on most handheld devices.

I'm no stranger to cross platform development. I was there when the first major plunge happened. Some of you might be old enough to remember the “Zinc It!” book series. It covered the Zinc Application Framework on DOS, OS/2, Windows, and to some extent MAC. The reason I know about that series is I was the author. Zinc Software was a poorly managed company. The product was purchased by Wind River Systems and rolled into various products they offer. Recently, perhaps because of this trend, I have found Zinc is once again being sold as a product by a company. I haven't dug too deeply into it, but they claim to support desktop and embedded platforms.

There is also talk of an Open Source version as well.

Let's be honest. While some commercial product vendors will use the cross platform capabilities of the library, many in-house developers will simply use the library on a single platform with an eye toward being able to migrate quickly if whims change. A lot of companies which had been staunchly (and stupidly) in the Windows desktop category are now starting to use Qt for Windows development. They are in the process of migrating everything to Qt so they can deploy Ubuntu, OpenSuSE, or some other Linux distribution on the desktop as soon as the bean counters realize just how much it costs to keep Microsoft around. Microsoft has been a victim of its own marketing now. When you actually total up *all* of the costs, it ends up being the most expensive platform per user of *any* computing platform, including IBM mainframes and OpenVMS midrange systems. A lot of companies have learned that the OpenVMS box which is still housing their data and some core applications is/was the cheapest per user solution they had even if they didn't like green screen applications.

1.5 Why Ubuntu?

If you have already read “The Minimum You Need to Know About Service Oriented Architecture”, then you aren't asking the above question. For those of you who skipped that book in the series, Ubuntu is quickly becoming the desktop Linux distribution of choice for most IT professionals and even quite a few corporations. Dell has even started pre-loading Ubuntu on machines for corporate customers. Yes, Lenovo is pre-loading OpenSuSE, but most other vendors are embracing Ubuntu. OpenSuSE has been tainted by a business deal between Novell and Microsoft which infuriated most of the various Unix/Linux factions and spawned a re-write of the GPL (GNU Public License) which put Novell in a world of hurt.

We won't get too deep into arguments about which is technically better. I have used both in the past. I used to actually pay for the professional edition of SuSE before it got the silent “Open” added to the front of it. All code in this book will be written and tested on version 10.10 AMD 64-bit KUbuntu. You should be able to adapt things for your own environment. I can warn you that many versions of OpenSuSE made it nearly impossible to get PostgreSQL running. It would eventually run, but you had to scour