# Welcome

Call it stage three of the computer revolution. Stage one worked out basic computing principles and applied them to mainframes to help run the world. That was pretty much in hand by 1980. Stage two turned those principles into software we could install on a desktop or laptop for a reasonable price. That was in hand by 1995. Stage three brings networks of billions of desktops and laptops and mobiles, and *free open source software development tools for any computer task you can imagine.*

As of 2019, software was a $457 billion industry, about 10% of it database software. MySQL® is one such tool. It runs a majority of the world's database-driven websites. In most market share surveys it ranks first or second, just ahead of or behind SQL Server. MySQL leads all open source databases in market share, and that share is now very nearly equal to closed-source databases. Version 5.0 came into production in 2005, 5.1 came in 2008, 5.5 in late 2011, 5.6 in early 2013, 5.7 in late 2015, 8.0 in 2018. All versions before 5.7 have been archived.

MySQL is fast, reliable, flexible, richly featured, customisable. It's built for developers, it's robust, and it's open source. Under a GPL/GNU licence it costs you exactly nothing. MySQL queries typically return results in less than 0.05 seconds. Well-designed multi-terabyte databases perform well without custom tweaks like sharding. Replication is straightforward. The relational InnoDB storage engine is robust. So is the server overall: Google engineer Jeremy Cole found a Twitter MySQL installation that had been running 212 days without downtime, over which time it had processed an average of 6.9 queries per second, returning about 1.4 million data rows per second.

Fine. But if you're a database developer, you probably already know about MySQL. Why another book about it?

Three answers: about recent MySQL versions, about our approach, and about you.

*About recent versions:* In 5.0 and 5.1, correlated subqueries, stored routines including Triggers, Views, and an `information_schema` implementation took MySQL well beyond the one-app-one-database model of early versions. 5.5 and 5.6 made transaction management more robust, added tools, and improved performance as much as 200%.. 5.7 improved security, logging, Triggers and Views; introduced JSON and derived columns. 8.0 improves REGEX, moves system tables and the global data dictionary to InnoDB; revamps partitioning; enhances query parsing and indexing; it adds component-based infrastructure, SQL roles, `utf8mb4` collation, Common Table Expressions (CTEs), windowing functions; persistent dynamic global variables, and function-based indeses. If you rather work with a MySQL founder than with Oracle Corp., there's Monty Widenius's MariaDB, mostly compatible with MySQL except for its *many additional features*.

*About our approach:* MySQL's original focus was on a small footprint and high performance, sometimes at the cost of relational database correctness. MySQL databases tended

to be maintained from single web sites. You can't argue with what MySQL AB accomplished, but more MySQL users began to ask for big-three-style relational power. In short order MySQL AB declared it was aiming at SQL:2003 compatibility, was bought by a big software company (Sun), which in 2009 was itself bought by a big-three database company, Oracle.

So in this book, instead of beginning with how to do up a limited MySQL project as quickly as possible, we ground design on use case analysis, we ground syntax on relational database theory, and we ground database maintenance on software life cycle analysis. Most everything else follows from those fundamentals. You spend a bit more time up front on the basics, and pretty soon you find you can drive the vehicle a lot faster, more steadily, to more interesting destinations.

*About you:* You may be new to MySQL. You may be a Windows/SQL Server developer or manager who now, for the first time, needs to go to Linux or MySQL.

You may have decided that for you or your working community, the time for database systems from huge corporations costing tens of thousands of dollars has passed, that the time for community-developed open-source software development has arrived.

You may work for an NGO with a certain amount of in-house skill and an insistence on being honest, but now lack the funds for expensive licences. You may have worked mostly on Oracle or SQL Server or Sybase or even FoxPro systems, but your next project specifies MySQL. Whatever the reason, you are, or soon will be using a major open-source RDBMS *which is changing as you use it, on demand from you.*

Traditional dead-tree publishing offers a 500-1000 page tome of which maybe 200 pages are relevant to you. Some of it is out of date by the time you read it. So you lug home this two-kilogram object, read the pages of interest, curse the out-of-dateness, and ignore the rest. Trees died, trucks lugged heavy pallets of books to bookstores, more $CO_2$ made your summer hotter, and you weren't all that satisfied.

Much of the world has moved on from that publishing model. A few chapters of this book are free. You can purchase the whole book as an inexpensive weightless download, with an optional five-year subscription for updates. Why isn't it on a Kindle or an iPad? We want to update it as often as MySQL changes; we want *you*, not Amazon or Apple, to own your copy; and we want you to be able to update your copy as often as you please.

From the invention of writing, through long mediaeval queues of copying monks, Gutenberg's printing press, industrial and mass-market publishing, to the Web and to current efforts to put all books ever written online, we can't be sure what will happen next—except that in a few years you'll find this book freely available on the Web just as surely as you can now find most any other book, written in English, in a library at Oxford or in the New York City Public Library. If that wrecks our current business model, so be it. Meanwhile, you're reading this because you've joined the knowledge race. There's no other marathon like it.

*Peter Brawley & Arthur Fuller*