Preface

Distributed database systems are an integral part of most businesses and the vast majority of software applications. These applications provide logic and a user interface, while database systems take care of data integrity, consistency, and redundancy.

Back in 2000, if you were to choose a database, you would have just a few options, and most of them would be within the realm of relational databases, so differences between them would be relatively small. Of course, this does not mean that all databases were completely the same, but their functionality and use cases were very similar.

Some of these databases have focused on *horizontal scaling* (scaling *out*)—improving performance and increasing capacity by running multiple database instances acting as a single logical unit: Gamma Database Machine Project, Teradata, Greenplum, Parallel DB2, and many others. Today, horizontal scaling remains one of the most important properties that customers expect from databases. This can be explained by the rising popularity of cloud-based services. It is often easier to spin up a new instance and add it to the cluster than scaling vertically (scaling *up*) by moving the database to a larger, more powerful machine. Migrations can be long and painful, potentially incurring downtime.

Around 2010, a new class of *eventually consistent* databases started appearing, and terms such as *NoSQL*, and later, *big data* grew in popularity. Over the last 15 years, the open source community, large internet companies, and database vendors have created so many databases and tools that it's easy to get lost trying to understand use cases, details, and specifics.

The Dynamo paper [DECANDIA07], published by the team at Amazon in 2007, had so much impact on the database community that within a short period it inspired many variants and implementations. The most prominent of them were Apache Cassandra, created at Facebook; Project Voldemort, created at LinkedIn; and Riak, created by former Akamai engineers.

Today, the field is changing again: after the time of key-value stores, NoSQL, and eventual consistency, we have started seeing more scalable and performant databases, able to execute complex queries with stronger consistency guarantees.

Audience of This Book

In conversations at technical conferences, I often hear the same question: "How can I learn more about database internals? I don't even know where to start." Most of the books on database systems do not go into details of storage engine implementation, and cover the access methods, such as B-Trees, on a rather high level. There are very few books that cover more recent concepts, such as different B-Tree variants and log-structured storage, so I usually recommend reading papers.

Everyone who reads papers knows that it's not that easy: you often lack context, the wording might be ambiguous, there's little or no connection between papers, and they're hard to find. This book contains concise summaries of important database systems concepts and can serve as a guide for those who'd like to dig in deeper, or as a cheat sheet for those already familiar with these concepts.

Not everyone wants to become a database developer, but this book will help people who build software that uses database systems: software developers, reliability engineers, architects, and engineering managers.

If your company depends on any infrastructure component, be it a database, a messaging queue, a container platform, or a task scheduler, you have to read the project change-logs and mailing lists to stay in touch with the community and be up-to-date with the most recent happenings in the project. Understanding terminology and knowing what's inside will enable you to yield more information from these sources and use your tools more productively to troubleshoot, identify, and avoid potential risks and bottlenecks. Having an overview and a general understanding of how database systems work will help in case something goes wrong. Using this knowledge, you'll be able to form a hypothesis, validate it, find the root cause, and present it to other project maintainers.

This book is also for curious minds: for the people who like learning things without immediate necessity, those who spend their free time hacking on something fun, creating compilers, writing homegrown operating systems, text editors, computer games, learning programming languages, and absorbing new information.

The reader is assumed to have some experience with developing backend systems and working with database systems as a *user*. Having some prior knowledge of different data structures will help to digest material faster.

Why Should I Read This Book?

We often hear people describing database systems in terms of the concepts and algorithms they implement: "This database uses gossip for membership propagation" (see Chapter 12), "They have implemented Dynamo," or "This is just like what they've described in the Spanner paper" (see Chapter 13). Or, if you're discussing the algorithms and data structures, you can hear something like "ZAB and Raft have a lot in common" (see Chapter 14), "Bw-Trees are like the B-Trees implemented on top of log structured storage" (see Chapter 6), or "They are using sibling pointers like in B^{link}-Trees" (see Chapter 5).

We need abstractions to discuss complex concepts, and we can't have a discussion about terminology every time we start a conversation. Having shortcuts in the form of common language helps us to move our attention to other, higher-level problems.

One of the advantages of learning the fundamental concepts, proofs, and algorithms is that they never grow old. Of course, there will always be new ones, but new algorithms are often created after finding a flaw or room for improvement in a classical one. Knowing the history helps to understand differences and motivation better.

Learning about these things is inspiring. You see the variety of algorithms, see how our industry was solving one problem after the other, and get to appreciate that work. At the same time, learning is rewarding: you can almost feel how multiple puzzle pieces move together in your mind to form a full picture that you will always be able to share with others.

Scope of This Book

This is neither a book about relational database management systems nor about NoSQL ones, but about the algorithms and concepts used in all kinds of database systems, with a focus on a *storage engine* and the components responsible for *distribution*.

Some concepts, such as query planning, query optimization, scheduling, the relational model, and a few others, are already covered in several great textbooks on database systems. Some of these concepts are usually described from the user's perspective, but this book concentrates on the internals. You can find some pointers to useful literature in the Part II Conclusion and in the chapter summaries. In these books you're likely to find answers to many database-related questions you might have.

Query languages aren't discussed, since there's no single common language among the database systems mentioned in this book.

To collect material for this book, I studied over 15 books, more than 300 papers, countless blog posts, source code, and the documentation for several open source

databases. The rule of thumb for whether or not to include a particular concept in the book was the question: "Do the people in the database industry and research circles talk about this concept?" If the answer was "yes," I added the concept to the long list of things to discuss.

Structure of This Book

There are some examples of extensible databases with pluggable components (such as [SCHWARZ86]), but they are rather rare. At the same time, there are plenty of examples where databases use pluggable storage. Similarly, we rarely hear database vendors talking about query execution, while they are very eager to discuss the ways their databases preserve consistency.

The most significant distinctions between database systems are concentrated around two aspects: how they *store* and how they *distribute* the data. (Other subsystems can at times also be of importance, but are not covered here.) The book is arranged into parts that discuss the subsystems and components responsible for *storage* (Part I) and *distribution* (Part II).

Part I discusses node-local processes and focuses on the storage engine, the central component of the database system and one of the most significant distinctive factors. First, we start with the architecture of a database management system and present several ways to classify database systems based on the primary storage medium and layout.

We continue with storage structures and try to understand how disk-based structures are different from in-memory ones, introduce B-Trees, and cover algorithms for efficiently maintaining B-Tree structures on disk, including serialization, page layout, and on-disk representations. Later, we discuss multiple variants to illustrate the power of this concept and the diversity of data structures influenced and inspired by B-Trees.

Last, we discuss several variants of log-structured storage, commonly used for implementing file and storage systems, motivation, and reasons to use them.

Part II is about how to organize multiple nodes into a database cluster. We start with the importance of understanding the theoretical concepts for building fault-tolerant distributed systems, how distributed systems are different from single-node applications, and which problems, constraints, and complications we face in a distributed environment.

After that, we dive deep into distributed algorithms. Here, we start with algorithms for failure detection, helping to improve performance and stability by noticing and reporting failures and avoiding the failed nodes. Since many algorithms discussed later in the book rely on understanding the concept of leadership, we introduce several algorithms for leader election and discuss their suitability.

As one of the most difficult things in distributed systems is achieving data consistency, we discuss concepts of replication, followed by consistency models, possible divergence between replicas, and eventual consistency. Since eventually consistent systems sometimes rely on anti-entropy for convergence and gossip for data dissemination, we discuss several anti-entropy and gossip approaches. Finally, we discuss logical consistency in the context of database transactions, and finish with consensus algorithms.

It would've been impossible to write this book without all the research and publications. You will find many references to papers and publications in the text, in square brackets with monospace font; for example, [DECANDIA07]. You can use these references to learn more about related concepts in more detail.

After each chapter, you will find a summary section that contains material for further study, related to the content of the chapter.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Database Internals* by Alex Petrov (O'Reilly). Copyright 2019 Oleksandr Petrov, 978-1-492-04034-7."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning

For almost 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, conferences, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, please visit *http://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/database-internals*.

To comment or ask technical questions about this book, please send an email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

This book wouldn't have been possible without the hundreds of people who have worked hard on research papers and books, which have been a source of ideas, inspiration, and served as references for this book.

I'd like to say thank you to all the people who reviewd manuscripts and provided feedback, making sure that the material in this book is correct and the wording is precise: Dmitry Alimov, Peter Alvaro, Carlos Baquero, Jason Brown, Blake Eggleston, Marcus Eriksson, Francisco Fernández Castaño, Heidi Howard, Vaidehi Joshi, Maximilian Karasz, Stas Kelvich, Michael Klishin, Predrag Knežević, Joel Knighton, Eugene Lazin, Nate McCall, Christopher Meiklejohn, Tyler Neely, Maxim Neverov, Marina Petrova, Stefan Podkowinski, Edward Ribiero, Denis Rytsov, Kir Shatrov, Alex Sorokoumov, Massimiliano Tomassi, and Ariel Weisberg.

Of course, this book wouldn't have been possible without support from my family: my wife Marina and my daughter Alexandra, who have supported me on every step on the way.