西 南 交 通 大 学

本科毕业设计（外文翻译）

智慧库房Web端前后台设计与实现

年 级： 2016 级

学 号： 2016112385

姓 名： 李松

专 业： 计算机科学与技术

指导教师： 陈剑波

二零二零年五月

**介绍 MySQL 8 文档存储**

查尔斯·贝尔

**介绍**

近年来，NoSQL被大肆炒作。与大多数新技术一样，基本原理很少是真正新的，而是已知形成和转换整体的技术，使其超过其部分的总和。对于MySQL 8和新的MySQL文档存储尤其如此。MySQL从未为这么多人提供这么多。无论您是想要具有牢固基础的传统关系数据库解决方案，还是希望最终灵活地将JSON文档存储在文档存储中，还是介于两者之间的任何位置，MySQL都可以做到这一点。

诀窍是学习如何使用以下每种技术迁移应用程序：无论是使用具有固定架构的传统表，还是使用一些JSON字段允许从刚性结构中获得一些自由，或者您采用基于JSON的文档存储。

本书将为您提供浏览MySQL文档所寻求的知识存储，包括如何迁移现有应用程序以及使用文档存储解决方案。

**目标受众**

我写这本书是为了分享我对新的MySQL文档存储和Python的热情。我特别想展示任何人都可以编写文档存储解决方案，而无需投资学习大型、复杂的语言和发展环境。因此，目标受众包括任何有兴趣了解MySQL文档存储的人，例如数据库管理员、开发人员以及信息技术经理和战略规划人员。

**本书的结构**

本书的编写旨在指导读者从MySQL 8中新功能的一般知识到构成MySQL文档存储的组件的详细说明。前几章包括一般主题，包括MySQL 8的简要介绍、如何安装MySQL 8以及如何配置新的文档存储组件。后面的章节将更详细地介绍MySQL文档存储组件，包括MySQL外壳、X DevAPI、X插件等。这些章节下面是一对章节，它们提供了一个示例应用程序，用于将信息存储在作为关系数据库、混合数据库和文档存储编写的书籍上的信息。因此，您可以看到如何通过这些变体迁移单个应用程序。本书最后提供了一些有关升级到MySQL 8的说明以及使用MySQL文档存储的最佳做法，包括迁移过程的回顾。以下是本书中每一章的简要概述。

第1章，"介绍MySQL 8\_一个新的开始"：本章探讨新的MySQL服务器版本8.0的一些亮点。您将发现那些最初在早期版本中引入的功能已适应版本8.0的新范例、新的功能以及真正革命性的新功能，如文档存储、组复制和InnoDB群集。

第2章"开始使用MySQL"：本章介绍MySQL教程，讨论使用SQL接口在传统角色中使用MySQL数据库服务器的功能;如何发出用于创建用于存储数据的数据库和表的命令以及用于检索该数据的命令。尽管本章在MySQL上只提供了一个小入门，但您将学习如何开始安装MySQL。

第3章"JSON文档"：本章更详细地探讨JSON数据类型。您将看到如何通过MySQL中提供的大量内置JSON函数在关系表中使用JSON数据的示例。JSON数据类型是允许用户开发跨越SQL和NoSQL应用程序范围的混合解决方案的关键。

第4章"MySQL外壳"：本章演示如何使用MySQL外壳，包括查看启动选项、shell命令、连接、会话，甚至我们如何在JavaScript和Python中执行一些交互式脚本。因此，本章是学习如何开始使用MySQL外壳和使用JSON和关系数据的关键章节。尽管本章并非详尽地介绍MySQL外壳的所有功能，但它提供了有关如何将其用于最常见的任务的广泛教程。

第5章"X开发人员API"：本章探讨X DevAPI，并探讨可用于连接到MySQL服务器、创建集合、使用结果以及如何使用关系数据的主要类和方法。最后，您将看到一组快速引用表，您可以将表用作开发文档存储应用程序的主要参考。

第6章"X插件"：本章讨论X插件及其工作原理。特别是，您将看到如何配置X插件，例如更改端口并通过独立于服务器的SSL启用安全连接。本章还介绍了一些其他系统变量，以及一个冗长的状态变量列表，您可以使用来监视X插件。

第7章，"X协议"：本章从创建X协议的动机、设计的主要原则或目标开始，以及如何以原体为基础实现X协议。您将看到X协议部分如何为简单用例工作。本章还介绍了如何在应用程序中使用原数据f在代码（磁盘上、线上等）中移动数据（消息）的示例，说明了原数据法的强大功能。

第8章"库应用程序：用户界面"：本章介绍名为Flask的Python Web应用程序库。您将了解如何将Flask构建为可扩展框架，该框架可轻松与组件进行增强，以使应用程序更加健壮。本章还介绍了基于我们对Flask了解的库应用程序的用户界面介绍。

第9章"库应用程序：数据库实现"：本章探讨了关系数据库解决方案与JSON字段增强的关系数据库解决方案之间的区别，最后是纯文档存储解决方案。本章演示如何使用完整代码为其中任何一个解决方案构建应用程序，这些代码演示了X DevAPI和MySQL文档存储的许多原则。

第10章"MySQL 8和文档存储的规划"：本章介绍了迁移到MySQL 8的一些策略，包括迁移应用程序以将文档存储与另一个迁移现有数据库应用程序的示例一起使用的注意事项和最佳实践。本章最后提供了使用MySQL 8的一些技巧和技巧。

**如何使用本书**

本书旨在指导您了解有关MySQL 8、JSON、MySQL文档存储的更多信息、发现X DevAPI的强大功能，并了解如何迁移现有文档存储应用程序并构建新的文档存储应用程序。

如果您是MySQL的新增功能，则应花一些时间浏览前四章，包括在您自己的系统上安装MySQL并学习如何使用MySQL外壳。

如果您熟悉Python并通过MySQL外壳使用X DevAPI，则可能需要阅读X DevAPI上的章节，浏览这些示例。另一方面，如果您尚未使用X DevAPI或Python，则应尝试重现章节中的所有代码示例。

从阅读前七章中熟悉MySQL和X DevAPI后，可以完成第8章和第9章，其中演示了如何构建关系数据库解决方案的完整解决方案、与使用JSON列的混合关系表相同的解决方案以及迁移到完整文档存储解决方案的解决方案。

规划人员可能会发现第10章在规划升级到MySQL 8和使MySQL文档存储适应基础结构方面特别有用。

**下载代码**

本书中显示的示例的代码可在apress网站上www.apress.com。您可以在"源代码/下载"选项卡上的书籍信息页面上找到链接。此选项卡位于页面的"相关标题"部分。

**第 1 章**

**介绍MySQL 8新开始**

MySQL不断以新功能改进，这证明了Oracle MySQL工程师（以及Oracle本身）的奉献精神。MySQL工程部门内的驱动力是继续为Internet开发颠覆性数据库技术。甲骨文不仅培养了这种侵略性，而且继续履行其投资和扩大MySQL业务的承诺。最新版本MySQL 8最终证明，甲骨文已经履行了确保MySQL继续保持全球最受欢迎的开源数据库系统的承诺。

早期版本的MySQL添加了一些新的和有趣的功能，因为MySQL 5.01使MySQL成为更好的产品。虽然这些特征已经受到欢迎，并用于解决许多问题，但这些变化主要是进化的改进，而不是革命性的变化。

这种趋势并非MySQL独有，在稳定、成熟的产品中也不罕见。这并不意味着进化发展是坏的，它不是。然而，鉴于出现了几种竞争技术，MySQL工程师意识到，如果他们要继续主导行业，就必须达到更高、更远的水平。

因此，此新版本的MySQL打破了早期版本的许多模具，增加了新的革命性功能，改变了一些人如何使用MySQL的坚实程度。事实上，仅版本号就从5.x跃升至8.02，这标志着技术复杂性的跃升，以及5.x代码库持续开发中断，该代码库持续了13年。

MySQL 8.0的更改包括对现有功能的更改以及一些新的游戏更改功能。本书考察了最重要和最新的功能之一：MySQL文档存储。但是，还有其他同样重要的功能，如组复制和InnoDB群集。虽然我专注于文档存储，但我还将了解如何利用这些其他功能来将MySQL安装到将来。

**MYSQL这个名字是什么意思？**

MySQL这个名字是正确名称和首字母缩略词的组合。SQL是结构化查询语言。我的部分不是占有形式，它是一个名字。在这种情况下，我是创始人的女儿的名字。至于发音，MySQL专家将其发音为"我的S-Q-L"——而不是"我的续集"。

在本章中，我介绍了MySQL 8的一些新功能，包括介绍早期版本中的一些新兴技术、MySQL 8独有的新功能以及使MySQL 8成为迄今为止最大的MySQL版本的革命性功能。

新功能具有很大的复杂性。正如您将看到的，某些功能旨在协同工作，而其他功能则设计为附加组件。以下各节没有解释每个次要细节或列表功能和优点，而是介绍MySQL 8中当前提供的各种功能的基础知识，以便您可以了解可用的功能。您还将看到新版本在传统的MySQL存储和检索机制之外的推进程度。

让我们先看看早期版本中的一些功能，但现在这些功能经过改进，并且更全面地集成到服务器中。

**旧功能再次新增**

第一类功能包括MySQL 5.7中开发的功能，无论是作为单独的实验性开发项目;插件;或作为以后稳定版本的计划功能。因此，这些功能已经以某种有限的形式发布。大多数被视为"开发版本"，并附有一份免责声明，强烈建议它们不在生产环境中使用。某些版本已包含在服务器的最新版本候选（RC）版本中。

更确切地说，Oracle在早期发布时发布了这些功能，以便系统和数据库管理员、信息技术架构师和其他规划人员可以试用这些功能并提供反馈以帮助功能成熟。它还允许客户在开发环境中尽早调整技术，以防功能需要更改基础结构或应用程序。

**什么是插件？**

插件是指向服务器添加功能，而无需编译和重建服务器，以正确合并新功能。插件技术已经存在很长时间了。事实上，MySQL最初支持可插拔的存储引擎，允许您动态添加和删除存储引擎选项。MySQL插件技术从那些时代起就有所发展，但概念是相同的。只要插件与服务器版本兼容，您就可以从Oracle下载MySQL插件并将其安装到服务器上，以便立即使用。

插件也是Oracle将新功能发布到现有稳定版本中的便捷方式。例如，新功能（如组复制）已作为插件引入（但包含在最新版本中）。即使插件作为开发版本发布（考虑早期beta），您仍可以将其与服务器的兼容GA（一般可用）版本一起使用。这使得Oracle能够比将功能与主要服务器版本捆绑在一起更快。对于组复制和类似技术，这通过在接近创纪录的时间内向用户提供这些功能，节省了Oracle多年的开发工作。

MySQL 5.7代码库中有几个功能已经发展。以下是我在本书中探讨的一些关键功能。其中包括JSON数据类型和MySQL外壳。

**JSON数据类型**

自MySQL版本5.7.8起，MySQL支持本机JSON数据类型，该数据类型允许高效访问表行中的JSON文档中的数据。因此，可以在JSON数据类型的表中具有列。JSON代表JavaScript对象表示法.4新的JSON数据类型是使用MySQL作为文档存储的关键组件。简而言之，JSON是用于交换数据的标记语言。它不仅具有人类可读性，还可以直接在应用程序中存储和检索数据，并将其存储到其他应用程序、服务器甚至MySQL中。

事实上，JSON看起来对程序员很熟悉，因为它类似于其他标记方案。JSON也非常简单，因为它仅支持两种类型的结构：1）包含对（名称、值）的集合 2）有序列表（或数组）。当然，您还可以混合和匹配对象中的结构。创建JSON对象时，我们称之为JSON文档。

与MySQL中的正常数据类型不同，JSON数据类型允许您将JSON格式化的对象（文档）存储在一行的列中。单个表中可以有多个JSON列（字段）。尽管可以使用TEXT或BLOB字段（许多人可以这样做）执行此操作，但MySQL中没有内置任何工具与TEXT和BLOB字段中的数据进行交互。因此，数据的操作很大程度上取决于应用程序。此外，数据通常结构，使每行具有相同的"格式"的列。将数据存储在TEXT和BLOB字段中并非新鲜事，并且已执行多年。

这可以通过使用单个字符串甚至数据的二进制表示形式并将其存储在TEXT或BLOB字段中来工作。如果数据足够小，则可以将其存储在VARCHAR和类似的字符串列中。要以这种方式存储和检索数据，您必须对其进行编码，然后解码数据（这可能很乏味，尤其是在尝试从其他人中引入数据时）。

使用JSON数据类型时，您不必编写专用代码来存储和检索数据。这是因为JSON文档被很好地理解，许多编程环境和脚本语言都支持它本机。将JSON视为XML文档应是一个扩展。也就是说，它们提供了一种灵活的方法来存储可能因应用程序而异的数据。JSON允许您存储当时拥有的数据。与典型的数据库表不同，您不必担心默认值（不允许）或是否有足够的列，甚至主/详细信息关系来规范化和存储所有数据在一个漂亮、整洁、结构化的包中。

让我们来看看一个简单的JSON文档，我们可以存储在MySQL中。假设我们有一个联系人列表，其中每个联系人可能都有或可能没有存档地址，也可能没有电子邮件、多个电话号码等。如果要创建一个典型的数据库表来存储此信息，则尽可能为只有名称和单个电话号码的条目存储大量空列。

事实上，我们可以随时添加新的数据项，而无需更改下划线表结构。例如，如果您发现以后需要向某些记录添加Skype ID，则可以在代码中执行此操作，为所需的条目添加密钥，而无需返回并更改任何现有数据。唯一的问题在于，读取数据的代码在访问密钥之前必须更改以测试密钥是否存在。我在第8章和第9章中显示了这方面的示例。

让我们考虑一个示例联系人列表，其中包含几个为我服务的人员，他们居住在我的区域。我需要存储的只是他们的名字和电话号码。有时我只知道（或关心存储）他们的名字。我不需要他们的地址，因为我从来不给他们寄任何东西，他们毕竟只是在街上。

清单1-1演示了一些条目的外观。我选择使用SQL INSERT语句来演示JSON的外观INSERT，以便您可以看到一种非结构化数据可以插入到我们的数据库中。

**Introducing the MySQL 8**

**Document Store**

Charles Bell

**Introduction**

NoSQL has been given a lot of hype in recent years. As with most new technologies, the underlying principles are rarely truly new, rather, it is the unique combination of known

technologies that forms and transforms the whole to become more than the sum of its parts. This is especially true for MySQL 8 and the new MySQL Document Store. Never has MySQL offered so much for so many. Whether you want a traditional relational database solution with a strong foundation or you want the ultimate flexibility to store JSON documents in a document store—or anywhere in between, MySQL can do it.

The trick then is learning how to migrate your applications using each of these technologies: whether you use traditional tables with fixed schemas or you have some

JSON fields to allow some freedom from rigid structure or you employ the flexibility of a

JSON-based document store.

This book will give you the knowledge you seek to navigate the MySQL Document

Store including how to migrate existing applications and best practices for using a

document store solution.

**Intended Audience**

I wrote this book to share my passion for the new MySQL Document Store and Python. I especially wanted to show how anyone can write document store solutions without investing in learning a large, complex language and development environment. The intended audience therefore includes anyone interested in learning about the MySQL Document Store such as database administrators, developers, and information technology managers and strategic planners.

**How This Book Is Structured**

The book was written to guide the reader from a general knowledge of the new features in MySQL 8 to detailed explanations of the components that make up the MySQL Document Store. The first several chapters cover general topics including a short introduction to MySQL 8, how to install MySQL 8, and how to configure the new document store components. Later chapters present more detailed coverage of the MySQL Document Store components including the MySQL Shell, X DevAPI, X Plugin, and more. Following those chapters is a pair of chapters that present an example application for storing information on books written as a relational database, a hybrid, and a document store. Thus, you can see how to migrate a single application through those variants. The book concludes with some notes about upgrading to MySQL 8 and best practices for using the MySQL Document Store including a recap of the migration process. The following is a brief overview of each chapter included in this book.

Chapter 1, “Introducing MySQL 8—A New Beginning”: This chapter explores some of the highlights of the new MySQL server version 8.0. You will discover those features originally introduced in earlier versions that have been adapted to the new paradigm that is version 8.0, features that are new, and those new features that are truly revolutionary such as the document store, Group Replication, and InnoDB Cluster.

Chapter 2, “Getting Started with MySQL”: This chapter presents a tutorial on MySQL discussing the power of using the MySQL database server in its traditional role using the SQL interface; how to issue commands for creating databases and tables for storing data as well as commands for retrieving that data. Although this chapter presents only a small primer on MySQL, you will learn how to get started with your own installation of MySQL.

Chapter 3, “JSON Documents”: This chapter explores the JSON data type in more detail. You will see examples of how to work with the JSON data in relational tables via the numerous built-in JSON functions provided in MySQL. The JSON data type is key to allowing users to develop hybrid solutions that span the gulf of SQL and NoSQL applications.

Chapter 4, “The MySQL Shell”: This chapter demonstrates how to use the MySQL Shell including a look at the startup options, shell commands, connections, sessions, and we even how to do a bit of interactive scripting in JavaScript and Python. This chapter therefore is the key chapter for learning how to get started with the MySQL Shell and working with JSON and relational data. Although this chapter is not an exhaustive coverage of all the features of the MySQL Shell, it provides a broad tutorial for how to use it for the most common tasks.

Chapter 5, “X Developer API”: This chapter explores the X DevAPI and examines the major classes and methods available for connecting to the MySQL server, creating collections, working with results, and even how to work with relational data. Finally, you will see a set of quick references tables that you can use as the primary reference for developing document store applications.

Chapter 6, “X Plugin”: This chapter discusses the X Plugin and how it works. In particular, you will see how to configure the X Plugin such as changing the port and enabling secure connections via SSL that are separate from the server. The chapter also presents some of the other system variables as well as a lengthy list of status variables that

you can use to monitor the X Plugin.

Chapter 7, “X Protocol”: This chapter examines the X Protocol starting with the motivations for why it was created, the chief tenets or goals of the design, and how it was implemented using protobuf as the foundation. You will see a walkthrough of how portions of the X Protocol work for simple use cases. The chapter also presents an example of how to use protobuf in our applications for moving data (messages) around in the code (on disk, over the wire, etc.), which illustrates the power of protobuf.

Chapter 8, “Library Application: User Interface”: This chapter presents a web application library for Python named Flask. You will learn how Flask is built as an extensible framework that is easily augmented with components to make your application more robust. The chapter also presents an introduction to the user interface for the library application built on the foundations of what we learned about Flask.

Chapter 9, “Library Application: Database Implementations”: This chapter explores the differences between a relational database solution and a relational database solution augmented with JSON fields, and finally a pure document store solution. This chapter demonstrates how to build applications for any of these solutions with complete code that demonstrates many of the tenets of the X DevAPI and the MySQL Document Store.

Chapter 10, “Planning for MySQL 8 and the Document Store”: This chapter presents some strategies for migrating to MySQL 8 including considerations and best practices for migrating applications to use the document store with another example of migrating existing database applications. The chapter concludes with some tips and tricks for working with MySQL 8.

**How to Use This Book**

This book is designed to guide you through learning more about MySQL 8, JSON, the MySQL Document Store, discovering the power of X DevAPI, and learning how to migrate existing and building new document store applications.

If you are new to MySQL, you should spend some time going through the first four chapters including installing MySQL on your own system and learning how to use the MySQL Shell.

If you are familiar with Python and have used the X DevAPI via the MySQL Shell, you may want to read the chapters on the X DevAPI skimming through the examples. On the other hand, if you have not used the X DevAPI or Python, you should attempt to reproduce all the code examples in the chapters.

Once you are familiar with MySQL and the X DevAPI from reading the first seven chapters, you can work through Chapters 8 and 9 that present a complete solution that demonstrates how to build a relational database solution, the same solution as a hybrid relational table with JSON columns, and a migration to a full document store solution.

Planners may find Chapter 10 especially helpful in planning to upgrade to MySQL 8 and to adapt the MySQL Document Store to your infrastructure.

**Downloading the Code**

The code for the examples shown in this book is available on the Apress web site, www.apress.com. You can find a link on the book’s information page on the Source Code/Downloads tab. This tab is in the Related Titles section of the page.

**CHAPTER 1**

**Introducing MySQL 8—A New Beginning**

It is a testament to the dedication of the Oracle MySQL engineers (and Oracle itself ) that MySQL continues to improve with new features. The drive within the MySQL engineering division is to continue to develop disruptive database technologies for the Internet. Oracle has not only fostered this aggressiveness but has continued to live up to its promise to invest in and expand their MySQL business. The newest version, MySQL 8, proves conclusively that Oracle has fulfilled the promise to ensure MySQL will remain the world’s most popular open source database system.

Previous versions of MySQL have added some new and interesting features since

MySQL 5.01 making MySQL a better product. Although the features have been well received and used to solve a lot of problems, the changes were largely evolutionary improvements rather than revolutionary changes.

This tendency is not unique to MySQL nor is it unusual in a stable, mature product. That doesn’t mean evolutionary development is bad—it isn’t. However, given that several competitive technologies have emerged, the MySQL engineers realized they must reach higher and further if they are to continue to dominate the industry.

Thus, this new release of MySQL breaks many of the molds of previous versions adding new, revolutionary features that change the firmament of how some will use MySQL. Indeed, the version number alone has jumped from 5.x to 8.02 signifying the jump in technological sophistication and the break from continuous development of the 5.x codebase, which lasted for over 13 years.

The changes to MySQL 8.0 include changes to existing features as well as some new, game changing features. This book examines one of the most important and newest features: the MySQL Document Store. However, there are other equally as important features such as Group Replication and the InnoDB Cluster. Although I focus on the document store, I will also see how these other features can be leveraged to you’re your MySQL installation into the future.

**MYSQL—WHAT DOES IT MEAN?**

The name MySQL is a combination of a proper name and an acronym. SQL is structured query language. The My part isn’t a possessive form—it is a name. In this case, My is the name of the founder’s daughter. As for pronunciation, MySQL experts pronounce it “My-S-Q-L”—not “my sequel.”

In this chapter, I examine some of the new features of MySQL 8 including a short introduction to some of the features that were emerging technologies from previous versions, new features unique to MySQL 8, and those revolutionary features that make MySQL 8 the greatest MySQL release to date.

The new features have a great deal of sophistication. As you will see, some of the features are designed to work together and others are designed as add-ons. Rather than explain every minor detail or list features and benefits, the following sections present the basics of the various features available today in MySQL 8 so that you can get an idea of what is available. You will also see just how far the new version has advanced beyond the traditional MySQL mechanism of storage and retrieval.

Let’s begin with a look at some of the features that were part of earlier releases but are now refined and more fully integrated into the server.

**Old Features New Again**

The first category of features includes those under development in MySQL 5.7 either as a separate, experimental development project; a plugin; or as a planned feature for a later stable release. Thus, these features had already been released in some limited form. Most were considered “development releases” and were accompanied with a disclaimer that strongly suggested they not be used in a production environment. Some had been included in the latest release candidate (RC) versions of the server.

To be more precise, Oracle released these features as early releases so that systems and database administrators, information technology architects, and other planners could try out the features and provide feedback to help the feature mature. It also allows customers to adapt the technologies early in development environments in case the features required changes to the infrastructure or applications.

**WHAT IS A PLUGIN?**

Plugins are means that add functionality to the server without having to compile and rebuild the server proper to incorporate the new features. Plugin technology has been around for a long time. In fact, MySQL originally supported pluggable storage engines that allowed you to add and remove storage engine options on the fly. The MySQL plugin technology has evolved since those days, but the concept is the same. As long as the plugin is compatible with the server version, you can download MySQL plugins from Oracle and install them on your server for immediate use.

Plugins are also a convenient way for Oracle to release new features into existing, stable releases. For example, new features, such as Group Replication, have been introduced as plugins (but are included in the latest release). Even if a plugin is released as a development release (think early beta), you could still use it with the compatible GA (generally available) release of the server. This allows Oracle to produce features much more quickly than having to bundle them with a major server release. In the case of Group Replication and similar technologies, this has saved Oracle years of development work by making the features available to users in near record time.

There are several features that have evolved in the MySQL 5.7 code base. The following are some of the key features that I explore in this book. These include the JSON data type and the MySQL Shell.

**JSON Data Type**

As of MySQL version 5.7.8, MySQL supports a native JSON data type that enables efficient access to data in JSON documents in a table row. Thus, you can have columns in your table of the JSON data type. JSON stands for JavaScript Object Notation.4 The new JSON data type is a key component to using MySQL as a document store. In short, JSON is a markup language used to exchange data. Not only is it human readable, it can be used directly in your applications to store and retrieve data to and from other applications, servers, and even MySQL.

In fact, JSON looks familiar to programmers because it resembles other markup schemes. JSON is also very simple in that it supports only two types of structures: 1) a collection containing pairs (name, value), and 2) an ordered list (or array). Of course, you can also mix and match the structures in an object. When we create a JSON object, we call it a JSON document.

The JSON data type, unlike the normal data types in MySQL, allows you to store JSON formatted objects (documents) in a column for a row. You can have more than one JSON column (field) in a single table. Although you could do this with TEXT or BLOB fields (and many people do), there is no facility built into MySQL to interact with the data in TEXT and BLOB fields. Thus, manipulation of the data is largely application dependent. In addition, the data is normally structured such that every row has the same “format” for the column. Storing data in TEXT and BLOB fields is not new and has been done for years.

This could work by using a single string or even a binary representation of data and storing it in the TEXT or BLOB field. If the data is small enough, you could store it in a VARCHAR and similar string column. To store and retrieve data in this manner, you have to encode then decode the data—something that could be tedious—especially if you're trying to ingest data from someone else.

With the JSON data type, you don't have to write specialized code to store and retrieve data. This is because JSON documents are well understood and many programming environments and scripting languages support it natively. Think of JSON as an outgrowth or extension of what XML documents were supposed to be. That is, they offer a flexible way to store data that may differ from one application to another. JSON allows you to store the data that you have at the time. Unlike a typical database table, you don't have to worry about default values (they’re not allowed) or whether you have enough columns or even master/detail relationships to normalize and store all of the data in a nice, neat, structured package.

Let's take a look at a simple JSON document that we can store in MySQL. Let's say we have a contact list in which each contact may or may not have an address on file, may or may not have an email, multiple phone numbers, and so forth. If you were to create a typical database table to store this information, you may go so far as to store a lot of empty columns for entries that you only have a name and a single phone number.

In fact, we can add new data items any time we want without having to alter the underlining table structure. For example, if you find you later need to add a Skype Id to some of the records, you can do that in your code adding the key for those entries you want without having to go back and change any existing data. The only catch is that your code for reading the data will have to change to test for the existence of the key before accessing it. I show an example of this in Chapters 8 and 9.

Let’s consider an example contact list that contains several people who perform a service for me who live in my area. All I need to store is their name and phone number. Sometimes I only know (or care to store) their first name. I don't need their address because I never send them anything and they're just down the street after all.

Listing 1-1 demonstrates what some of the entries could look like. I chose to demonstrate what JSON looks like by using SQL INSERT statements so that you can see one way unstructured data can be inserted in our database.