大数据应用与技术丛书

Spark 实战

(第 2 版)

[法] 吉恩·乔治·佩林(Jean-Georges Perrin) 著

林 赐 译

清华大学出版社

北 京

北京市版权局著作权合同登记号 图字：01-2020-6237

Jean-Georges Perrin
Spark in Action, Second Edition
EISBN: 978-1-61729-552-2
Original English language edition published by Manning Publications, USA © 2020 by Manning Publications. Simplified Chinese-language edition copyright © 2021 by Tsinghua University Press Limited. All rights reserved.

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。举报：010-62782989，beiqinquan@tup.tsinghua.edu.cn。

图书在版编目(CIP)数据

Spark实战：第2版 / (法)吉恩 • 乔治 • 佩林著；林赐译. —北京：清华大学出版社，2022.1
(大数据应用与技术丛书)
ISBN 978-7-302-59678-3

Ⅰ. ①S… Ⅱ. ①吉… ②林… Ⅲ. ①数据处理软件 Ⅳ. ①TP274

中国版本图书馆 CIP 数据核字(2021)第 263040 号

责任编辑：王　军
装帧设计：孔祥峰
责任校对：成凤进
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：涿州京南印刷厂
经　　销：全国新华书店
开　　本：170mm×240mm　　印　　张：24.5　　字　　数：817 千字
版　　次：2022 年 3 月第 1 版　　印　　次：2022 年 3 月第 1 次印刷
定　　价：99.80 元

产品编号：087712-01

Liz，在此努力的过程中，谢谢你的耐心、支持和爱。

Ruby、Nathaniel、Jack 和 Pierre-Nicolas，我在编写本书时，抽不开身来陪伴你们，谢谢你们对我的理解！

我爱你们！

译 者 序

本书翻译于新冠疫情在全世界肆虐之时，可谓“生于忧患”。短短一年半的时间，全世界数不胜数的人感染了新冠肺炎，甚至死在病毒的攻击之下。

古语有云：“殷忧启圣，多难兴邦。”任何事物的发展都是辩证的。纵观人类与病毒纠葛的历史，每次疫情的暴发，都深刻改变了人类文化、经济和军事的发展进程。例如，14 世纪的黑死病造成了西欧崛起，15 世纪末美洲天花带来了全球降温，18 世纪末黄热病结束了法国在海地的殖民统治，19 世纪非洲牛瘟加速了欧洲殖民扩张，等等。而本次的新冠大流行依然在蔓延之中，疫苗的快速研发似乎让人们看到了隧道尽头的曙光，但是由于变异病毒的出现，人类又陷入了各种不确定之中。

值此鱼游沸鼎之际，身处海外的我，在 YouTube 上观看各路专家、网红对如何防治病毒，以及人类如何与自然共存，议论纷纷，莫衷一是。在各种分析过程中，专家们最常见的做法就是引用各种数据。根据概率论的大数定律，数据量越大，剖分得越细致，所得到的分析结果就越有说服力。

大数据、云计算、人工智能，是当今计算领域发展的三驾马车。云计算为大数据提供了存储和运算之所，人工智能为云计算提供了算法逻辑，而所有这一切都要建立在 Spark 的大数据处理框架的基础之上。在介绍 Spark 之前回顾集群计算的历史，我们不得不谈谈 MapReduce 和 Hadoop，如果没有巨人的肩膀可供站立，Spark 不可能如此成功。

Spark 基于内存计算，整合了内存计算单元，提高了大数据处理的实时性。它兼具高容错性和可伸缩性，因此相对于 Hadoop 的集群处理方法，Spark 在性能方面更具优势。从另一角度看，Spark 可被看作 MapReduce 的一种扩展。在计算的各个阶段，MapReduce 无法进行有效的资源共享，因此不擅长迭代式、交互式和流式的计算工作。针对这一点，Spark 创造性地引入了 RDD(弹性分布式数据集)，实现了计算过程中的资源共享。因为采用了弹性内存分布式数据集，所以 Spark 不仅能提供交互式查询，还可优化迭代工作的负载。

本书循序渐进地向读者介绍 Spark 的历史渊源和运作原理，并利用各种示例生动展示 Spark 的各种应用。本书面向数据工程师和数据分析师。Spark 的技术繁复庞杂，我们很难在一时之间掌握，因此读者要时常温故而知新，在实践中学习，在学习中实践，这样循环反复，才能学有所成。

本书成书于“危难之际”，得到了清华大学出版社的领导和编辑们的信任和鼎力支持。在此，要特别感谢他们的耐心和帮助。在我翻译本书的过程中，就读于西交利物浦大学大数据专业的袁于博同学，也给予了帮助，在此表示感谢。

译者才疏学浅，见闻浅薄，言辞多有不足之处，还望谅解并不吝指正。

林赐

2021 年 10 月 3 日

于加拿大渥太华大学

序　言

分析操作系统

在 20 世纪，商业的规模效应主要由广度和分布所驱动。一家在全球拥有制造业务的公司具有内在的成本和分布优势，因而可生产更具竞争力的产品。一家在全球拥有门店的零售商拥有小公司无法比拟的分布优势。几十年来，这些规模效应推动了竞争优势。

互联网改变了这一切。如今，存在三种主要的规模效应：

- 网络——锁定，由忠诚网络驱动(Facebook、Twitter、Etsy 等)
- 规模经济——降低单位成本，由数量驱动(苹果公司、TSMC 等)
- 数据——卓越的机器学习和洞察力，由动态数据语料库驱动

在 *Big Data Revolution*(Wiley，2015)一书中，我介绍了一些利用数据加强规模效应的公司。但在 2019 年，就全世界范围的机构而言，大数据在很大程度上仍然是未开发的资产。分析操作系统 Spark 是改变这种状况的助推剂。

Spark 一直是 IBM 改革创新、不断发展的催化剂，Spark 分析操作系统统一了数据源和数据访问，其统一的编程模型使其成为开发人员构建数据丰富的分析应用程序的最佳选择。Spark 减少了构建分析工作流的时间并降低了其复杂性，使构建者能专注于机器学习和 Spark 周遭的生态系统。正如我们屡次所见，这个开源项目正以前所未有的速度和规模点燃创新。

本书将带你深入了解 Spark 的世界。它彰显技术的力量，描述 Spark 生态系统的活力，探讨如何将 Spark 投入到当今公司的实际应用程序。无论你是数据工程师、数据科学家、应用程序开发人员，还是 IT 运营者，本书都将为你揭示推动公司或社区创新所需要了解的工具和秘密。

IBM 的战略是以一个成功的开放平台及其周遭环境为基础，添加一些实质性以及差异化的内容。Spark 就是这样一个平台。在 IBM，我们有无数成功的 Spark 示例，当你踏上这段旅程后，你的公司将拥有同样成功的 Spark 示例。

Spark 代表创新——一个分析操作系统将会带来层出不穷的新解决方案，释放大数据规模效应。Spark 也代表一个由精通 Spark 的数据科学家和数据分析师组成的社区，他们可快速地将今天的问题转化为明天的解决方案。Spark 是历史上发展最快的开源项目之一。欢迎加入 Spark！

——Rob Thomas

IBM 云和数据平台的高级副总裁

作 者 简 介

Jean-Georges Perrin 对软件工程和数据的各个方面充满热情。最新项目促使他转向分布式的数据工程，在此项目中，他在混合云环境中广泛使用 Apache Spark、Java 和其他工具。他很自豪地成为法国第一个公认的 IBM Champion，并连续 12 年获奖。作为获奖的数据和软件工程专家，现在，他在全球范围内都开展了业务，但重心在他所居住的美国。Jean-Georges 是资深的会议演讲者和参与者，他以书面或在线媒体的形式发表文章，分享他在 IT 行业超过 25 年的经验。

致　谢

在本部分，我要对在成书过程中对我施以援手的人表示感谢。我是一个容易忘记别人恩典的人，因此，如果你感觉受到了冷落，在此我真心表示抱歉。完成本书需要我们付出艰苦的努力，如果是一个人单独完成此书，估计它在亚马逊上只能获得两星或三星，而不会是五星的评分(你很快就可给出这个评分。这是行动号召，在此表示真诚的感谢)。

首先，我衷心感谢在这个项目(编写本书)上信任我的工作团队：Zaloni(Anupam Rakshit 和 Tufail Khan)、Lumeris(Jon Farn、Surya Koduru、Noel Foster、Divya Penmetsa、Srini Gaddam 和 Bryce Tutt，所有这些人都在 Spark 潮流中忠实地追随我)、Veracity Solutions 的员工以及我在 Advance Auto Parts 的新团队。

感谢得克萨斯大学奥斯汀分校统计系的 Mary Parker。她帮助澄清了一些问题。

我要感谢整个社区，包括 Jim Hughes、Michael Ben-David、Marcel-Jan Krijgsman、Jean-Francois Morin 以及所有在 GitHub 上匿名发布拉取(pull)请求的人。在此，我要向 Databricks、IBM、Netflix、优步、英特尔、苹果、Alluxio、Oracle、微软、Cloudera、英伟达、Facebook、Google、阿里巴巴、众多大学以及许许多多为 Spark 做出贡献的人深表谢意。具体而言，我要感谢 Holden Karau、Jacek Laskowski、Sean Owen、Matei Zaharia 和 Jules Damji，他们为本书的顺利出版做出了巨大的贡献，并给予我莫大的鼓舞和支持。

在这个项目中，我参加了几个播客。感谢 Tobias Macey 的数据工程播客、IBM 的 Al Martin“让数据变得简单”，以及 Jhon Masschelein 和 Dave Russell 的 Roaring Elephant。

作为 IBM Champion，在这个项目期间，与众多 IBM 员工一起工作，我感到非常愉快。他们要么直接或间接提供帮助，要么鼓励我，他们就是：Rob Thomas、Marius Ciortea、Albert Martin、Steve Moore、Sourav Mazumder、Stacey Ronaghan、Mei-Mei Fu、Vijay Bommireddipalli、Sunitha Kambhampati、Sahdev Zala 和我的兄弟 Stuart Litel。

我要感谢 Manning 出版社采纳了这个疯狂的项目。和所有优秀的电影一样，下面按出场顺序列出致谢名单：组稿编辑 Michael Stephens，出版商 Marjan Bace，文稿编辑 Marina Michaels 和 Toni Arritola，制作人员 Erin Twohey、Rebecca Rinehart、Bert Bates、Candace Gillhoolley、Radmila Ercegovac、Aleks Dragosavljevic、Matko Hrvatin、Christopher Kaufmann、Ana Romac、Cheryl Weisman、Lori Weidert、Sharon Wilkey 和 Melody Dolab。

我还要感谢 Manning 出版社的所有审稿人：Anupam Sengupta、Arun Lakkakulam、Christian Kreutzer-Beck、Christopher Kardell、Conor Redmond、Ezra Schroeder、Gábor László Hajba、Gary A. Stafford、George Thomas、Giuliano Araujo Bertoti、Igor Franca、Igor Karp、Jeroen Benckhuijsen、Juan Rufes、Kelvin Johnson、Kelvin Rawls、Mario-Leander Reimer、Markus Breuer、Massimo Dalla Rovere、Pavan Madhira、Sambaran Hazra、Shobha Iyer、Ubaldo Pescatore、Victor Durán 和 William E. Wheeler。

写一本好书确实需要许多幕后人员。我还要感谢 Petar Zečević 和 Marco Banaći，他们编写了本书的第 1 版。感谢 Thomas Lockney 详细的技术审查，也感谢 Rambabu Posa 移植了本书中的代码，还要感谢 Jon Rioux 开始撰写 *PySpark in Action* 一书。他提出了“Manning 的 Spark 团队”这个想法。

我要再次感谢 Marina。在我撰写本书的大部分时间里，Marina 是我的文稿编辑。当我遇到问题时，她会提出建议。虽然她对我非常严厉(是的，我不能稍有懈怠)，但是在这个项目(即编写本书)中，她发挥了重要作用。我会记得我们围绕这本书进行的长时间讨论。我会想念你的，姐姐。

最后，我要感谢我的父母，他们对我的支持超出了应有的水平；我要感谢我的妻子，她给予我很大的帮助，包括对编辑的理解；我还要感谢我的孩子们——Pierre-Nicolas、Jack、Nathaniel 和 Ruby，为了完成本书，我无暇陪伴他们，感谢他们对我的谅解。

关于封面插图

《Spark 实战(第 2 版)》封面上的插图标题为“Homme et Femme de Housberg, près Strasbourg”(斯特拉斯堡附近 Housberg 的男人和女人)。如今，Housberg 已更名为 Hausbergen，它是阿尔萨斯内的自然区域和历史领土，被分为三个村庄：Niederhausbergen(下 Hausbergen)、Mittelhausbergen(中 Hausbergen)和 Oberhausbergen(上 Hausbergen)。该插图来自 Jacques Grasset de Saint-Sauveur (1757—1810 年)于 1797 年在法国出版的名为《各国风俗习惯》(*Costumes de Différents Pays*)一书中的不同国家的服饰。每幅插图都是手工精心绘制和着色的。

这幅独特的插图对我来说具有特殊的意义。它可以用于此书，我真的很高兴。我出生在阿尔萨斯的斯特拉斯堡，目前居住在法国。我非常珍惜阿尔萨斯的传统。当我决定移民到美国时，我知道我即将离开这种文化和家人，尤其是我的父母和姐妹。我的父母住在一个叫 Souffelweyersheim 的小镇，与 Niederhausbergen 直接相邻。每次看到封面的这个插图，我都会想起他们(虽然我爸爸的头发掉了很多)。

Grasset de Saint-Sauveur 丰富多样的收藏生动地提醒我们，早在 200 年前，世界各地的城镇和地区的文化差异就很大。人们彼此隔离，讲不同的方言(这里是阿尔萨斯语)。在街上或乡下，通过着装即可轻松确定某人住在哪里，从事什么职业，以及生活状况如何。

从那时起，人们的着装方式发生了天翻地覆的变化，曾经如此丰富的地区多样性已走向消亡。现在，我们已经很难从服饰角度区分不同大陆的居民了，更不用说区分来自不同国家、地区和城镇的人们了。也许我们已经用文化多样性换取了更加多样化的个人生活，即换取了更加多样化和快节奏的科技生活。

如今，不同的计算机书籍令人难以区分，Manning 基于两个世纪前丰富多样的地区生活，使用 Grasset de Saint-Sauveur 的作品制作书籍封面，使书籍绽放异彩，展现了计算机工作的创造性和能动性。

关 于 本 书

当我开始撰写你现在正在阅读的《Spark 实战(第 2 版)》一书时，我的目标是：

- 帮助 Java 社区开发人员使用 Apache Spark，这说明你不必学习 Scala 或 Python。
- 解释 Apache Spark、大数据工程和数据科学背后的关键概念，除了需要具备关系数据库和一些 SQL 知识，你不必了解其他任何知识。
- 宣传 Spark 是专为分布式计算和分析而设计的操作系统。

我认为利用大量的示例是讲授计算机科学的不二法门。本书中的示例是学习过程的重要组成部分。为此，我尽可能设计接近现实生活情形的示例。数据库来自现实生活，附带了一些缺陷，它们并不是理想教科书中的数据集，不会“始终有效”。因此，当你将这些示例和数据集结合在一起时，你会以比较务实的方式(而非死板的方式)工作和学习。我将这些示例称为实验(lab)，希望它们对你具有启发性，也希望你尝试使用它们进行实验。

常言道：“一图胜千言。”因此本书基于该谚语使用了很多插图，总的算下来，我让你少读了 183 000 个字。

本书读者对象

将一本书与相关的工作岗位联系起来，是一件比较困难的事情。如果你是数据工程师、数据科学家、软件工程师或数据/软件架构师，那么你阅读本书时肯定十分开心；如果你是企业架构师，知识面广，那么你很可能已知道了所有这些知识，不是吗？比较诚恳地说，如果你希望获得有关以下任何主题的更多知识，那么本书将大有裨益：

- 使用 Apache Spark 构建分析和数据管道：数据提取、数据转换和数据导出/数据发布。
- 不必学习 Scala 或 Hadoop 即可使用 Spark：使用 Java 学习 Spark。
- 了解关系数据库和 Spark 之间的区别。
- 关于大数据的基本概念，包括在 Spark 环境中可能遇到的关键 Hadoop 组件。
- 确定 Spark 在企业架构中的位置。
- 在大数据环境中使用现有的 Java 和 RDBMS 技能。
- 了解数据帧 API。
- 通过在 Spark 中提取数据来集成关系数据库。
- 通过流收集数据。
- 了解行业的演变以及适合使用 Spark 的原因。
- 理解数据帧的核心作用，并使用数据帧。

- 了解弹性分布式数据集(Resilient Distributed Dataset，RDD)的概念，以及我们不再使用它们的原因。
- 了解如何与 Spark 交互。
- 了解 Spark 的各个组件：驱动器、执行器、主服务器和工作器、Catalyst、Tungsten。
- 了解 Hadoop 派生的关键技术(如 YARN 或 HDFS)的作用。
- 了解资源管理器(如 YARN、Mesos 和内置管理器)的作用。
- 以批处理模式，经由流从各种文件中提取数据。
- 在 Spark 中使用 SQL。
- 操作 Spark 提供的静态函数。
- 了解不变性的概念及其重要性。
- 使用 Java 用户定义函数(User-defined Function，UDF)扩展 Spark。
- 使用新数据源扩展 Spark。
- 将 JSON 中的数据线性化，以便使用 SQL。
- 对数据帧执行聚合和合并操作。
- 使用用户定义的聚合函数(User-defined Aggregate Function，UDAF)扩展聚合操作。
- 了解缓存和检查点之间的区别，提高 Spark 应用程序的性能。
- 将数据导出到文件和数据库。
- 了解 AWS、Azure、IBM Cloud、GCP 和本地集群上的部署。
- 从 CSV、XML、JSON、文本、Parquet、ORC 和 Avro 文件中提取数据。
- 扩展数据源，以如何使用 EXIF 提取照片元数据为例，重点介绍数据源 API v1。
- 在构建管道时，将 Delta Lake 与 Spark 结合起来使用。

本书涵盖的内容

本书旨在教你如何在应用程序中使用 Spark，或基于 Spark 构建特定的应用程序。

本书为数据工程师和 Java 软件工程师而编写。当我开始学习 Spark 时，所有的知识都是基于 Scala 的，几乎所有的文档都出自官方网站，Stack Overflow 上每隔很长的一段时间显示一个 Spark 问题。当然，文档声称 Spark 有 Java API，但是鲜有高级示例。

当时，我的队友很困惑，摇摆于学习 Spark 和学习 Scala 之间，而我们的管理层想要成果。团队成员成为我撰写本书的动力。

假设你具有基本的 Java 和 RDBMS 知识。虽然 Java 11 已经发布了，但是本书所有的示例都使用 Java 8。

你不必具备 Hadoop 知识即可阅读本书，但由于你将需要一些 Hadoop 组件(非常少)，本书将介绍这些组件。如果你对 Hadoop 已有所了解，那么你肯定会发现本书令人耳目一新。这是一本关于 Spark 和 Java 的书，因此你不必具备任何 Scala 知识。

当我还是个小孩时(我必须承认，现在依然如此)，我喜欢阅读法语漫画(介于漫画书和图画小说之间)，结果，我喜欢上了插图。在本书中，我使用了许多插画。如图 1 所示，这是一张包含多个组件、图标和图例的典型图表。

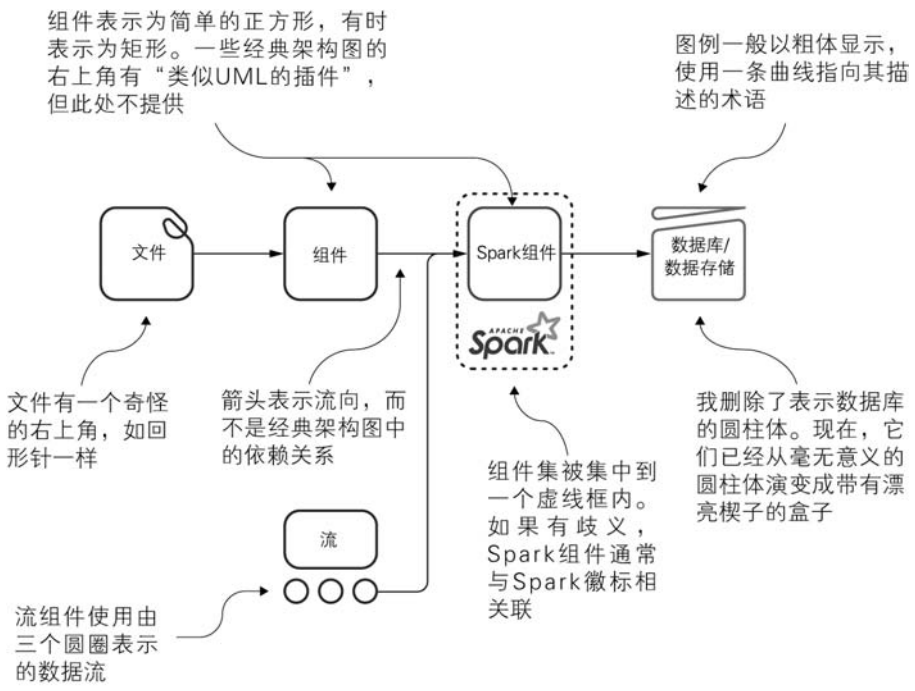

图 1　本书典型插图所用的图像

本书的组织方式

本书共分为 4 个部分和 18 个附录。

第Ⅰ部分提供有关 Spark 的关键内容。该部分将探讨 Spark 的理论和一般概念，内容比较生硬，但你不要绝望；本书将展示很多示例和图表，读起来就像一本漫画书。

- 第 1 章使用简单的示例进行总体介绍。本章将讲解为什么 Spark 是分布式分析操作系统。
- 第 2 章将引导你完成一个简单的 Spark 处理流程。
- 第 3 章将探讨数据帧的强大之处，它结合了 API 和 Spark 的存储功能。
- 第 4 章将解释惰性机制，比较 Spark 和 RDBMS，介绍有向无环图(Directed Acyclic Graph，DAG)。
- 第 5 章和第 6 章相互关联：我们将构建一个小型应用程序，构建集群，部署应用程序。第 5 章主要描述如何构建小型应用程序，而第 6 章介绍如何部署应用程序。

第Ⅱ部分将开始深入研究有关数据提取的实用示例。数据提取是将数据带入 Spark 的过程。虽然该过程并不复杂，但是有多种可能性和组合。

- 第 7 章说明如何从文件中提取数据，文件格式包括：CSV、文本、JSON、XML、Avro、ORC 和 Parquet。每种文件格式都有一个示例。
- 第 8 章介绍如何从数据库进行数据提取：数据来自关系数据库和其他数据存储产品。
- 第 9 章描述如何从自定义数据源提取数据。
- 第 10 章重点介绍流数据。

第III部分描述数据转换：这就是我所说的繁重的数据提升工作。此部分将讲解有关数据质量、数据转换和数据发布的知识。本书的大部分篇幅讨论如何使用 SQL 及其 API 操作数据帧，以及如何使用 UDF 聚合、缓存和扩展 Spark。

- 第 11 章描述众所周知的查询语言 SQL。
- 第 12 章介绍如何执行数据转换。
- 第 13 章将数据转换扩展到整个文档层面。该章还将解释静态函数，这是 Spark 的一个重要方面。
- 第 14 章讨论如何使用用户定义函数扩展 Spark。
- 数据聚合也是众所周知的数据库概念，也许还是数据分析的关键。第 15 章介绍数据聚合：Spark 所包括的数据聚合和自定义的数据聚合。

最后，第Ⅳ部分讨论如何迈向生产环境，专注于更高级的主题。此部分介绍分区和数据导出、部署约束(包括云环境)和优化。

- 第 16 章重点介绍优化技术：缓存和检查点。
- 第 17 章讨论如何将数据导出到数据库和文件。该章还将说明如何使用 Delta Lake(位于 Spark 内核旁的数据库)。
- 第 18 章详细介绍部署所需的参考架构及其安全性。该章少了动手操作的部分，但却充满了关键信息。

附录虽然不是必需的，但也提供丰富的信息：安装方法、故障排除和情境化。其中很多是有关 Java 上下文中 Apache Spark 的精选参考知识。附录部分的内容请扫描封底二维码下载。

关于代码

如前所述，每一章(除了第 6 章和第 18 章)都有结合了代码和数据的实验。源代码为编号的代码清单，与普通文本对齐。在两种情况下，源代码使用等宽字体进行格式化，与一般文本区别开来。有时候，代码也以粗体(**bold**)突出显示，以彰显其在代码块中的重要地位。

根据 Apache 2.0 许可条例，所有代码都可在 GitHub 上免费获得。数据可能具有不同的许可权限。每章都有自己的存储库。有两个例外：

- 第 6 章使用了第 5 章的代码。
- 第 18 章详细讨论部署，没有代码。

源代码控制工具允许分支，因此主分支包含针对最新生产版本的代码。每个存储库都包含针对特定版本的分支(如果适用)。

实验(lab)使用三位数字的编号，从 100 开始。有两种实验(lab)：书中描述的实验和在线提供的额外实验：

- 书中描述的实验按章节的小节进行编号。因此，第 12 章的实验#200 在第 12 章的第 2 节中介绍。与此类似，第 17 章的实验#100 在第 17 章的第 1 节中详细介绍。
- 书中未描述的实验以 9 开头，如 900、910 等。

900 系列的实验数量仍在增长：我在不断增加此类实验。实验(lab)编号不连续，与 BASIC 代码中的行号类似。

在 GitHub 中，我们可找到 Python、Scala 和 Java 代码(除非它不适用)。但为了保持清晰性，本

书仅使用 Java。

许多情况下，本书对原始源代码重新进行格式化；我们添加换行符，重新设计缩进，以适应书中可用的页面空间。在极少数情况下，这样做还是不够，因此代码清单还包括行继续标记(➥)。此外，如果文本中已描述代码，那么代码清单中经常会移除源代码中的注释。许多代码清单中附有代码注解，以突出重要的概念。

前　言

Apache Spark 早已闻名遐迩，不必赘述。如果你正在阅读本部分，那么你可能多多少少对本书的内容有所了解：大规模数据工程和数据科学、分布式处理等。但是从 Rob Thomas 的序言和第 1 章开始，很快你就会发现，Spark 的内容远不止于此。

就像 Obelix 沉迷于魔药[1]一样，2015 年，我开始痴迷于 Spark。当时，我在一家法国计算机硬件公司工作，协助设计高性能的数据分析系统。与众人一样，一开始我对 Spark 持怀疑态度。之后，我开始使用它，到如今，你就看到了本书的问世。从最初的怀疑，到最后我对如此神奇的工具产生了真正的热情，这个工具使我们能以一种非常简单的方式处理数据——这就是我真诚的信念。

我用 Spark 启动了几个项目，这让我能够在 Spark Summit、IBM Think 以及 All Things Open、Open Source 101 上发表演讲。通过本地的 Spark 用户组，我在北卡罗来纳州的 Raleigh-Durham 地区与他人合作进行了动画制作。这让我结识了一些优秀的人，还看到了大量与 Spark 相关的项目。结果，我的热情继续燃烧。

本书分享了我的这种热情。

虽然本书中的示例(或实验)基于 Java，但唯一的存储库也包含 Scala 和 Python。随着 Spark 3.0 的推出，Manning 团队和我决定确保本书讲解的是最新版本，而不是过期的想法。

也许你已经猜到了，我喜欢漫画书，且伴随着漫画书长大。我喜欢这种交流方式，你将在本书中看到这种交流方式。虽然这不是一本漫画书，但是它有近 200 张图片，应该可帮助你了解 Apache Spark 这个奇妙的工具。

Asterix 有 Obelix 作为朋友，同样，《Spark 实战(第 2 版)》有参考资料作为补充。你可从 Manning 网站的资源部分免费下载参考资料。此补充材料包含 Spark 静态函数的参考信息，我希望最终它将成为更有用的参考资源。

如果你喜欢本书，请在亚马逊上撰写评论。如果你不喜欢本书，那么请如人们在婚礼上所说的那样，永远保持沉默。尽管如此，我仍然真诚地希望你喜欢本书。

大局已定，木已成舟(Alea iacta est)。[2]

1　Obelix 是一个漫画卡通人物，他是 Asterix 不离不弃的朋友。高卢人 Asterix 喝下魔法药水，获得超能力后，可正常击败罗马人(和海盗)。小时候，Obelix 掉进了制作魔法药水的坩埚中，因此魔法药水对他产生了永久的影响。Asterix 是欧洲流行的漫画。

2　本句源于凯撒大帝(Asterix 的死敌)。当时凯撒率领他的军队到卢比孔河：事情已经发生，无法改变。就像本书已经为读者印刷，不可改变一样。

目　录

第 I 部分　通过示例讲解理论

第 1 章　Spark 介绍……3
1.1　Spark 简介及其作用……4
1.1.1　什么是 Spark……4
1.1.2　Spark 神力的四个支柱……5
1.2　如何使用 Spark……7
1.2.1　数据处理/工程场景中的 Spark……7
1.2.2　数据科学场景中的 Spark……8
1.3　使用 Spark，能做些什么……9
1.3.1　使用 Spark 预测 NC 餐饮行业的餐馆质量……10
1.3.2　Spark 允许 Lumeris 进行快速数据传输……10
1.3.3　Spark 分析 CERN 的设备日志……10
1.3.4　其他用例……11
1.4　为什么你应该喜欢数据帧……11
1.4.1　从 Java 角度了解数据帧……11
1.4.2　从 RDBMS 角度理解数据帧……12
1.4.3　数据帧的图形表示……12
1.5　第一个示例……13
1.5.1　推荐软件……13
1.5.2　下载代码……13
1.5.3　运行第一个应用程序……14
1.5.4　第一份代码……15
1.6　小结……16

第 2 章　架构和流程……17
2.1　构建思维模型……17
2.2　使用 Java 代码构建思维模型……18
2.3　运行应用程序……21
2.3.1　连接到主机……21
2.3.2　加载或提取 CSV 文件……22
2.3.3　转换数据……25
2.3.4　将数据帧中完成的工作保存到数据库中……26
2.4　小结……29

第 3 章　数据帧的重要作用……31
3.1　数据帧在 Spark 中的基本作用……32
3.1.1　数据帧的组织……32
3.1.2　不变性并非贬低之词……33
3.2　通过示例演示数据帧的使用……35
3.2.1　简单提取 CSV 后的数据帧……36
3.2.2　数据存储在分区中……40
3.2.3　挖掘模式……41
3.2.4　提取 JSON 后的数据帧……43
3.2.5　合并两个数据帧……48
3.3　数据帧 Dataset<Row>……53
3.3.1　重用 POJO……53
3.3.2　创建字符串数据集……54
3.3.3　来回转换……55
3.4　数据帧的祖先：RDD……60
3.5　小结……61

第 4 章　Spark 的“惰性”本质……63
4.1　现实中懒惰但高效的示例……64
4.2　懒惰但高效的 Spark 示例……65
4.2.1　查看数据转换和数据操作的结果……65
4.2.2　数据转换的过程，逐步进行……66
4.2.3　数据转换/操作流程的后台代码……68

4.2.4 在 182 毫秒内创建 700 多万个数据点的奥秘 …… 71
4.2.5 操作计时背后的奥秘 …… 72
4.3 与 RDBMS 和传统应用程序进行比较 …… 76
4.3.1 使用青少年生育率数据集 …… 76
4.3.2 分析传统应用程序和 Spark 应用程序之间的区别 …… 77
4.4 对于以数据为中心的应用程序而言，Spark 的表现出乎意料 …… 78
4.5 Catalyst 是应用程序的催化器 …… 79
4.6 小结 …… 81

第 5 章 构建一个用于部署的简单应用程序 …… 83
5.1 无数据提取的示例 …… 83
5.1.1 计算 π …… 84
5.1.2 计算近似值 π 的代码 …… 85
5.1.3 Java 中的 lambda 函数是什么 …… 90
5.1.4 使用 lambda 函数估算 π …… 92
5.2 与 Spark 交互 …… 93
5.2.1 本地模式 …… 94
5.2.2 集群模式 …… 95
5.2.3 Scala 和 Python 的交互模式 …… 97
5.3 小结 …… 102

第 6 章 部署简单的应用程序 …… 105
6.1 示例之外：组件的作用 …… 106
6.1.1 快速浏览组件及其之间的交互 …… 107
6.1.2 Spark 架构的故障排除技巧 …… 110
6.1.3 知识拓展 …… 110
6.2 构建集群 …… 111
6.2.1 如何构建集群 …… 111
6.2.2 设置环境 …… 112
6.3 构建应用程序，在集群上运行 …… 115
6.3.1 构建应用程序的超级 JAR …… 115
6.3.2 使用 Git 和 Maven 构建应用程序 …… 117
6.4 在集群上运行应用程序 …… 119
6.4.1 提交超级 JAR …… 119
6.4.2 运行应用程序 …… 120
6.4.3 分析 Spark 的用户界面 …… 121
6.5 小结 …… 122

第 II 部分 数据提取

第 7 章 从文件中提取数据 …… 125
7.1 解析器的常见行为 …… 126
7.2 从 CSV 中提取数据(比较复杂) …… 126
7.2.1 预期输出 …… 128
7.2.2 代码 …… 128
7.3 使用已知模式提取 CSV …… 129
7.3.1 预期输出 …… 130
7.3.2 代码 …… 130
7.4 提取 JSON 文件 …… 132
7.4.1 预期输出 …… 134
7.4.2 代码 …… 134
7.5 提取多行 JSON 文件 …… 135
7.5.1 预期输出 …… 137
7.5.2 代码 …… 137
7.6 提取 XML 文件 …… 138
7.6.1 预期输出 …… 140
7.6.2 代码 …… 140
7.7 提取文本文件 …… 142
7.7.1 预期输出 …… 143
7.7.2 代码 …… 143
7.8 用于大数据的文件格式 …… 144
7.8.1 传统文件格式的问题 …… 144
7.8.2 Avro 是基于模式的序列化格式 …… 145
7.8.3 ORC 是一种列式存储格式 …… 145
7.8.4 Parquet 也是一种列式存储格式 …… 146
7.8.5 比较 Avro、ORC 和 Parquet …… 146
7.9 提取 Avro、ORC 和 Parquet 文件 …… 146
7.9.1 提取 Avro …… 146

7.9.2 提取 ORC ····· 148
7.9.3 提取 Parquet ····· 150
7.9.4 用于提取 Avro、ORC 或 Parquet 的参考表格 ····· 151
7.10 小结 ····· 151

第 8 章 从数据库中提取数据 ····· 153
8.1 从关系数据库中提取数据 ····· 154
8.1.1 数据库连接备忘录 ····· 154
8.1.2 了解示例中使用的数据 ····· 155
8.1.3 预期输出 ····· 156
8.1.4 代码 ····· 157
8.1.5 可替代的代码 ····· 159
8.2 dialect 的作用 ····· 160
8.2.1 什么是 dialect ····· 160
8.2.2 Spark 提供的 JDBC dialect ····· 161
8.2.3 构建自定义 dialect ····· 161
8.3 高级查询和提取 ····· 163
8.3.1 使用 WHERE 子句进行过滤 ····· 163
8.3.2 在数据库中连接数据 ····· 166
8.3.3 执行数据提取和分区 ····· 168
8.3.4 高级功能总结 ····· 171
8.4 从 Elasticsearch 中提取数据 ····· 171
8.4.1 数据流 ····· 171
8.4.2 Spark 提取的 NYC 餐馆数据集 ····· 172
8.4.3 从 Elasticsearch 中提取 NYC 餐馆数据集的代码 ····· 173
8.5 小结 ····· 175

第 9 章 数据提取进阶：寻找数据源与构建自定义数据源 ····· 177
9.1 什么是数据源 ····· 179
9.2 直接连接数据源的好处 ····· 179
9.2.1 临时文件 ····· 180
9.2.2 数据质量脚本 ····· 181
9.2.3 按需提供数据 ····· 181
9.3 查找 Spark 软件包中的数据源 ····· 181
9.4 构建自己的数据源 ····· 181
9.4.1 示例项目的范围 ····· 182
9.4.2 数据源 API 和选项 ····· 183
9.5 幕后工作：构建数据源本身 ····· 185
9.6 使用注册器文件和广告器类 ····· 186
9.7 理解数据和模式之间的关系 ····· 188
9.7.1 数据源构建关系 ····· 189
9.7.2 关系内部 ····· 191
9.8 使用 JavaBean 构建模式 ····· 194
9.9 使用实用程序构建数据帧的神奇方法 ····· 196
9.10 其他类 ····· 201
9.11 小结 ····· 201

第 10 章 提取结构化流数据 ····· 203
10.1 什么是流数据 ····· 204
10.2 创建首个流数据 ····· 205
10.2.1 生成文件流数据 ····· 206
10.2.2 消费记录 ····· 208
10.2.3 获取记录，而非数据行 ····· 213
10.3 从网络流数据中提取数据 ····· 214
10.4 处理多个流数据 ····· 216
10.5 区分离散化流数据和结构化流数据 ····· 221
10.6 小结 ····· 221

第 III 部分 转换数据

第 11 章 使用 SQL ····· 225
11.1 使用 Spark SQL ····· 225
11.2 本地视图与全局视图之间的区别 ····· 229
11.3 混合使用数据帧 API 和 Spark SQL ····· 230
11.4 不要删除数据 ····· 233
11.5 进一步了解 SQL ····· 235
11.6 小结 ····· 235

第 12 章 转换数据 ····· 237
12.1 数据转换是什么 ····· 238

12.2 在记录层面进行数据转换的过程和示例……238
12.2.1 数据发现，了解数据的复杂性……240
12.2.2 数据映射，绘制过程……241
12.2.3 编写转换代码……244
12.2.4 审查数据转换，确保质量流程……249
12.2.5 如何排序……251
12.2.6 结束 Spark 数据转换的首次演示……251
12.3 连接数据集……251
12.3.1 仔细查看要连接的数据集……252
12.3.2 构建各县的高等教育机构列表……253
12.3.3 执行连接操作……258
12.4 执行更多的数据转换……263
12.5 小结……263

第 13 章 转换整个文档……265
13.1 转换整个文档及其结构……265
13.1.1 展平 JSON 文档……266
13.1.2 构建嵌套文档，用于数据传输和存储……270
13.2 静态函数背后的魔力……274
13.3 执行更多的数据转换……275
13.4 小结……275

第 14 章 使用自定义函数扩展数据转换……277
14.1 扩展 Apache Spark……278
14.2 注册和调用 UDF……279
14.2.1 在 Spark 中注册 UDF……281
14.2.2 将 UDF 与数据帧 API 结合起来使用……282
14.2.3 使用 SQL 处理 UDF……283
14.2.4 实现 UDF……284
14.2.5 编写服务代码……285
14.3 使用 UDF，确保数据高质量……287
14.4 考虑 UDF 的约束……289
14.5 小结……289

第 15 章 聚合数据……291
15.1 使用 Spark 聚合数据……291
15.1.1 简单回顾数据聚合……292
15.1.2 使用 Spark 执行基本的数据聚合……294
15.2 使用实时数据执行数据聚合……296
15.2.1 准备数据集……297
15.2.2 聚合数据，更好地了解学校……301
15.3 使用 UDAF 构建自定义的聚合操作……306
15.4 小结……311

第 IV 部分 百尺竿头，更进一步

第 16 章 缓存和检查点：增强 Spark 的性能……315
16.1 使用缓存和检查点可提高性能……315
16.1.1 Spark 缓存的用途……317
16.1.2 Spark 检查点的妙用……318
16.1.3 使用缓存和检查点……318
16.2 缓存实战……326
16.3 有关性能优化的知识拓展……335
16.4 小结……335

第 17 章 导出数据，构建完整数据管道……337
17.1 导出数据的主要概念……337
17.1.1 使用 NASA 数据集构建管道……338
17.1.2 将列转换为日期时间(datetime)……341
17.1.3 将置信度百分比转换为置信度等级……341
17.1.4 导出数据……342

17.1.5 导出数据：实际发生了什么……344
17.2 Delta Lake：使用系统核心的数据库……346
17.2.1 理解需要数据库的原因……346
17.2.2 在数据管道中使用 Delta Lake……347
17.2.3 消费来自 Delta Lake 的数据……351
17.3 从 Spark 访问云存储服务……353
17.4 小结……354
第 18 章 探索部署约束：了解生态系统……355
18.1 使用 YARN、Mesos 和 Kubernetes 管理资源……356
18.1.1 使用内置的独立模式管理资源……356
18.1.2 在 Hadoop 环境中，使用 YARN 管理资源……357
18.1.3 Mesos 是独立的资源管理器……358
18.1.4 Kubernetes 编排容器……360
18.1.5 选择合适的资源管理器……360
18.2 与 Spark 共享文件……361
18.2.1 访问文件中包含的数据……362
18.2.2 通过分布式文件系统共享文件……362
18.2.3 访问共享驱动器或文件服务器上的文件……363
18.2.4 使用文件共享服务分发文件……364
18.2.5 访问 Spark 文件的其他选项……365
18.2.6 用于与 Spark 共享文件的混合解决方案……365
18.3 确保 Spark 应用程序的安全……365
18.3.1 保护基础架构的网络组件……366
18.3.2 保护 Spark 磁盘的使用……367
18.4 小结……367

附录部分内容通过封底二维码下载获取

附录 A 安装 Eclipse……369
附录 B 安装 Maven……375
附录 C 安装 Git……379
附录 D 下载代码，启用 Eclipse……381
附录 E 企业数据的历史……387
附录 F 有关关系数据库的帮助信息……391
附录 G 静态函数使数据转换变得容易……397
附录 H 简略的 Maven 备忘单……407
附录 I 数据转换和数据操作的参考资料……411
附录 J Scala 简介……421
附录 K 在生产环境中安装 Spark，以及一些提示……423
附录 L 关于数据提取的参考资料……435
附录 M 关于连接的参考资料……447
附录 N 安装 Elasticsearch 和示例数据……459
附录 O 生成流数据……465
附录 P 有关流数据的参考资料……469
附录 Q 有关导出数据的参考资料……479
附录 R 遇到困难，寻找帮助……487

第 I 部分

通过示例讲解理论

与使用任何技术一样，在深入使用 Spark 之前，你需要先了解一些“无聊”的理论。本部分内容共分为 6 章，每一章合理地阐述了一些概念，并通过示例进行了说明。

第 1 章使用简单示例总体介绍 Spark。你将了解为什么 Spark 不只是一组简单的工具，还是一个真正的分布式分析操作系统。学完第 1 章之后，你将能使用 Spark 进行简单的数据提取。

第 2 章从较高的层次展示 Spark 的工作原理。该章教你通过逐步构建思维模型(代表你的思维过程)来构建 Spark 组件。该章的实验还将教你如何从数据库中导出数据。该章包含许多插图，相比于仅使用字词和代码的方法，这种编写方式将简化你的学习过程！

第 3 章带你进入一个全新的维度：揭示功能强大的数据帧，这个数据帧将 Spark 的 API 和存储功能结合在一起。在该章的实验中，你将加载两个数据集并将它们合并。

第 4 章为“惰性”翻案，解释为什么 Spark 要使用惰性优化(Lazy Optimization)。你将学习有向无环图(Directed Acyclic Graph，DAG)，并比较 Spark 和 RDBMS。该章实验将教你如何使用数据帧 API 操作数据。

第 5 章和第 6 章的内容紧密相关：你将构建一个小型应用程序，构建集群并部署应用程序。学习这两章时，你需要动手实践。

第1章

Spark介绍

本章内容涵盖

- Apache Spark是什么及其用例
- 分布式技术的基础
- Spark的四大支柱
- 存储和API：喜欢数据帧

在20世纪80年代，我还是一个孩童。通过Basic和Atari，我发现了编程的乐趣。当时，我不明白为什么不能使一些基本的执法活动自动化，如速度控制、闯红灯行为的处分和停车收费表。这一切似乎都很容易：我曾在书中说过，要成为一名优秀的程序员，就应该避免使用GOTO语句。我确实做到了这一点。从12岁开始，我就开始尝试使代码结构化。但是，当我还在开发《大富翁》这样的游戏时，我不可能想到，数据量会如此之大；还有蓬勃发展的物联网，也超出了我的想象。当64 KB的内存可容纳我的游戏时，我也绝对想不到数据集会变得如此之大(不在一个数量级上)；当我耐心地将游戏保存在Atari 1010录音带上时，也想不到数据传输还需要速度。

短短的35年后，我想象中的所有用例似乎都是可访问的(我的游戏没有得到任何结果)。数据快速增长，超过了支持它的硬件技术。[1]小型计算机集群的成本低于一台大型计算机的成本。与2005年相比，内存便宜了一半，而2005年的内存价格又比2000年的价格便宜了80%。[2]网络速度提高了很多，现代数据中心所提供的速度高达100Gbps，比5年前的家用WiFi快了近2000倍。所有这些因素促使人们提出下面这个问题：如何使用分布式内存计算来分析海量数据？

当你阅读文献，或在网上搜索有关Apache Spark的信息时，可能会发现它是一款大数据工具，是Hadoop的后继产品，是用于数据分析的平台，是集群计算机框架等。当然不是(Que nenni)！[3]

实验：本章实验的GitHub链接为：https://github.com/jgperrin/net.jgp.books.spark.ch01。这是实验#400。如果你不熟悉GitHub和Eclipse，那么可参阅附录A、B、C和D中提供的一些指导。

1 参见《麻省理工学院技术评论》汤姆·西蒙尼特(Tom Simonite)于2016年3月发表的“Intel Puts the Brakes on Moore's Law”(http://mng.bz/gVj8)。

2 参见约翰·麦卡勒姆(John C. McCallum)发表的“Memory Price (1957—2017)”(https://jcmit.net/memoryprice.htm)。

3 中世纪的法语表达，意思为：当然没有。

1.1 Spark 简介及其作用

正如小王子对安托万 • 德 • 圣艾修伯里(Antoine de Saint-Exupéry)说的那样：“Draw me a Spark(给我画出 Spark)。”在本节中，你将首先弄清楚 Spark 是什么，然后通过一些用例明白 Spark 能做些什么。第 1.1.1 节描述如何将 Spark 集成为软件堆栈，以及数据科学家如何使用 Spark。

1.1.1 什么是 Spark

对数据科学家而言，Spark 不只是软件堆栈。在构建应用程序时，你会将它们构建在操作系统之上，如图 1.1 所示。操作系统提供服务，使应用程序的开发更加轻松；换句话说，你不必为自己开发的每个应用程序构建文件系统或网络驱动程序。

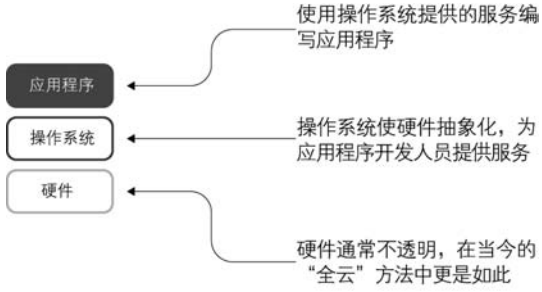

图 1.1 在编写应用程序时，使用操作系统提供的服务，使硬件抽象化

随着人们对算力的需求增长，对分布式计算的需求也相应增长。随着分布式计算的出现，分布式应用程序不得不集成这些分布式函数。图 1.2 显示了因向应用程序添加组件而使复杂性增加的情形。

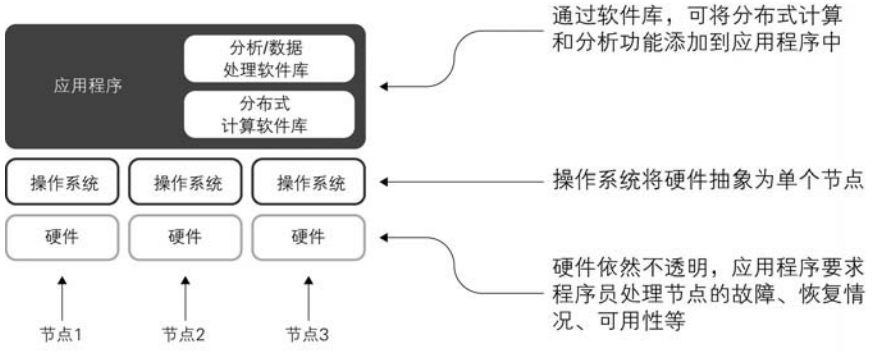

图 1.2 编写面向数据分布式应用程序的一种方法是使用库或其他工件，在应用程序层面嵌入所有控件。结果应用程序由于臃肿而变得难以维护

综上所述，Apache Spark 看似一个复杂的系统，需要你具备很多先验知识。但是，我坚信你只需要具备 Java 和关系数据库管理系统(Relational Database Management System，RDBMS)的相关技能就能理解、使用、构建带有 Spark 的应用程序，以及扩展 Spark。

应用程序也变得更加智能，可生成报告，执行数据分析(包括数据聚合、线性回归或简单地显示甜甜圈图)。因此，当你希望向应用程序添加此类分析功能时，必须链接库或构建自己的库。所有这些会使应用程序变得臃肿且复杂(如同一个胖客户端)，难以维护，进而增加企业成本。

你可能会问：“那么，为什么不在操作系统层面放置这些功能呢？”将这些功能放置在较低层次，如操作系统，会有很多益处，如下所示：

- 提供处理数据的一种标准方法(如同关系数据库中的结构化查询语言，即 SQL)。
- 降低应用程序的开发和维护成本。
- 使你可专注于理解如何使用该工具，而不必理解该工具的工作机制。(例如，Spark 执行分布式提取，你可学习如何从中受益，而不必完全理解 Spark 完成任务的方式。)

对我来说，这正是 Spark 转变的目标：一个分析操作系统。

图 1.3 显示了简化的技术栈。

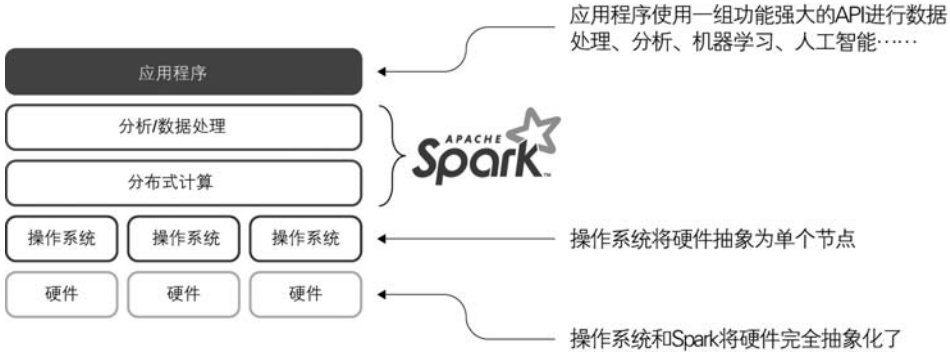

图 1.3　与操作系统一样，Apache Spark 通过向应用程序提供服务来简化面向分析的应用程序的开发

在本章中，你将发现基于不同行业和各种项目规模的 Apache Spark 的一些用例。这些示例将概述可实现的目标。

我坚信，若要更好地了解我们现在所处的位置，就应该回顾历史。这也适用于信息技术(Information Technology，IT)：如果你想了解相关信息，请阅读附录 E。

现在，场景已经准备就绪，你可深入研究 Spark 了。我们将从全局的角度出发，观察存储和 API，然后，开始演示第一个示例。

1.1.2　Spark 神力的四个支柱

根据波利尼西亚人的说法，神力(mana)为体现在物体或人体内的自然基本要素的力量。此定义适用于在所有 Spark 文档中都能找到的经典图表，其中显示了将这些基本要素带入 Spark 的四个支柱：Spark SQL、Spark Streaming、Spark MLlib(用于机器学习)以及位于 Spark Core 之上的 GraphX。尽管这精确表示了 Spark 技术栈，但我发现它具有局限性。这个技术栈需要扩展，以显示出硬件、操作系统和应用程序，如图 1.4 所示。

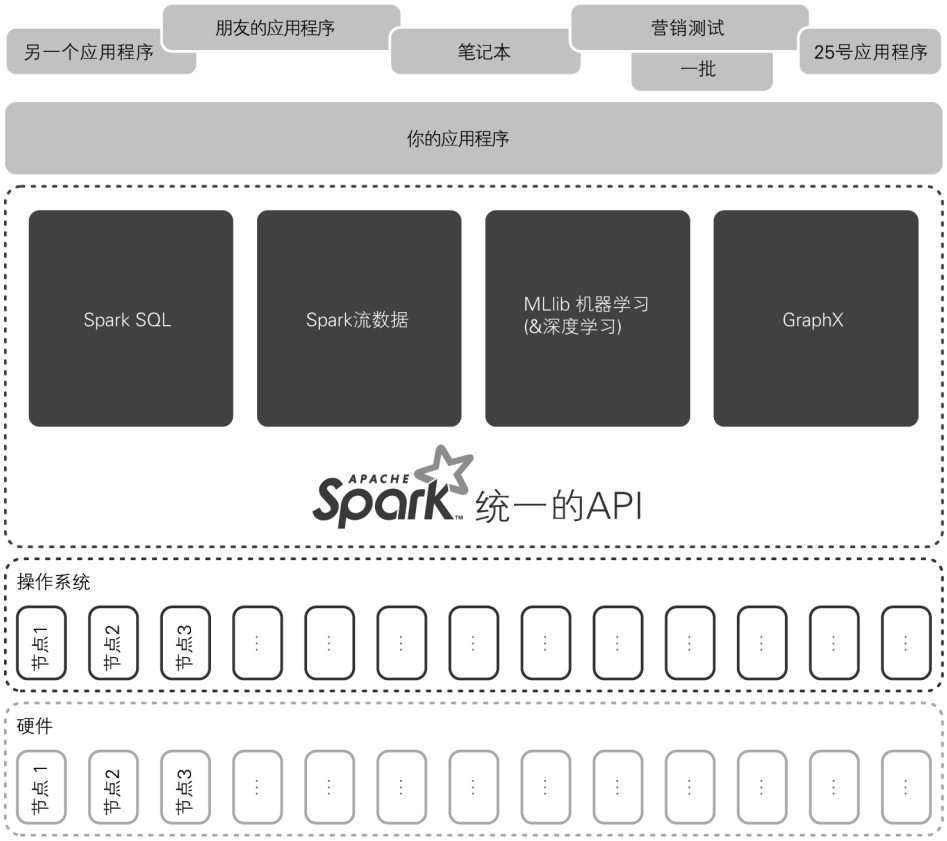

图 1.4 你的应用程序与其他应用程序通过统一的 API 与 Spark 的四个支柱(SQL、流式传输、机器学习和图)进行沟通。Spark 向程序员屏蔽了操作系统和硬件限制：程序员不必担心应用程序的运行，以及数据是否正确。Spark 将处理这些问题。但如有必要，应用程序仍可访问操作系统或硬件

单个应用程序当然不可能单独占用 Spark 集群，但是单个应用程序可使用由 Spark 提供的以下工具。

- 使用 Spark SQL 进行数据操作，如 RDBMS 中传统的 SQL 操作。Spark SQL 提供的 API 和 SQL 用来处理数据。在第 11 章中，你将学习 Spark SQL，并且将在之后的大部分章节中阅读到相关内容。Spark SQL 是 Spark 的基石。
- 使用 Spark Streaming，特别是 Spark 结构化的 Streaming 来分析流数据。Spark 的统一 API 有助于你使用相似的方式处理数据，无论是流数据还是批数据。在第 10 章中，你将了解有关流式传输的详细信息。
- 将 Spark MLlib 用于机器学习，以及深度学习的最新扩展中。机器学习、深度学习和人工智能的相关知识都值得单独成书。
- 利用图数据结构的 GraphX。要了解有关 GraphX 的更多信息，可阅读 Michael Malak 和 Robin East 编写的 *Spark GraphX in Action* 一书。

1.2　如何使用 Spark

在本节中，你将通过深入学习典型的数据处理场景以及数据科学场景，详细了解如何使用 Apache Spark。无论是数据工程师还是数据科学家，都可在工作中使用 Apache Spark。

1.2.1　数据处理/工程场景中的 Spark

Spark 可通过多种不同的方式处理数据。但在大数据场景中，如在提取、清理、转换和重新发布数据时，Spark 的表现尤其出色。

我倾向于将数据工程师视为数据准备者和数据后勤员。他们需要确保数据可用，确保成功应用了数据质量规则、成功执行了数据转换，并确保数据可用于其他系统或部门(包括业务分析师和数据科学家)。数据工程师也可承担数据科学家的工作，使数据产业化。

Spark 是数据工程师的理想工具。数据工程所执行的典型 Spark 场景(大数据)一般分为如下 4 个步骤：

(1) 提取

(2) 改善数据质量(Data Quality，DQ)

(3) 转换

(4) 发布

图 1.5 显示了此过程。

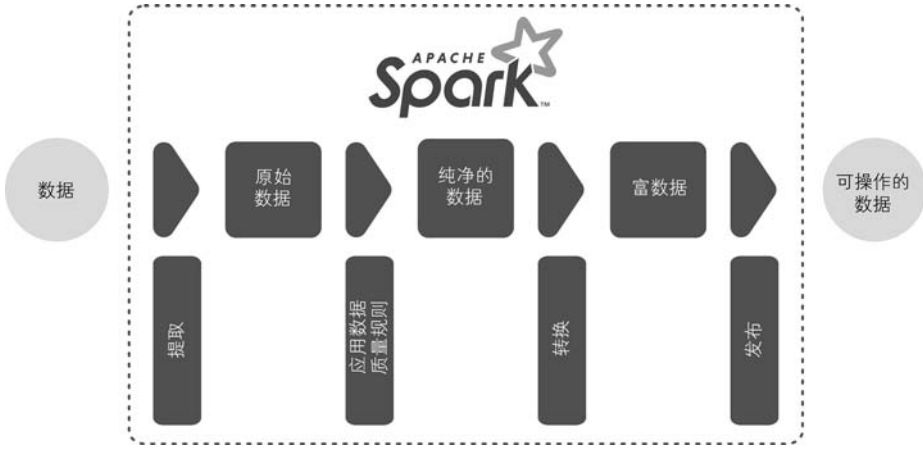

图 1.5　典型数据处理场景中的 Spark。第一步是提取数据。在这个阶段，数据未经处理(原始数据)；接下来需要提高数据质量(DQ)。然后，转换数据。一旦完成数据转换，数据就变得更加丰富。接着应该发布或共享该文件，这样组织中的人员才可对其执行操作，并基于数据做出决策

这个过程包括 4 个步骤。在每个步骤之后，数据会进入一个区域(zone)。

(1) 提取数据——Spark 可从多种来源提取数据(请参阅有关数据提取的第 7～9 章)。如果找不到所支持的格式，可构建自己的数据源。本书将此阶段的数据称为原始数据(raw data)。你可能还会发现，人们将此区域称为暂存(staging)、着陆(landing)、青铜(bronze)甚至是沼泽(swamp)区域。

(2) 提高数据质量(DQ)——在处理数据之前，需要检查数据本身的质量。DQ 的一个示例是确保所有出生日期都在当前日期之前。在此过程中，你可选择屏蔽一些数据：如果你在医疗保健环境中处理社会安全号码(Social Security Number，SSN)，那么需要确保开发人员或未经授权的人员无法访问 SSN[1]。此阶段在优化数据之后，本书称之为纯数据(pure data)区域。你可能还会发现，人们将此区域称为粗炼(refinery)、银矿(silver)、池化(pond)、沙盒(sandbox)或勘探(exploration)区域。

(3) 转换数据——下一步是处理数据。可将其并入其他数据集，应用自定义函数、执行聚合、实施机器学习等。此步骤的目标是获取分析工作的成果——包含丰富信息的数据。大多数章节都将讨论数据转换。也可将此区域称为产品(production)、黄金(gold)、精炼(refined)、泻湖(lagoon)或可操作(operationalization)区域。

(4) 加载和发布——与 ETL 流程[2]一样，可将数据加载到数据仓库中，使用商业智能(Business Intelligence，BI)工具，调用 API 或将数据保存到文件中，从而完成工作。所得结果是企业可操作的数据。

1.2.2 数据科学场景中的 Spark

数据科学家以交互的方式进行数据转换操作，因此他们采用的方法与软件工程师或数据工程师的方法略有不同。出于这个目的，数据科学家使用不同的工具，如笔记本(notebook)，包括 Jupyter、Zeppelin、IBM Watson Studio 和 Databricks Runtime。

数据科学项目可消费企业数据，因此人们最终有可能会将数据交付给数据科学家，将工作(如机器学习模型)卸载到企业数据存储中，或使其发现工业化。可见数据科学家的工作与人们的生活休戚相关。

因此，类似 UML 的序列图，如图 1.6 所示，可较好地解释数据科学家使用 Spark 的方式。

如果你希望进一步了解 Spark 和数据科学，可阅读以下这些书籍：

- *PySpark in Action*, Jonathan Rioux 著(Manning, 2020, www.manning.com/books/pyspark-in-action?a_aid=jgp)。
- *Mastering Large Datasets with Python*, John T. Wolohan 著(Manning, 2020, www.manning.com/books/mastering-large-datasets-with-python?a_aid=jgp)。

在图 1.6 所描述的用例中，数据被加载到 Spark 中，然后，用户操作数据，进行数据转换，显示部分数据。显示数据并不代表过程的结束。用户能以交互的方式继续操作数据，如在实体笔记本中一样，描述过程、做笔记等。最后，笔记本用户可将数据保存到文件或数据库中，或生成(交互式)报告。

1 如果你不住在美国，那么你需要了解 SSN 的重要性。它支配着人们的一生。它几乎与最初的目的没有关系：它曾是社会福利的标识符，现在成为税收标识符和财务跟踪器，与人们寸步不离。身份盗用者会寻找 SSN 和其他个人数据，这样他们就可开设银行账户或访问现有账户。

2 提取、转换和加载数据是经典的数据仓库过程。

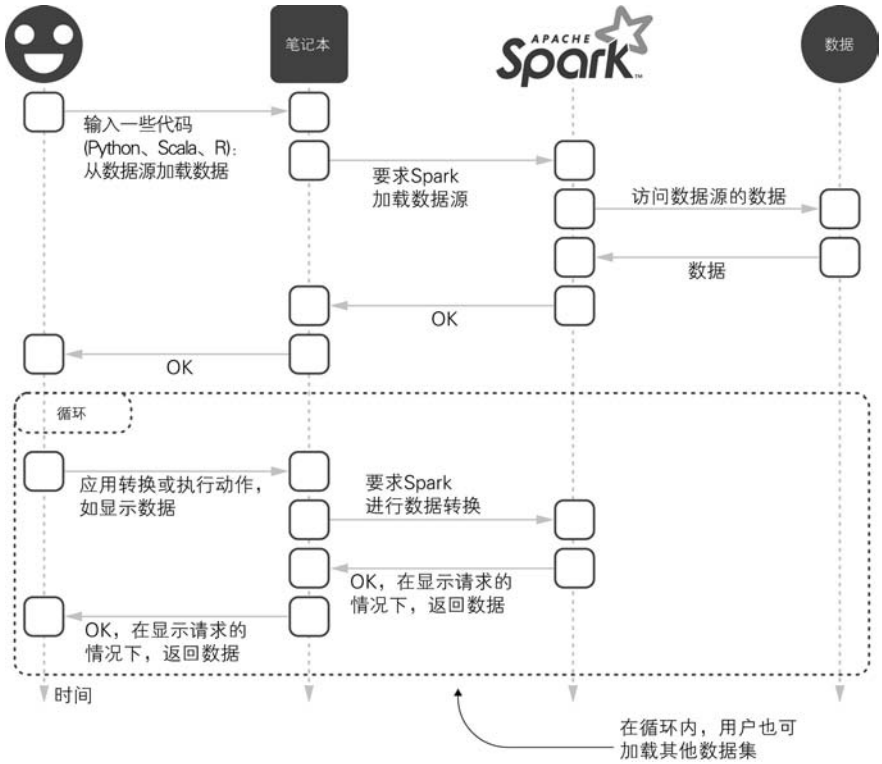

图 1.6　数据科学家使用 Spark 的序列图：用户与笔记本“交流”，需要时调用 Spark。Spark 直接处理数据提取。每个正方形代表一个步骤，每个箭头代表一个序列。应按时间顺序从顶部开始阅读该图表

1.3　使用 Spark，能做些什么

Spark 可用于各种不同的项目，下面我们探索其中几种项目。所有用例都涉及无法容纳在单台计算机上，或无法在单台计算机上处理的数据(也称大数据)，这需要计算机集群，因此需要专门用于分析的分布式操作系统。

大数据的定义随着时间的推移在逐步演变，从具有 5 个 V[1]特征的数据演变到“无法容纳在单台计算机的数据”，我不喜欢这个定义。也许正如你所知，许多 RDBMS 将数据拆分到多个服务器上。这与许多概念一样，你必须有自己的理解。希望本书会对你有所帮助。

对我而言，大数据就是数据集的集合，在企业中的任何地方都可获得。数据聚集在单个地点。在大数据上，你可运行基本分析，也可运行更高层次的分析，如机器学习和深度学习。这些较大的数据集可成为人工智能(AI)的基础。技术、大小或计算机数量都与这个概念无关。

通过 Spark 的分析功能和与生俱来的分布式架构，Spark 可处理大数据，这与人们普遍观念中的数据量大小，以及是否需要多台计算机处理无关。只需要记住，在 132 列点矩阵打印机上的传统

1　这 5 个 V 分别是体量(Volume，生成和存储的数据量)、多样性(Variety，数据的类型和性质)、速度(Velocity，生成和处理数据的速度)、差异性(Variability，数据集不一致)和准确性(Veracity，数据质量可能存在巨大差异)——改编自 Wikipedia 和 IBM。

报告输出并不是 Spark 的典型用例。让我们探索现实世界的一些示例。

1.3.1 使用 Spark 预测 NC 餐饮行业的餐馆质量

在美国的大部分地区，当地卫生部门需要对餐馆进行检查并基于这些检查进行打分，这样餐馆才能运营。较高的分数并不代表食物可口，但是这些分数可告诉食客，在前往南方的某个棚屋中吃烧烤后，是否会丧命。分数衡量了厨房的清洁度、食物存储的安全性等诸多标准，以避免食源性疾病。

在北卡罗来纳州，餐馆评分标准为 0～100。每个县都提供咨询服务，允许人们获取餐馆的分数，但对于全州而言，没有一个中央位置提供相关信息。

NCEatery.com 是一个面向消费者的网站，列出了餐馆在不同时间段的分数。NCEatery.com 的目标是集中这些信息，对餐馆进行预测分析，从而了解我们是否可发现餐馆质量的模式。我两年前所喜欢的餐馆，现在走下坡路了吗？

在网站后端，Apache Spark 接收来自不同县城的餐馆、其检查和违规的数据集，对数据进行整理并在网站上发布摘要。在整理阶段，应用一些数据质量规则，也尝试应用机器学习进行检查和评分。Spark 使用小型集群，每 18 个小时处理 1.6×10^{21} 个数据点，并发布了大约 2500 页的数据。这个项目正在如火如荼地开展，越来越多的县城加入进来。

1.3.2 Spark 允许 Lumeris 进行快速数据传输

Lumeris 是一家基于信息的医疗服务公司，位于密苏里州的圣路易斯市。传统上，该公司可帮助医疗保健提供商从数据中获取更多有内涵的信息。Lumeris 公司需要增强最先进的 IT 系统，以容纳更多的客户，从数据中获取更多更具内涵的信息。

在 Lumeris 的数据工程流程中，Apache Spark 提取存储在 Amazon Simple Storage Service(S3)中的数千份 CSV(Comma-separated Value，逗号分隔值)文件，构建符合医疗保健要求的 HL7 FHIR 资源[1]，将它们保存在专门的文档库中，这样现有的应用程序和新一代客户端应用程序都可使用它们。

这种技术栈使 Lumeris 在处理数据方面的能力可持续增长，其应用也随之变得多样。借助这种技术，Lumeris 旨在拯救生命。

1.3.3 Spark 分析 CERN 的设备日志

CERN(欧洲核研究组织)成立于 1954 年。它的大型强子对撞机(Large Hadron Collider，LHC)位于日内瓦，在法国和瑞士之间的边界处。机器在地下 100m，长 27km，呈环状。大型强子对撞机可进行规模宏大的物理实验，每秒生成 1 PB 数据。数据每天经过大规模的筛选后，减少为 900 GB。

在对 Oracle、Impala 和 Spark 进行实验之后，围绕 Spark，CERN 团队设计了下一代的 CERN 加速器日志服务(Next CERN Accelerator Logging Service，NXCALS)。Spark 被安装在运行 OpenStack

1 HL7(国际健康等级 7)是一家非营利性、经 ANSI 认证的标准开发组织，致力于促进电子健康信息的交换、集成、共享和检索。HL7 得到了来自 50 多个国家和地区的 1600 多个会员的支持。快速医疗保健互操作性资源(FHIR)是用于交换医疗保健信息的最新规范标准之一。

且具有 25 万个内核的内部自有云上。科学家(通过自定义应用程序和 Jupyter 笔记本)、开发人员和应用程序都可使用这个令人叹为观止的架构。CERN 有志于装载更多数据，提高数据处理的整体速度。

1.3.4　其他用例

Spark 也参与了许多其他用例，包括：

- 构建交互式数据整理工具，如 IBM 的 Watson Studio 和 Databricks 的笔记本。
- 监视 MTV 或 Nickelodeon[1]等电视频道的视频馈送质量。
- 通过 Riot Games 公司监控在线视频游戏玩家的不良行为，准实时调整玩家互动，使所有玩家的正面体验最大化。

1.4　为什么你应该喜欢数据帧

本节的目标是让你喜欢上数据帧。你将学到足够多的知识，并渴望有更多的发现，在第 3 章以及整本书中，你可进行更深入的探索。数据帧既是数据容器又是 API。

数据帧的概念对 Spark 至关重要，不过这个概念并不难以理解。你将一直使用数据帧。在本节中，你将从 Java(软件工程师)和 RDBMS(数据工程师)的角度了解数据帧是什么。在你逐渐熟悉其中一些相似之处时，本书会将它们总结为一个图表。

关于拼写问题

在大多数文献中，你会发现 dataframe 的另一种拼写方式：DataFrame。我决定采用英式拼写方式。我承认，对于法国人，这种拼写可能很奇怪。尽管 DataFrame 的拼写雄伟壮观，但它仍然是一个普通名词，因此没有理由使用大写字母。这又不是餐馆的名称。

1.4.1　从 Java 角度了解数据帧

如果你具有 Java 的背景，具有 Java 数据库连接(JDBC)的经验，那么对你而言数据帧看起来与 ResultSet 类似。它包含数据、API 等。

ResultSet 和数据帧的相似之处如下：

- 可通过简单的 API 访问数据。
- 可访问模式。

二者之间的一些区别如下：

- 不能使用 next()方法浏览数据。
- 其 API 可通过用户定义函数(UDF)扩展。可编写或封装现有代码，将其添加到 Spark。然后可在分布式模式下访问此代码。你将在第 16 章学习 UDF。
- 要访问数据，首先要获取行(Row)，然后使用 getter(与 ResultSet 类似)遍历行的各列。
- 在 Spark 中，元数据是基础，它们没有主键、外键或索引。

1　参见伯纳德 • 马尔于 2017 年 1 月发表在《福布斯》上的《 MTV 和 Nickelodeon 如何使用实时大数据分析改善客户体验》(http://bit.ly/2ynJvUt)。

在 Java 中，可使用 Dataset <Row>实现数据帧。

1.4.2 从 RDBMS 角度理解数据帧

如果你具有 RDBMS 背景，你会发现数据帧与表格一样。其相似之处如下：

- 使用列和行描述数据。
- 列为强类型。

二者间的一些区别如下：

- 数据可嵌套，与在 JSON 或 XML 文档中一样。第 7 章将介绍这些文档的提取方式，第 13 章将教你使用这些嵌套的构造。
- 不能更新或删除整行，但可创建新的数据帧。
- 可轻松添加或删除列。
- 数据帧上无约束、索引、主键(外键)、触发器。

1.4.3 数据帧的图形表示

数据帧是一个功能强大的工具。在本书中，以及在使用 Spark 的过程中，都需要使用数据帧。其强大的 API 和存储功能使其成为影响一切的关键元素。图 1.7 显示了表示 API、实现和存储的一种方法。

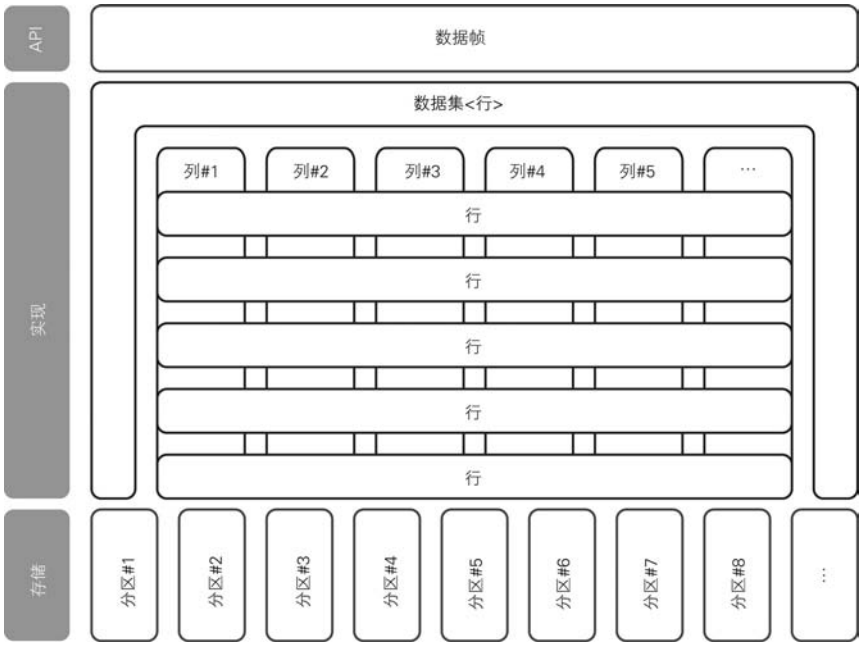

图 1.7 图形化表示的数据帧，使用 Java 实现(Dataset <Row>)、模式和分区存储。开发人员应使用数据帧 API，这样就可操作列和行，可访问分区中的存储，实现优化。在第 2 章中，你将学习更多关于分区的知识

1.5　第一个示例

现在是时候学习第一个示例了。本节的目标是教你使用简单的应用程序运行 Spark，读取文件，将文件内容存储在数据帧中，并显示结果。你将学习如何设置工作环境，在整本书中，你都将用到这个环境。你还将学习如何与 Spark 进行交互，并进行基本操作。

你会发现，大多数章节都包含专门的实验，你可研究这些代码，进行实验。每个实验都有一个数据集(尽可能使用真实的数据集)和一个或多个代码清单。

首先，你需要进行以下操作。

- 安装基本软件(你可能已经安装了)：Git、Maven、Eclipse。
- 从 GitHub 克隆代码，下载代码。
- 执行示例，加载基本 CSV 文件，显示一些行。

1.5.1　推荐软件

本节提供本书将使用的软件列表。附录 A 和附录 B 提供安装所需软件的详细说明。

本书将使用以下软件：

- Apache Spark 3.0.0。
- 主要是 macOS Catalina，但示例也可能运行在 Ubuntu 14～18 或 Windows 10 上。
- Java 8(尽管你不会使用版本 8 中引入的许多构造函数，如 lambda 函数)。据我所知，Java 11 已经可用，但是对于采用新版本，大多数企业反应缓慢(我发现 Oracle 最近的 Java 策略有点令人困惑)。到目前为止，只有 Spark v3 在 Java 11 上得到了认证。

这些示例将使用命令行或 Eclipse。在命令行的情况下，可使用以下命令。

- Maven：本书使用 3.5.2 版，但任何最近的版本都可使用。
- Git：使用 2.13.6 版本，但任何最新版本也可使用。在 macOS 上，可使用以 Xcode 包装的版本。在 Windows 上，可从 https://git-scm.com/download/win 下载。如果你喜欢图形用户界面(GUI)，强烈推荐你使用 Atlassian Sourcetree，可通过 www.sourcetreeapp.com 下载它。

项目使用 Maven 的 pom.xml 结构，在许多集成开发环境(IDE)中，该结构可被导入或直接使用。但是，所有可视示例都可使用 Eclipse。你可使用 4.7.1a(Eclipse Oxygen)之前的任何版本，但在 Eclipse 发布的 Oxygen 版本中，Maven 和 Git 集成得到了增强。强烈建议你至少使用 Oxygen 发行版，到目前为止，Oxygen 发行版已经相对较老了。

1.5.2　下载代码

源代码位于 GitHub 上的公共存储库中。存储库的 URL 是 https://github.com/jgperrin/net.jgp.books.spark.ch01。附录 D 详细描述了如何在命令行上使用 Git，以及如何使用 Eclipse 下载代码。

1.5.3 运行第一个应用程序

现在可运行应用程序了！如果你在运行应用程序时遇到了任何问题，请参阅附录 R。

1.命令行

在命令行上，切换工作目录：

```
$ cd net.jgp.books.spark.ch01
```

然后运行：

```
$ mvn clean install exec:exec
```

2. Eclipse

导入项目后(请参阅附录 D)，在项目资源管理器中找到 CsvToDataframeApp.java 文件。右击该文件，然后选择 Run As | 2 Java Application，如图 1.8 所示。在控制台中查看结果。

图 1.8　项目资源管理器中带有项目树的 Eclipse

无论是使用命令行还是 Eclipse，几秒钟后，结果都如下所示。

```
+---+--------+--------------------+------------+--------------------+
| id|authorId|               title| releaseDate|link|
+---+--------+--------------------+------------+--------------------+
|  1|       1|Fantastic Beasts ...|    11/18/16| http://amzn.to/2k...|
|  2|       1|Harry Potter and ...|     10/6/15| http://amzn.to/2l...|
|  3|       1|The Tales of Beed...|     12/4/08| http://amzn.to/2k...|
|  4|       1|Harry Potter and ...|     10/4/16| http://amzn.to/2k...|
|  5|       2|Informix 12.10 on...|     4/23/17| http://amzn.to/2i...|
+---+--------+--------------------+------------+--------------------+
仅显示前 5 行
```

现在，让我们了解发生了什么。

1.5.4　第一份代码

最后，进行编码！在上一节中，你看到了输出。现在是时候运行第一个应用程序了。该程序将获取会话，要求 Spark 加载 CSV 文件，然后显示数据集的 5 行(最多)数据。代码清单 1.1 提供了完整的程序。

在显示代码方面，存在两种思想流派：一种是显示摘要，另一种是显示所有代码。我支持后一流派：我喜欢完整的示例，而不是片段。我不希望在片段代码中，寻找不显示的代码或所需的软件包，即使这些缺失的代码显而易见。

代码清单 1.1　提取 CSV 文件

```
package net.jgp.books.spark.ch01.lab100_csv_to_dataframe;

import org.apache.spark.sql.Dataset;
import org.apache.spark.sql.Row;
import org.apache.spark.sql.SparkSession;

public class CsvToDataframeApp {

    public static void main(String[] args) {
        CsvToDataframeApp app = new CsvToDataframeApp();
        app.start();
    }

    private void start() {
        SparkSession spark = SparkSession.builder()
                .appName("CSV to Dataset")
                .master("local")
                .getOrCreate();

        Dataset<Row> df = spark.read().format("csv")
                .option("header", "true")
                .load("data/books.csv");

        df.show(5);
    }
}
```

main() 是应用程序的入口点

在当地的主服务器创建会话

使用表头(header)读取 CSV 文件，调用 books.csv，将其存储到数据帧中

数据帧中最多显示 5 行

尽管示例简单，但你已完成以下工作：

- 安装使用 Spark 所需的所有组件。(是的，就是这么简单！)
- 创建可执行代码的会话。
- 加载 CSV 数据文件。
- 显示此数据集的 5 行。

现在，你可更深入地了解 Apache Spark，进一步理解其背后的机制。

1.6 小结

- Spark 是一种分析操作系统，可使用此系统，以分布式的方式处理工作量和算法。这不仅适用于数据分析，也适用于数据传输、海量数据转换、日志分析等。
- Spark 支持以 SQL、Java、Scala、R 和 Python 作为编程接口；但本书将重点介绍 Java(有时是 Python)。
- Spark 内部的主要数据存储方式是数据帧。数据帧将存储容量与 API 结合在一起。
- 如果你具有 JDBC 开发的经验，那么你会发现这与 JDBC ResultSet 有相似之处。
- 如果你具有关系数据库开发的经验，那么可将数据帧与具有较少元数据的表格进行比较。
- 在 Java 中，使用 Dataset <Row>实现数据帧。
- 不必安装 Spark，就可快速设置 Spark，与 Maven 和 Eclipse 一同使用。
- Spark 不仅限于 MapReduce 算法：其 API 允许将多种算法应用于数据。
- 企业希望访问实时分析，它们越来越频繁地使用流媒体。Spark 支持流式传输。
- 分析已从简单的连接演化为聚合。企业希望计算机为其思考，因此 Spark 支持机器学习和深度学习。
- 图是分析的一种特殊用例，但 Spark 支持它们。