

---

## 第 6 章

---

---

---

# 数据存储与管理

数据存储与管理是大数据分析流程中的重要一环。通过数据采集得到的数据，必须进行有效的存储和管理，才能用于高效的处理和分析。数据存储与管理是利用计算机硬件和软件技术对数据进行有效的存储和应用的过程，其目的在于充分有效地发挥数据的作用。在大数据时代，数据存储与管理面临着巨大的挑战，一方面，需要存储的数据类型越来越多，包括结构化数据、半结构化数据和非结构化数据；另一方面，涉及的数据量越来越大，已经超出了很多传统数据存储与管理技术的处理范围。因此，大数据时代涌现出了大量新的数据存储与管理技术，包括分布式文件系统和分布式数据库等。

本章首先介绍传统的数据存储与管理技术，包括文件系统、关系数据库、数据仓库、并行数据库，然后介绍大数据时代的数据存储与管理技术，包括分布式文件系统、NewSQL 和 NoSQL 数据库、云数据库等，同时给出了一些代表性的产品，包括 Hadoop、HDFS、HBase、Spanner 等。

## 6.1 传统的数据存储与管理技术

传统的数据存储与管理技术包括文件系统、关系数据库、数据仓库和并行数据库等。

### 6.1.1 文件系统

操作系统中负责管理和存储文件信息的软件机构称为文件管理系统，简称“文件系统”。文件系统由三部分组成：文件系统的接口、对对象操纵和管理的软件集合、对象及属性。从系统角度来看，文件系统是对文件存储设备的空间进行组织和分配，负责文件存储，并对存入的文件进行保护和检索的系统。具体地说，它负责为用户建立文件，存入、读出、修改、转储文件，控制文件的存取，当用户不再使用时撤销文件等。

我们平时在计算机上使用的 Word 文件、PPT 文件、文本文件、音频文件、视频文件等，都由操作系统中的文件系统进行统一管理。

### 6.1.2 关系数据库

除了文件系统之外，数据库是另外一种主流的数据存储和管理技术。数据库指的是以一定方

式储存在一起、能为多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合。对数据库进行统一管理的软件被称为“数据库管理系统”(Database Management System, DBMS),在不引起歧义的情况下,经常会混用“数据库”和“数据库管理系统”这两个概念。在数据库的发展历史上,先后出现过网状数据库、层次数据库、关系数据库等不同类型的数据库,这些数据库分别采用了不同的数据模型(数据组织方式)。目前比较主流的数据库是关系数据库,它采用了关系数据模型来组织和管理数据。一个关系数据库可以看成许多关系表的集合,每个关系表可以看成一张二维表格,表 6-1 所示的为学生信息表。目前市场上常见的关系数据库产品包括 Oracle、SQL Server、MySQL、DB2 等。因为关系数据库的数据通常具有规范的结构,所以,通常把保存在关系数据库中的数据称为“结构化数据”。与此相对应,类似图片、视频、声音文件所包含的数据,没有规范的结构,被称为“非结构化数据”,而类似网页文件(如 HTML 文件)这种具有一定结构但又不是完全规范化的数据,被称为“半结构化数据”。

表 6-1 学生信息表

学号	姓名	性别	年龄	考试成绩
95001	张三	男	21	88
95002	李四	男	22	95
95003	王梅	女	22	73
95004	林莉	女	21	96

总体而言,关系数据库具有如下特点。

(1) 存储方式。关系数据库采用表格的存储方式,数据以行和列的方式进行存储,要读取和查询都十分方便。

(2) 存储结构。关系数据库按照结构化的方法存储数据,每个数据表的结构都必须事先定义好(比如表的名称、字段名称、字段类型、约束等),然后根据表的结构存入数据。这样做的好处是,由于数据的形式和内容在存入数据之前就已经定义好了,所以,整个数据表的可靠性和稳定性都比较高,但带来的问题是,数据模型不够灵活,一旦存入数据后,如果需要修改数据表的结构就会十分困难。

(3) 存储规范。关系数据库为了规范化数据、减少重复数据,以及充分利用存储空间,将数据按照最小关系表的形式进行存储,这样数据就可以变得很清晰、一目了然。当存在多个表时,表和表之间通过主外键关系发生关联,并通过连接查询获得相关结果。

(4) 扩展方式。由于关系数据库将数据存储和数据表中,数据操作的瓶颈出现在多张数据表的操作中,而且数据表越多这个问题越严重。如果要缓解这个问题,只能提高数据库处理能力,也就是选择处理速度更快、性能更高的计算机,这样的方法虽然具有一定的拓展空间,但是这样的拓展空间是非常有限的,也就是一般的关系数据库只具备有限的纵向扩展能力。

(5) 查询方式。关系数据库采用结构化查询语言(Structured Query Language, SQL)来对数据库进行查询。结构化查询语言是高级的非过程化编程语言,允许用户在高层数据结构上对数据

库进行操作。它不要求用户指定对数据的存放方法，也不需要用户了解具体的数据存放方式，所以，各种具有完全不同底层结构的数据库系统，可以使用相同的结构化查询语言作为数据输入与管理的接口。结构化查询语言语句可以嵌套，这使它具有极大的灵活性和强大的功能。

(6) 事务性。关系数据库可以支持事务的 ACID 特性——原子性 (Atomicity)、一致性 (Consistency)、隔离性 (Isolation)、持久性 (Durability)。当事务被提交给了 DBMS，则 DBMS 需要确保该事务中的所有操作都成功完成且其结果被永久保存在数据库中。如果事务中有的操作没有成功完成，则事务中的所有操作都需要被回滚，回到事务执行前的状态，从而确保数据库状态的一致性。

(7) 连接方式。不同的关系数据库产品都遵守一个统一的数据库连接接口标准，即开放数据库连接 (Open Database Connectivity, ODBC)。ODBC 的一个显著优点是，用它生成的程序是与具体的数据库产品无关的，这样可以为数据库用户和开发者屏蔽不同数据库异构环境的复杂性。ODBC 提供了数据库访问的统一接口，为应用程序实现与平台的无关性和可移植性提供了基础，因而获得了广泛的支持和应用。

### 6.1.3 数据仓库

数据仓库 (Data Warehouse) 是一个面向主题的、集成的、相对稳定的、反映历史变化的数据集合，用于支持管理决策。

(1) 面向主题。操作型数据库的数据组织面向事务处理任务，而数据仓库中的数据按照一定的主题域进行组织。主题是指用户使用数据仓库进行决策时所关心的重点，一个主题通常与多个操作型信息系统相关。

(2) 集成。数据仓库的数据来自分散的操作型数据，将所需数据从原来的数据中抽取出来，进行加工与集成、统一与综合之后才能进入数据仓库。

(3) 相对稳定。数据仓库是不可更新的。数据仓库主要为决策分析提供数据，所涉及的操作主要是数据的查询。

(4) 反映历史变化。在构建数据仓库时，会每隔一定的时间 (比如每周、每天或每小时) 从数据源抽取数据并加载到数据仓库，比如，1月1日晚上12点“抓拍”数据源中的数据保存到数据仓库，然后1月2日、1月3日一直到月底，每天“抓拍”数据源中的数据保存到数据仓库，这样，经过一个月以后，数据仓库中就会保存1月份每天的数据“快照”，由此得到的31份数据“快照”，就可以用来进行商务智能分析，比如，分析一个商品在1个月内的销量变化情况。

综上所述，数据库是面向事务的设计，数据仓库是面向主题的设计。数据库一般存储在线交易数据，数据仓库存储的一般是历史数据。数据库为捕获数据而设计，数据仓库为分析数据而设计。

一个典型的数据仓库系统通常包含数据源、数据存储和管理、OLAP 服务器、前端工具和应用等4个部分，如图6-1所示。

(1) 数据源。数据源是数据仓库的基础，即系统的数据来源，通常包含企业的各种内部数据和外部数据。内部数据包括存在于联机事务处理 (On-Line Transaction Processing, OLTP) 系统中

的各种业务数据和办公自动化系统中的各类文档资料等。外部数据包括各类法律法规、市场信息、竞争对手的信息，以及各类外部统计数据和其他相关文档等。

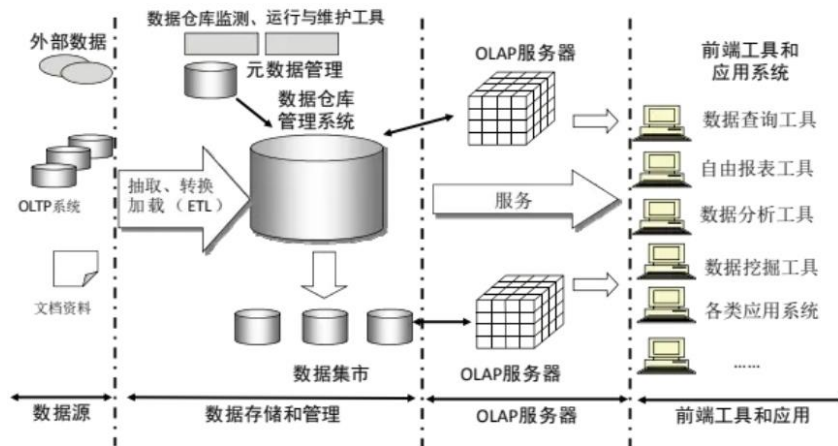


图 6-1 数据仓库体系架构

(2) 数据存储和管理。数据存储和管理是整个数据仓库的核心。在现有各业务系统的基础上，对数据进行抽取、转换，并加载到数据仓库中，按照主题进行重新组织，最终确定数据仓库的物理存储结构，同时存储数据库的各种元数据（包括数据仓库的数据字典、记录系统定义、数据转换规则、数据加载频率以及业务规则等）。对数据仓库系统的管理，也就是对相应数据库系统的管理，通常包括数据的安全、归档、备份、维护和恢复等工作。

(3) OLAP 服务器。OLAP 服务器对需要分析的数据按照多维数据模型进行重组，以支持用户随时从多角度、多层次来分析数据，发现数据规律与变化趋势。

(4) 前端工具和应用。前端工具和应用主要包括数据查询工具、自由报表工具、数据分析工具、数据挖掘工具和各类应用系统。

## 6.1.4 并行数据库

并行数据库是指那些在无共享的体系结构中进行数据操作的数据库系统。这些系统大部分采用了关系数据模型并且支持 SQL 语句查询。但为了能够并行执行 SQL 的查询操作，系统中采用了两个关键技术：关系表的水平划分和 SQL 查询的分区执行。并行数据库的目标是高性能和高可用性，通过多个节点并行执行数据库任务，提高整个数据库系统的性能和可用性。最近这几年涌现出一些提高系统性能的新技术，如索引、压缩、实体化视图、结果缓存、I/O 共享等，这些技术都比较成熟且经得起时间的考验。与一些早期的系统（如 Teradata 等）必须部署在专有硬件上不同，最近开发的系统（如 Aster、Vertica 等）可以部署在普通的商业机器上。

并行数据库的主要缺点是没有较好的弹性，而这种特性对中小企业和初创企业是有利的。人们在对并行数据库进行设计和优化的时候认为集群中节点的数量是固定的，若需要对集群进行扩展和收缩，则必须为数据转移过程制订周全的计划。这种数据转移的代价是昂贵的，并且会导致

系统在某段时间内不可访问，而这种较差的灵活性直接影响到并行数据库的弹性和现用现付商业模式的实用性。

并行数据库的另一个问题是系统的容错性较差。过去人们认为节点故障是个特例，并不经常出现，因此系统只提供事务级别的容错功能，如果在查询过程中节点发生故障，那么整个查询都要从头开始执行。这种重启任务的策略使得并行数据库难以在拥有数千个节点的集群上处理较长的查询，因为在这类集群中节点的故障经常发生。基于这种分析，并行数据库只适合于资源需求相对固定的应用程序。不管怎样，并行数据库的许多设计原则为其他海量数据系统的设计和优化提供了比较好的借鉴。

## 6.2 大数据时代的数据存储与管理技术

本节介绍大数据时代的数据存储与管理技术，包括分布式文件系统、NewSQL 和 NoSQL 数据库、云数据库等。

### 6.2.1 分布式文件系统

大数据时代必须解决海量数据的高效存储问题，为此，分布式文件系统（Distributed File System）应运而生。相对于传统的本地文件系统而言，分布式文件系统是一种通过网络实现文件在多台主机上进行分布式存储的文件系统。分布式文件系统的设计一般采用“客户端/服务器”（Client/Server）模式，客户端以特定的通信协议通过网络与服务器建立连接，提出文件访问请求，客户端和服务器可以通过设置访问权来限制请求方对底层数据存储块的访问。

谷歌开发了分布式文件系统（Google File System, GFS），通过网络实现文件在多台机器上的分布式存储，较好地满足了大规模数据存储的需求。Hadoop 分布式文件系统（Hadoop Distributed File System, HDFS）是针对 GFS 的开源实现，它是 Hadoop 两大核心组成部分之一，提供了在廉价服务器集群中存储大规模分布式文件的能力。HDFS 具有很好的容错能力，并且兼容廉价的硬件设备，因此，可以以较低的成本利用现有机器实现大流量和大数据量的读写操作。

### 6.2.2 NewSQL 和 NoSQL 数据库

传统的关系数据库（这里可以称为“OldSQL 数据库”）可以较好地支持结构化数据存储和管理，它以完善的关系代数理论作为基础，具有严格的标准，支持事务 ACID 四性，借助索引机制可以实现高效的查询，因此，关系数据库自从 20 世纪 70 年代诞生以来就一直是数据库领域的主流产品类型。但是，Web 2.0 时代的迅速发展以及大数据时代的到来，使关系数据库越来越力不从心。在大数据时代，数据类型繁多，包括结构化数据和各种非结构化数据等，其中，非结构化数据在其中所占的比例更是高达 90% 以上。传统的关系数据库由于数据模型不灵活、横向扩展能力较差等局限性，已经无法满足各种类型的非结构化数据的大规模存储需求。不仅如此，传统关

系数据库引以为豪的一些关键特性，如事务机制和支持复杂查询，在 Web 2.0 时代的很多应用中都成为“鸡肋”。因此，在新的应用需求驱动下，各种新型数据库不断涌现，并逐渐获得市场的青睐。新型数据库主要包括 NewSQL 数据库和 NoSQL 数据库等。

### 1. NewSQL 数据库

NewSQL 是各种新的可扩展、高性能数据库的简称，这类数据库不仅具有对海量数据的存储管理能力，还保持了传统数据库支持 ACID 和 SQL 等特性。不同的 NewSQL 数据库的内部结构差异很大，但是，它们有两个显著的共同特点：都支持关系数据模型；都使用 SQL 作为其主要的接口。

目前具有代表性的 NewSQL 数据库主要包括 Spanner、Clustrix、GenieDB、ScalArc、Schooner、VoltDB、RethinkDB、ScaleDB、Akiban、CodeFutures、ScaleBase、Translattice、NimbusDB、Drizzle、Tokutek、JustOneDB 等，此外，还有一些在云端提供的 NewSQL 数据库，包括 Amazon RDS、Microsoft SQL Azure、Xeround 和 FathomDB 等。在众多 NewSQL 数据库中，Spanner 备受瞩目。它是一个可扩展、多版本、全球分布式并且支持同步复制的数据库，是 Google 的第一个可以全球扩展并且支持外部一致性的数据库。Spanner 能做到这些，离不开一个用 GPS 和原子钟实现的时间 API。这个 API 能将数据中心之间的时间同步精确到 10ms 以内。

一些 NewSQL 数据库比传统的关系数据库具有明显的性能优势。比如，VoltDB 系统使用了 NewSQL 创新的体系架构，释放了主内存运行的数据库中消耗系统资源的缓冲池，在执行交易时可比传统关系数据库快 45 倍。VoltDB 可扩展服务器数量为 39 个，并可以每秒处理 160 万个交易（300 个 CPU 核心），而具备同样处理能力的 Hadoop 需要更多的服务器。

### 2. NoSQL 数据库

NoSQL 数据库具有一种不同于关系数据库的数据库管理系统设计方式，是对非关系数据库的统称，它所采用的数据模型并非传统关系数据库的关系模型，而是类似键值、列族、文档等非关系模型。NoSQL 数据库没有固定的表结构，通常也不存在连接操作，也没有严格遵守 ACID 约束，因此，与关系数据库相比，NoSQL 具有灵活的水平可扩展性，可以支持海量数据存储。此外，NoSQL 数据库支持 MapReduce 风格的编程，可以较好地应用于大数据时代的各种数据管理。NoSQL 数据库的出现，一方面弥补了关系数据库在当前商业应用中存在的各种缺陷，另一方面也撼动了关系数据库的传统垄断地位。

典型的 NoSQL 数据库通常包括键值数据库、列族数据库、文档数据库和图数据库。近些年，NoSQL 数据库发展势头非常迅猛。在四五年时间内，NoSQL 领域就爆炸性地产生了 50~150 个新的数据库。据一项网络调查显示，行业中最需要开发者掌握的“技能”前十名依次是 HTML5、MongoDB、iOS、Android、Mobile Apps、Puppet、Hadoop、jQuery、PaaS 和 Social Media。可以看出，其中 MongoDB（一种文档数据库，属于 NoSQL）的热度甚至高于 iOS，足以看出 NoSQL 的受欢迎程度。

当应用场合需要简单的数据模型、灵活性的 IT 系统、较高的数据库性能和较低的数据库一致性时，NoSQL 数据库是一个很好的选择。通常 NoSQL 数据库具有以下几个特点。

(1) 灵活的可扩展性。传统的关系数据库由于自身设计的局限性, 通常很难实现“横向扩展”。当数据库负载大规模增加时, 往往需要通过升级硬件来实现“纵向扩展”。但是, 当前的计算机硬件制造工艺已经达到一个限度, 性能提升的速度开始趋缓, 已经远远赶不上数据库系统负载的增加速度, 而且, 配置高端的高性能服务器价格不菲, 因此, 寄希望于通过“纵向扩展”满足实际业务需求, 已经变得越来越不现实。相反, “横向扩展”仅需要使用非常普通且廉价的标准化刀片服务器, 它不仅具有较高的性价比, 也提供了理论上近乎无限的扩展空间。NoSQL 数据库在设计之初就是为了满足“横向扩展”的需求, 因此, 天生具备良好的横向扩展能力。

(2) 灵活的数据模型。关系数据模型是关系数据库的基石, 它以完备的关系代数理论为基础, 具有规范的定义, 遵守各种严格的约束条件。关系数据模型虽然保证了业务系统对数据一致性的需求, 但是, 过于死板的数据模型意味着无法满足各种新兴的业务需求。相反, NoSQL 数据库天生就旨在摆脱关系数据库的各种束缚条件, 摒弃了流行多年的关系数据模型, 转而采用键值、列族等非关系数据模型, 允许在一个数据元素里存储不同类型的数据。

(3) 与云计算紧密融合。云计算具有很好的横向扩展能力, 可以根据资源使用情况进行自由伸缩, 各种资源可以动态加入或退出。NoSQL 数据库可以凭借自身良好的横向扩展能力, 充分利用云计算基础设施, 很好地将数据库融入云计算环境, 构建基于 NoSQL 的云数据库服务。

### 3. 大数据引发数据库架构变革

综合来看, 大数据时代的到来, 引发了数据库架构的变革。以前, 业界和学术界追求的是一种架构支持多类应用 (One Size Fits All), 如图 6-2 所示, 包括事务型应用 (OLTP 系统)、分析型应用 (OLAP、数据仓库) 和互联网应用 (Web 2.0)。但是, 实践证明, 这种理想愿景是不可能实现的, 不同应用场景的数据管理需求截然不同, 一种数据库架构根本无法满足所有场景。因此, 到了大数据时代, 数据库架构开始向着多元化方向发展, 并形成了传统关系数据库 (OldSQL)、NoSQL 数据库和 NewSQL 数据库 3 个阵营, 三者各有自己的应用场景和发展空间。尤其是传统关系数据库, 并没有就此被其他两者完全取代。在基本架构不变的基础上, 许多关系数据库产品开始引入内存计算和一体机技术以提升处理性能。在未来一段时间内, 3 个阵营共存的局面还将持续, 不过有一点是肯定的, 那就是传统的关系数据库辉煌的时期已经过去了。

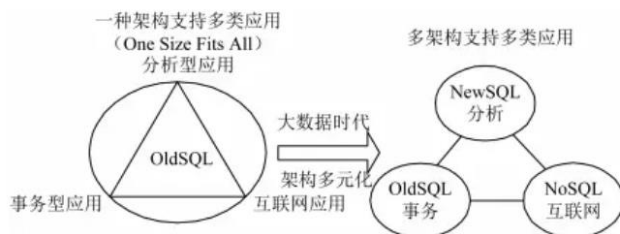


图 6-2 大数据引发数据库架构变革

## 6.2.3 云数据库

研究机构 IDC 预言, 大数据将按照每年 60% 的速度增加, 其中包含结构化和非结构化数据。

如何方便、快捷、低成本地存储海量数据，是许多企业和机构面临的一个严峻挑战。云数据库是一个非常好的解决方案，目前云服务提供商正通过云技术推出更多可在公有云中托管数据库的方法，将用户从烦琐的数据库硬件定制中解放出来，同时让用户拥有强大的数据库扩展能力，满足海量数据的存储需求。此外，云数据库还能够很好地满足企业动态变化的数据存储需求和中小企业的低成本数据存储需求。可以说，在大数据时代，云数据库将成为许多企业数据的目的地。

为了更加清晰地认识传统关系数据库、NoSQL、NewSQL 和云数据库的相关产品，图 6-3 给出了 4 种数据库相关产品的分类情况。

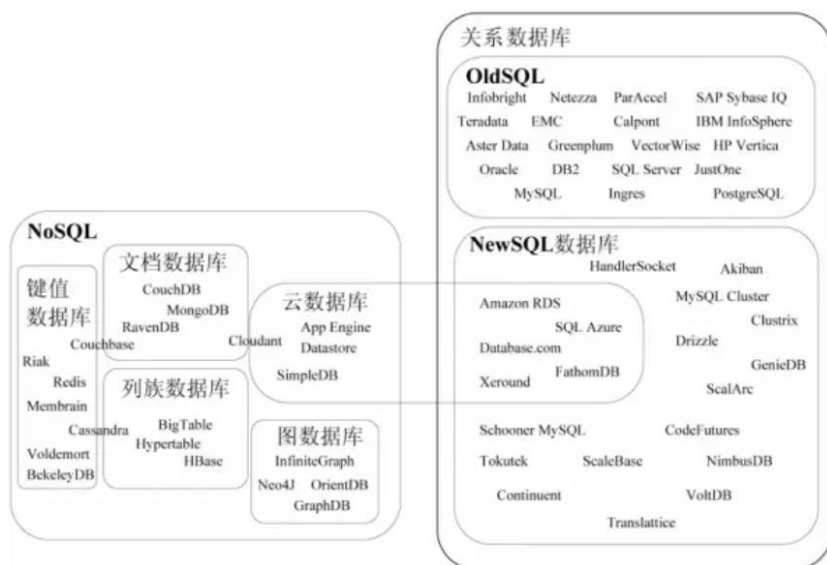


图 6-3 关系数据库、NoSQL、NewSQL 和云数据库相关产品的分类情况

## 6.3 大数据处理架构 Hadoop

Hadoop 是 Apache 软件基金会旗下的一个开源分布式计算平台，为用户提供系统底层细节透明的分布式基础架构。它实现了 MapReduce 计算模型和分布式文件系统 HDFS 等功能，在业内得到了广泛的应用。借助于 Hadoop，程序员可以轻松地编写分布式并行程序，将其运行于计算机集群上，完成海量数据的存储与处理分析。本章后面将要介绍的分布式文件系统 HDFS 和分布式数据库 HBase，都是 Hadoop 生态系统的组成部分。当然，Hadoop 不仅包括 HDFS 这个实现存储功能的组件，还包括实现计算功能的组件 MapReduce( 在第 7 章介绍)。因此，在介绍 HDFS 和 HBase 之前，这里先简要介绍一下 Hadoop。

### 6.3.1 Hadoop 特性

Hadoop 是一个能够对大量数据进行分布式处理的软件框架，并且能以一种可靠、高效、可伸

缩的方式进行处理，它具有以下几个方面的特性。

(1) 高可靠性。Hadoop 采用冗余数据存储方式，即使一个副本发生故障，其他副本也可以保证正常对外提供服务。

(2) 高效性。作为并行分布式计算平台，Hadoop 采用分布式存储和分布式处理两大核心技术，能够高效地处理 PB 级数据。

(3) 高可扩展性。Hadoop 的设计目标是可以高效稳定地运行在廉价的计算机集群上，可以扩展到数以千计的计算机节点上。

(4) 高容错性。Hadoop 采用冗余数据存储方式，自动保存数据的多个副本，并且能够自动将失败的任务进行重新分配。

(5) 成本低。Hadoop 采用廉价的计算机集群，成本比较低，普通用户也很容易用自己的 PC 搭建 Hadoop 运行环境。

(6) 运行在 Linux 操作系统上。Hadoop 是基于 Java 开发的，可以较好地运行在 Linux 操作系统上。

(7) 支持多种编程语言。Hadoop 上的应用程序也可以使用其他语言编写，如 C++。

Hadoop 凭借其突出的优势，已经在各个领域得到了广泛的应用，而互联网领域是其应用的主阵地。国内采用 Hadoop 的公司主要有百度、淘宝、网易、华为、中国移动等。

### 6.3.2 Hadoop 生态系统

经过多年的发展，Hadoop 生态系统不断完善和成熟，目前已经包含了多个子项目(见图 6-4)。除了核心的 HDFS 和 MapReduce 以外，Hadoop 生态系统还包括 ZooKeeper、HBase、Hive、Pig、Mahout、Sqoop、Flume、Ambari 等。

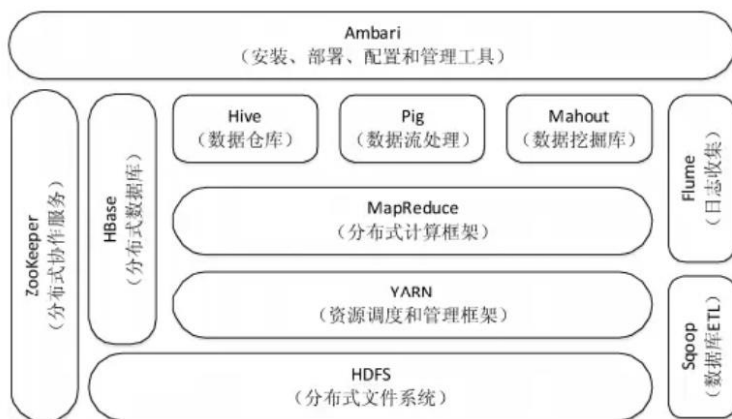


图 6-4 Hadoop 生态系统子项目

#### 1. HDFS

Hadoop 分布式文件系统(HDFS)是 Hadoop 项目的两大核心之一,是针对谷歌文件系统(GFS)的开源实现。HDFS 具有可处理超大数据、流式处理、可以运行在廉价商用服务器上等优点。HDFS

在设计之初就是要运行在廉价的大型服务器集群上，因此在设计上就把硬件故障作为一种常态来考虑，实现在部分硬件发生故障的情况下仍然能够保证文件系统的整体可用性和可靠性。HDFS 放宽了一部分可移植操作系统接口（Portable Operating System Interface, POSIX）约束，从而实现以流的形式访问文件系统中的数据。HDFS 在访问应用程序数据时，可以具有很高的吞吐率，因此对于超大数据集的应用程序而言，选择 HDFS 作为底层数据存储系统是较好的选择。

## 2. HBase

HBase 是一个提供高可靠性、高性能、可伸缩、实时读写、分布式的列式数据库，一般采用 HDFS 作为其底层数据存储系统。HBase 是针对谷歌 Bigtable 的开源实现，二者采用了相同的数据模型，具有强大的非结构化数据存储能力。HBase 与传统关系数据库的一个重要区别是，前者采用基于列的存储，后者采用基于行的存储。HBase 具有良好的横向扩展能力，可以通过不断增加廉价的商用服务器来提高存储能力。

## 3. MapReduce

Hadoop MapReduce 是针对谷歌 MapReduce 的开源实现。MapReduce 是一种编程模型，用于大规模数据集（大于 1 TB）的并行运算，它将复杂的、运行于大规模集群上的并行计算过程高度地抽象到两个函数——Map 和 Reduce 上，并且允许用户在不了解分布式系统底层细节的情况下开发并行应用程序，并将其运行于廉价的计算机集群上，完成海量数据的处理。通俗地说，MapReduce 的核心思想就是“分而治之”。它把输入的数据集切分为若干独立的小数据块，分发给一个主节点管理下的各个分节点来共同并行完成。最后，通过整合各个节点的中间结果得到最终结果。

## 4. Hive

Hive 是一个基于 Hadoop 的数据仓库工具，可以用于对 Hadoop 文件中的数据集进行数据整理、特殊查询和分析存储。Hive 的学习门槛较低，因为它提供了类似于关系数据库 SQL 的查询语言——HiveQL，可以通过 HiveQL 语句快速实现简单的 MapReduce 任务，Hive 自身可以将 HiveQL 语句转换为 MapReduce 任务运行，而不必开发专门的 MapReduce 应用，因而十分适合数据仓库的统计分析。

## 5. Pig

Pig 是一种数据流语言和运行环境，适合于使用 Hadoop 和 MapReduce 平台来查询大型半结构化数据集。虽然 MapReduce 应用程序的编写不是十分复杂，但毕竟也需要一定的开发经验。Pig 的出现大大简化了 Hadoop 常见的工作任务，它在 MapReduce 的基础上创建了更简单的过程语言抽象，为 Hadoop 应用程序提供了一种更加接近 SQL 的接口。Pig 是一种相对简单的语言，它可以执行语句，因此当我们需要从大型数据集中搜索满足某个给定搜索条件的记录时，采用 Pig 要比 MapReduce 具有明显的优势，前者只需要编写一个简单的脚本在集群中自动并行处理与分发，后者则需要编写一个单独的 MapReduce 应用程序。

## 6. Mahout

Mahout 是 Apache 软件基金会旗下的一个开源项目，提供一些可扩展的机器学习领域经典算法的实现，旨在帮助开发者更加方便快捷地创建智能应用程序。Mahout 包含许多实现，如聚类、

分类、推荐过滤、频繁子项挖掘等。此外，通过使用 Apache Hadoop 库，Mahout 可以有效地扩展到云中。

### 7. ZooKeeper

ZooKeeper 是针对谷歌 Chubby 的一个开源实现，是高效和可靠的协同工作系统，提供分布式锁之类的基本服务（如统一命名服务、状态同步服务、集群管理、分布式应用配置项的管理等），用于构建分布式应用，减轻分布式应用程序所承担的协调任务。ZooKeeper 使用 Java 编写，很容易编程接入，它使用了一个和文件树结构相似的数据模型，可以使用 Java 或者 C 来进行编程接入。

### 8. Flume

Flume 是 Cloudera 提供的一个高可用的、高可靠的、分布式的海量日志采集、聚合和传输系统。Flume 支持在日志系统中定制各类数据发送方，用于收集数据；同时，Flume 提供对数据进行简单处理并写到各种数据接收方的能力。

### 9. Sqoop

Sqoop 是 SQL-to-Hadoop 的缩写，主要用来在 Hadoop 和关系数据库之间交换数据，可以改进数据的互操作性。通过 Sqoop 可以方便地将数据从 MySQL、Oracle、PostgreSQL 等关系数据库中导入 Hadoop（可以导入 HDFS、HBase 或 Hive），或者将数据从 Hadoop 导出到关系数据库，使得传统关系数据库和 Hadoop 之间的数据迁移变得非常方便。Sqoop 主要通过 Java 数据库连接（Java DataBase Connectivity, JDBC）和关系数据库进行交互，理论上，支持 JDBC 的关系数据库都可以使用 Sqoop 和 Hadoop 进行数据交互。Sqoop 是专门为大数据集设计的，支持增量更新，可以将新记录添加到最近一次导出的数据源上，或者指定上次修改的时间戳。

### 10. Ambari

Apache Ambari 是一种基于 Web 的工具，支持 Apache Hadoop 集群的安装、部署、配置和管理。Ambari 目前已支持大多数 Hadoop 组件，包括 HDFS、MapReduce、Hive、Pig、HBase、ZooKeeper、Sqoop 等。

## 6.4 分布式文件系统 HDFS

分布式文件系统 HDFS 开源实现了 GFS 的基本思想。HDFS 原来是 Apache Nutch 搜索引擎的一部分，后来独立出来作为一个 Apache 子项目，并和 MapReduce 一起成为 Hadoop 的核心组成部分。HDFS 支持流数据读取和处理超大规模文件，并能够运行在由廉价的普通机器组成的集群上，这主要得益于 HDFS 在设计之初就充分考虑了实际应用环境的特点，那就是，硬件出错在普通服务器集群中是一种常态，而不是异常。因此，HDFS 在设计上采取了多种机制保证在硬件出错的环境中实现数据的完整性。

本节介绍 HDFS 的设计目标和 HDFS 体系结构。

## 6.4.1 HDFS 的设计目标

总体而言，HDFS 要实现以下目标。

(1) 兼容廉价的硬件设备。在成百上千台廉价服务器中存储数据，常会出现节点失效的情况，因此 HDFS 设计了快速检测硬件故障和进行自动恢复的机制，可以实现持续监视、错误检查、容错处理和自动恢复，从而在硬件出错的情况下也能实现数据的完整性。

(2) 流数据读写。普通文件系统主要用于随机读写以及与用户进行交互，而 HDFS 是为了满足批量数据处理的要求而设计的，因此为了提高数据吞吐率，HDFS 放松了一些 POSIX 的要求，从而能够以流式方式来访问文件系统数据。

(3) 大数据集。HDFS 中的文件通常可以达到 GB 甚至 TB 级别，一个数百台服务器组成的集群可以支持千万级别这样的文件。

(4) 简单的文件模型。HDFS 采用了“一次写入、多次读取”的简单文件模型，文件一旦完成写入，关闭后就无法再次写入，只能被读取。

(5) 强大的跨平台兼容性。HDFS 是采用 Java 实现的，具有很好的跨平台兼容性，支持 JVM 的机器都可以运行 HDFS。

HDFS 特殊的设计，在实现上述优良特性的同时，也使得自身具有一些应用局限性，主要包括以下几个方面。

(1) 不适合访问低延迟数据。HDFS 主要是面向大规模数据批量处理而设计的，采用流式数据读取，具有很高的数据吞吐率，但是，这也意味着较高的延迟。因此，HDFS 不适合用在需要低延迟（如数十毫秒）的应用场合。对于具有低延迟要求的应用程序而言，HBase 是一个更好的选择。

(2) 无法高效存储大量小文件。小文件是指文件大小小于一个块的文件。HDFS 无法高效存储和处理大量小文件，过多小文件会给系统扩展性和性能带来诸多问题。首先，HDFS 采用名称节点来管理文件系统的元数据，这些元数据被保存在内存中，从而使客户端可以快速获取文件实际存储位置。通常，每个文件、目录和块大约占 150Byte，如果有 1000 万个文件，每个文件对应一个块，那么，名称节点至少要消耗 3GB 的内存来保存这些元数据信息。很显然，这时元数据检索的效率就比较低了，需要花费较多的时间找到一个文件的实际存储位置。而且，如果继续扩展到数十亿个文件，名称节点保存元数据所需要的内存空间就会大大增加，以现有的硬件水平，是无法在内存中保存如此大量的元数据的。其次，用 MapReduce 处理大量小文件时，会产生过多的 Map 任务，线程管理开销会大大增加，因此处理大量小文件的速度远远低于处理同等规模的大文件的速度。再次，访问大量小文件的速度远远低于访问几个大文件的速度，因为访问大量小文件，需要不断从一个数据节点跳到另一个数据节点，严重影响性能。

(3) 不支持多用户写入及任意修改文件。HDFS 只允许一个文件有一个写入者，不允许多个用户对同一个文件执行写操作，而且只允许对文件执行追加操作，不能执行随机写操作。

## 6.4.2 HDFS 体系结构

HDFS 采用了主从 (Master/Slave) 结构模型, 一个 HDFS 集群包括一个名称节点和若干个数据节点 (见图 6-5)。名称节点作为中心服务器, 负责管理文件系统的命名空间及客户端对文件的访问。集群中的数据节点一般是一个节点运行一个数据节点进程, 负责处理文件系统客户端的读/写请求, 在名称节点的统一调度下进行数据块的创建、删除和复制等操作。每个数据节点的数据实际上是保存在本地 Linux 文件系统上的。每个数据节点会周期性地向名称节点发送“心跳”信息, 报告自己的状态, 没有按时发送心跳信息的数据节点会被标记为“宕机”, 不会再给它分配任何 I/O 请求。

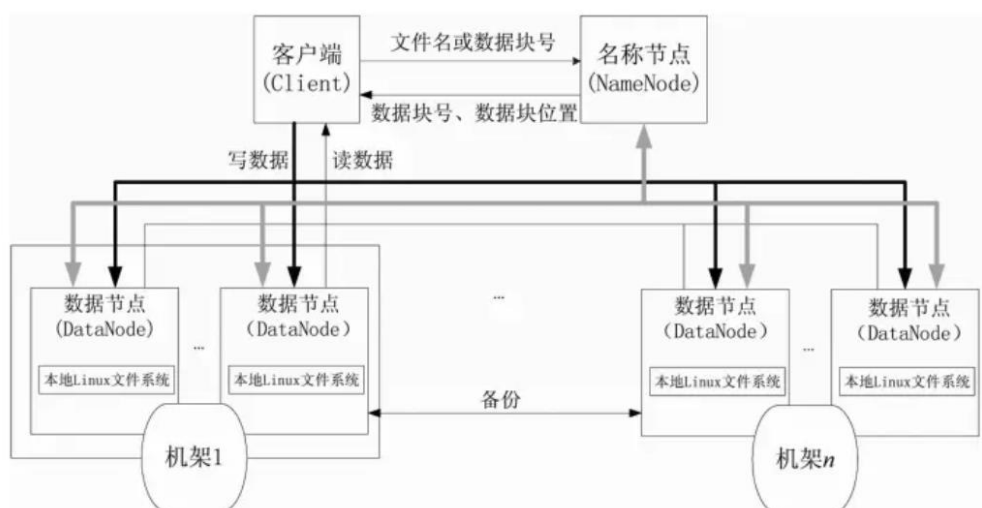


图 6-5 HDFS 的体系结构

用户在使用 HDFS 时, 仍然可以像在普通文件系统中那样, 使用文件名去存储和访问文件。实际上, 在系统内部, 一个文件会被切分成若干个数据块, 这些数据块被分布存储到若干个数据节点上。当客户端需要访问一个文件时, 首先把文件名发送给名称节点, 名称节点根据文件名找到对应的数据块 (一个文件可能包括多个数据块), 再根据每个数据块信息找到实际存储各个数据块的数据节点的位置, 并把数据节点位置发送给客户端, 最后客户端直接访问这些数据节点获取数据。在整个访问过程中, 名称节点并不参与数据的传输。这种设计方式, 使得一个文件的数据能够在不同的数据节点上实现并发访问, 大大提高数据访问速度。

## 6.5 NoSQL 数据库

NoSQL 数据库虽然数量众多, 但是归结起来, 典型的 NoSQL 数据库通常包括键值数据库、列族数据库、文档数据库和图数据库, 如图 6-6 所示。

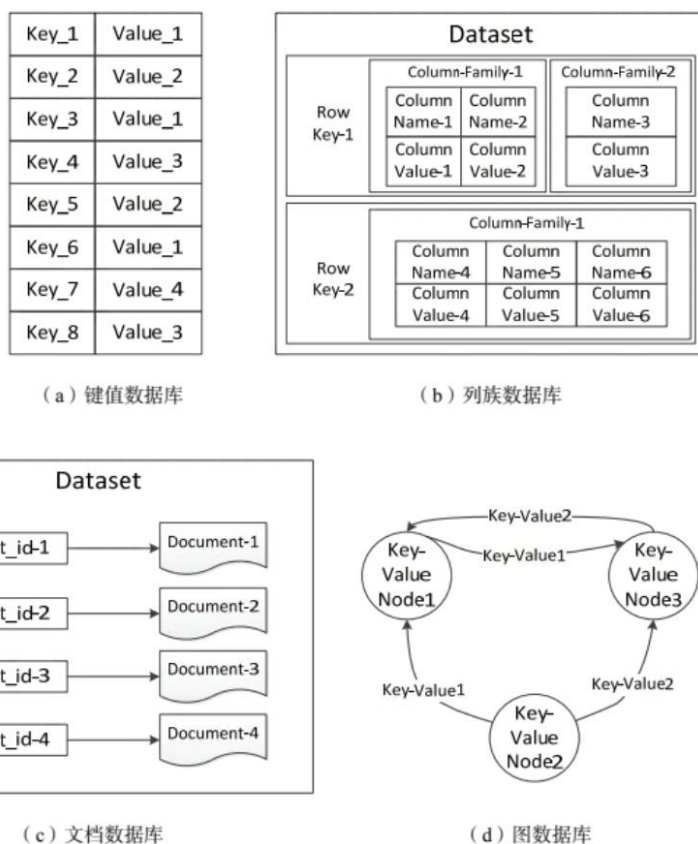


图 6-6 不同类型的 NoSQL 数据库

### 6.5.1 键值数据库

键值数据库 (Key-Value Database) 会使用一个哈希表, 这个表中有一个特定的 Key 和一个指针指向特定的 Value。Key 可以用来定位 Value, 即存储和检索具体的 Value。Value 对数据库而言是透明不可见的, 不能对 Value 进行索引和查询, 只能通过 Key 进行查询。Value 可以用来存储任意类型的数据, 包括整型、字符型等。在存在大量写操作的情况下, 键值数据库可以比关系数据库取得更好的性能。因为关系数据库需要建立索引来加速查询, 当存在大量写操作时, 索引会频繁更新, 由此会产生高昂的索引维护代价。关系数据库通常很难横向扩展, 但是键值数据库天生具有良好的伸缩性, 理论上几乎可以实现数据量的无限扩容。键值数据库可以进一步划分为内存键值数据库和持久化 (Persistent) 键值数据库。内存键值数据库把数据保存在内存中, 如 Memcached 和 Redis; 持久化键值数据库把数据保存在磁盘中, 如 BerkeleyDB、Voldemort 和 Riak。

当然, 键值数据库也有自身的局限性, 条件查询就是键值数据库的弱项。因此, 如果只对部分值进行查询或更新, 效率就会比较低。在使用键值数据库时, 应该尽量避免多表关联查询, 可以采用双向冗余存储关系来代替表关联, 把操作分解成单表操作。此外, 键值数据库在发生故障时不支持回滚操作, 因此无法支持事务。键值数据库的相关产品、数据模型、典型应用、优点、

缺点和使用者如表 6-2 所示。

表 6-2 键值数据库

项目	描述
相关产品	Redis、Riak、SimpleDB、Chordless、Scalaris、Memcached
数据模型	键值对
典型应用	内容缓存, 如会话、配置文件、参数、购物车等
优点	扩展性好、灵活性好, 进行大量写操作时性能高
缺点	无法存储结构化信息、条件查询效率较低
使用者	百度云数据库 (Redis)、GitHub (Riak)、BestBuy (Riak)、StackOverFlow (Redis)、Instagram (Redis)

Redis 是一款具有代表性的键值数据库产品, 可以对关系数据库起到很好的补充作用, 目前正在被越来越多的互联网公司采用。Redis 提供了 Java、C/C++、C#、PHP、JavaScript、Perl、Object-C、Python、Ruby、Erlang 客户端, 使用很方便。Redis 支持存储的值 (Value) 类型包括 string (字符串)、list (链表)、set (集合) 和 zset (有序集合)。这些数据类型都支持 push/pop、add/remove 以及取交集、并集和差集等丰富的操作, 而且这些操作都是原子性的。在此基础上, Redis 支持各种不同方式的排序。为了保证效率, Redis 中的数据都是缓存在内存中的, 它会周期性地更新的数据写入磁盘, 或者把修改操作写入追加的记录文件。

## 6.5.2 列族数据库

列族数据库一般采用列族数据模型, 数据库由多个行构成, 每行数据包含多个列族, 不同的行可以具有不同数量的列族, 属于同一列族的数据会被存放在一起。每行数据通过行键进行定位, 与这个行键对应的是一个列族。从这个角度来说, 列族数据库也可以被视为一个键值数据库。列族可以被配置成支持不同类型的访问模式, 一个列族也可以被设置成放入内存, 以消耗内存为代价来换取更好的响应性能。列族数据库的相关产品、数据模型、典型应用、优点、缺点和使用者如表 6-3 所示。

表 6-3 列族数据库

项目	描述
相关产品	Bigtable、HBase、Cassandra、HadoopDB、GreenPlum、PNUTS
数据模型	列族
典型应用	分布式数据存储与管理
优点	查找速度快、可扩展性强、容易进行分布式扩展、复杂性低
缺点	功能较少, 大多不支持强事务一致性
使用者	Ebay (Cassandra)、Instagram (Cassandra)、Yahoo! (HBase)

HBase 就是一款具有代表性的列族数据库产品。HBase 具有高可扩展性, 可以支持超大规模数据存储, 它可以通过横向扩展的方式, 利用廉价计算机集群处理超过 10 亿行数据和数百万列

元素组成的数据表。

### 6.5.3 文档数据库

在文档数据库中，文档是数据库的最小单位。虽然每一种文档数据库的部署有所不同，但是大多文档以某种标准化格式封装并对数据进行加密，同时用多种格式进行解码，包括 XML、YAML、JSON 和 BSON 等，或者也可以使用二进制格式进行解码（如 PDF、微软 Office 文档等）。文档数据库通过键来定位一个文档，因此可以看成键值数据库的一个衍生品，而且前者比后者具有更高的查询效率。对于那些可以把输入数据表示成文档的应用而言，文档数据库是非常合适的。一个文档可以包含非常复杂的数据结构，如嵌套对象，并且不需要采用特定的数据模式，每个文档可能具有完全不同的结构。文档数据库既可以根据键（Key）来构建索引，也可以基于文档内容来构建索引。基于文档内容的索引和查询能力，是文档数据库不同于键值数据库的地方。因为在键值数据库中，值（Value）对数据库是透明不可见的，不能根据值来构建索引。文档数据库主要用于存储并检索文档数据，当文档数据库需要考虑很多关系和标准化约束以及需要事务支持时，传统的关系数据库是更好的选择。文档数据库的相关产品、数据模型、典型应用、优点、缺点和使用者如表 6-4 所示。

表 6-4 文档数据库

项目	描述
相关产品	CouchDB、MongoDB、Terrastore、ThruDB、RavenDB、SisoDB、RaptorDB、CloudKit、Persevere、Jackrabbit
数据模型	版本化的文档
典型应用	存储、索引并管理面向文档的数据或者类似的半结构化数据
优点	性能好、灵活性高、复杂性低、数据结构灵活
缺点	缺乏统一的查询语法
使用者	百度云数据库( MongoDB )、SAP( MongoDB )、Codecademy( MongoDB )、Foursquare( MongoDB )、NBC News ( RavenDB )

MongoDB 就是一款具有代表性的文档数据库产品，它是一个基于分布式文件存储的文档数据库，介于关系数据库和非关系数据库之间，是非关系数据库当中功能最丰富、最像关系数据库的一种 NoSQL 数据库。MongoDB 支持的数据结构非常松散，是类似 JSON 的 BSON 格式，因此可以存储比较复杂的数据类型。MongoDB 最大的特点是支持的查询语言非常强大，语法有点类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且支持对数据建立索引。

### 6.5.4 图数据库

图数据库以图论为基础，一个图是一个数学概念，用来表示一个对象集合，包括顶点以及连接顶点的边。图数据库使用图作为数据模型来存储数据，完全不同于键值、列族和文档数据模型，

可以高效地存储不同顶点之间的关系。图数据库专门用于处理具有高度相互关联关系的数据，可以高效地处理实体之间的关系，比较适合于社交网络、模式识别、依赖分析、推荐系统以及路径寻找等问题。有些图数据库（如 Neo4J），完全兼容 ACID。但是，图数据库除了在处理图和关系这些应用领域具有很好的性能以外，在其他领域，其性能不如其他 NoSQL 数据库。图数据库的相关产品、数据模型、典型应用、优点、缺点和使用者如表 6-5 所示。

表 6-5 图数据库

项目	描述
相关产品	Neo4j、OrientDB、InfoGrid、Infinite Graph、GraphDB
数据模型	图结构
典型应用	可用于大量复杂、互连接、低结构化的图结构场合，如社交网络、推荐系统等
优点	灵活性高、支持复杂的图算法、可用于构建复杂的关系图谱
缺点	复杂性高、只能支持一定的数据规模
使用者	Adobe (Neo4j)、Cisco (Neo4j)、T-Mobile (Neo4j)

Neo4j 是一个具有代表性的图数据库产品。它是一个嵌入式、基于磁盘的、支持完整事务的 Java 持久化引擎，它在图（网络）中而不是表中存储数据。Neo4j 提供了大规模可扩展性，在一台机器上可以处理数十亿节点/关系/属性的图，可以扩展到多台机器并行运行。相对于关系数据库来说，图数据库善于处理大量复杂、互连接、低结构化的数据，这些数据变化迅速，需要频繁地查询。在关系数据库中，这些查询会导致大量的表连接，因此会产生性能上的问题。Neo4j 重点解决了传统关系数据库在处理涉及大量连接操作的查询时出现的性能衰退问题。通过围绕图进行数据建模，Neo4j 会以相同的速度遍历节点与边，其遍历速度与构成图的数据量没有任何关系。此外，Neo4j 还提供了非常快的图算法、推荐系统和 OLAP 风格的分析，而这一切在目前的关系数据库系统中都是无法实现的。

## 6.6 云数据库

本节介绍云数据库的概念和特性、云数据库与其他数据库的关系，以及代表性的云数据库产品。

### 6.6.1 云数据库的概念

云数据库是部署在云计算环境中的虚拟化数据库。云数据库是在云计算的大背景下发展起来的一种新兴的共享基础架构的数据库，它极大地增强了数据库的存储能力，避免了人员、硬件、软件的重复配置，让软、硬件升级变得更加容易，同时虚拟化了许多后端功能。云数据库具有高可扩展性、高可用性、采用多租形式和支持资源有效分发等特点。

在云数据库中，所有数据库功能都是在“云端”提供的，客户端可以通过网络远程使用云数

数据库提供的服务，如图 6-7 所示。客户端不需要了解云数据库的底层细节，所有的底层硬件都已经被虚拟化，对客户端而言是透明的。客户端就像在使用一个运行在单一服务器上的数据库一样，非常方便、容易，同时可以获得理论上近乎无限的存储和处理能力。

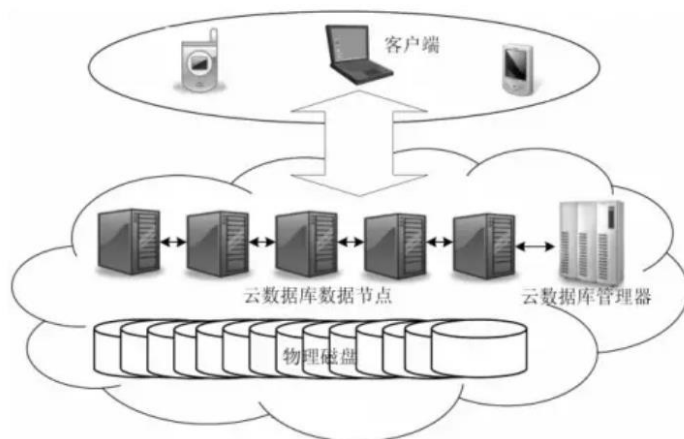


图 6-7 云数据库示意图

需要指出的是，有人认为数据库属于应用基础设施（即中间件），因此把云数据库列入 PaaS 的范畴，也有人认为数据库本身也是一种应用软件，因此把云数据库划入 SaaS。对于这个问题，本书把云数据库划入 SaaS，但同时认为，云数据库到底应该被划入 PaaS 还是 SaaS，这并不是最重要的。实际上，云计算 IaaS、PaaS 和 SaaS 这 3 个层次之间的界限有些时候也不是非常清晰。对于云数据库而言，最重要的是它允许用户以服务的方式通过网络获得云端的数据。

## 6.6.2 云数据库的特性

云数据库具有以下特性。

(1) 动态可扩展。理论上，云数据库具有无限可扩展性，可以满足不断增加的数据存储需求。在面对不断变化的条件时，云数据库可以表现出很好的弹性。例如，对于一个从事产品零售的电子商务公司，会存在季节性或突发性的产品需求变化，或者对于类似 Animoto 的网络社区站点，可能会经历一个指数级的用户增长阶段。这时，它们就可以分配额外的数据库存储资源来处理增加的需求，这个过程只需要几分钟。一旦需求过去以后，它们就可以立即释放这些资源。

(2) 高可用性。云数据库不存在单点失效问题。如果一个节点失效了，剩余的节点就会接管未完成的事务。而且，在云数据库中，数据通常是冗余存储的，在地理上也是分散的。诸如 Google、Amazon 和 IBM 等大型云计算供应商，具有分布在世界范围内的数据中心，通过在不同地理区内进行数据复制，可以提供高水平的容错能力。例如，Amazon SimpleDB 会在不同的区域内进行数据复制，这样，即使某个区域内的云设施失效，也可以保证数据继续可用。

(3) 较低的使用代价。云数据库厂商通常采用多租户（Multi-tenancy）的形式，同时为多个用户提供服务，这种共享资源的形式对于用户而言可以节省开销，而且用户采用“按需付费”的

方式使用云计算环境中的各种软、硬件资源，不会产生不必要的资源浪费。另外，云数据库底层存储通常采用大量廉价的商业服务器，这大大降低了开销。腾讯云数据库官方公布的资料显示，当实现类似的数据库性能时，如果采用自己投资自建 MySQL 的方式，则成本为每台服务器每天 50.6 元，实现双机容灾需要 2 台服务器，即成本为 101.2 元，平均存储成本是每吉字节每天 0.25 元，平均 1 元可获得的每秒查询率（Query Per Second，QPS）为 24 次/秒；而如果采用腾讯云数据库产品，企业不需要投入任何初期建设成本，成本仅为 72 元/天，平均存储成本为每吉字节每天 0.18 元，平均 1 元可获得的 QPS 为 83 次/秒，相对于自建，云数据库平均 1 元获得的 QPS 提高为原来的 346%，具有极高的性价比。

（4）易用性。使用云数据库的用户不必控制运行原始数据库的机器，也不必了解它身在何处。用户只需要一个有效的连接字符串（URL）就可以开始使用云数据库，而且就像使用本地数据库一样。许多基于 MySQL 的云数据库产品（如腾讯云数据库、阿里云 RDS 等），完全兼容 MySQL 协议，用户可通过基于 MySQL 协议的客户端或者 API 访问实例。用户可无缝地将原有 MySQL 应用迁移到云存储平台，无需进行任何代码改造。

（5）高性能。云数据库采用大型分布式存储服务集群，支撑海量数据访问，多机房自动冗余备份，自动读写分离。

（6）免维护。用户不需要关注后端机器及数据库的稳定性、网络问题、机房灾难、单库压力等各种风险，云数据库服务商提供“7×24h”的专业服务，扩容和迁移对用户透明且不影响服务，并且可以提供全方位、全天候立体式监控，用户无需半夜去处理数据库故障。

（7）安全。云数据库提供数据隔离，不同应用的数据会存在于不同的数据库中而不会相互影响；提供安全性检查，可以及时发现并拒绝恶意攻击性访问；提供数据多点备份，确保不会发生数据丢失。

### 6.6.3 云数据库与其他数据库的关系

关系数据库采用关系数据模型，NoSQL 数据库采用非关系数据模型，二者属于不同的数据库技术。从数据模型的角度来说，云数据库并非一种全新的数据库技术，而是以服务的方式提供数据库功能的技术。云数据库并没有专属于自己的数据模型，云数据库所采用的数据模型可以是关系数据库所使用的关系模型（如微软的 SQL Azure 云数据库、阿里云 RDS 都采用了关系模型），也可以是 NoSQL 数据库所使用的非关系模型（如 Amazon Dynamo 等云数据库采用的是“键值”存储）。同一个公司也可能提供采用不同数据模型的多种云数据库服务，例如百度云数据库提供了 3 种数据库服务，即分布式关系数据库服务（基于关系数据库 MySQL）、分布式非关系数据库服务（基于文档数据库 MongoDB）、键值型非关系数据库服务（基于键值数据库 Redis）。实际上，许多公司在开发云数据库时，后端数据库都是直接使用现有的各种关系数据库或 NoSQL 数据库产品。比如，腾讯云数据库采用 MySQL 作为后端数据库，微软的 SQL Azure 云数据库采用 SQL Server 作为后端数据库。从市场的整体应用情况来看，由于 NoSQL 应用对开发者要求较高，而 MySQL 拥有成熟的中间件、运维工具，已经形成一个良性的生态系统等特性，因此从现阶段来

看，云数据库的后端数据库以 MySQL 为主、NoSQL 为辅。

在云数据库这种 IT 服务模式出现之前，企业要使用数据库，就需要自建关系数据库或 NoSQL 数据库，它们被称为“自建数据库”。云数据库与这些“自建数据库”最本质的区别如下。云数据库是部署在云端的数据库，采用 SaaS 模式，用户可以通过网络租赁使用数据库服务，在有网络的地方都可以使用，不需要前期投入和后期维护，使用价格比较低廉。云数据库对用户而言是完全透明的，用户根本不知道自己的数据被保存在哪里。云数据库通常采用多租户模式，即多个租户共用一个实例，租户的数据既有隔离又有共享，从而解决了数据存储的问题，同时降低了用户使用数据库的成本。而自建的关系数据库和 NoSQL 数据库本身都没有采用 SaaS 模式，需要用户自己搭建 IT 基础设施和配置数据库，成本相对而言比较昂贵，而且需要自己进行机房维护和数据库故障处理。

#### 6.6.4 代表性云数据库产品

云数据库供应商主要分为三类。

- (1) 传统的数据库厂商，如 Teradata、Oracle、IBM DB2 和 Microsoft SQL Server 等。
- (2) 涉足数据库市场的云数据库厂商，如 Amazon、Google、Yahoo!、阿里、百度、腾讯等。
- (3) 新兴厂商，如 Vertica、LongJump 和 EnterpriseDB 等。

市场上常见的云数据库产品如表 6-6 所示。

表 6-6 市场上常见的云数据库产品

企业	产品
Amazon	DynamoDB、SimpleDB、RDS
Google	Google Cloud SQL
Microsoft	Microsoft SQL Azure
Oracle	Oracle Cloud
Yahoo!	PNUTS
Vertica	Analytic Database for the Cloud
EnterpriseDB	Postgres Plus in the Cloud
阿里巴巴	阿里云 RDS
百度	百度云数据库
腾讯	腾讯云数据库

## 6.7 分布式数据库 HBase

HBase 是 Google Bigtable 的开源实现，因此，本节首先对 Bigtable 作简要介绍，然后介绍 HBase，最后给出 HBase 的数据模型和系统架构。

### 6.7.1 从 Bigtable 说起

Bigtable 是一个分布式存储系统，利用谷歌提出的 MapReduce 分布式并行计算模型来处理海量数据，使用谷歌分布式文件系统 GFS 作为底层数据存储系统，并采用 Chubby 提供协同服务管理，可以扩展到 PB 级别的数据和上千台机器，具备广泛应用性、可扩展性、高性能和高可用性等特点。从 2005 年 4 月开始，Bigtable 已经在谷歌的实际生产系统中使用，谷歌的许多项目都存储在 Bigtable 中，包括搜索、地图、财经、打印、社交网站 Orkut、视频共享网站 YouTube 和博客网站 Blogger 等。这些应用无论在数据量方面（从 URL 到网页到卫星图像），还是在延迟需求方面（从后端批量处理到实时数据服务），都对 Bigtable 提出了与传统存储系统截然不同的需求。尽管这些应用的需求大不相同，但是 Bigtable 依然能够为所有谷歌产品提供灵活的、高性能的解决方案。当用户的资源需求随着时间变化时，只需要简单地往系统中添加机器，就可以实现服务器集群的扩展。

总的来说，Bigtable 具备以下特性：支持大规模海量数据、分布式并发数据处理效率极高、易于扩展且支持动态伸缩、适用于廉价设备、适合读操作不适合写操作。

### 6.7.2 HBase 简介

HBase 是一个高可靠、高性能、面向列、可伸缩的分布式数据库，是谷歌 Bigtable 的开源实现，主要用来存储非结构化和半结构化的数据。HBase 的目标是处理非常庞大的表，可以通过横向扩展的方式，利用廉价计算机集群处理由超过 10 亿行数据和数百万列元素组成的数据表。

图 6-8 描述了 Hadoop 生态系统中 HBase 与其他部分的关系。HBase 利用 Hadoop MapReduce 来处理 HBase 中的海量数据，实现高性能计算；利用 ZooKeeper 作为协同服务，实现稳定服务和失败恢复；使用 HDFS 作为高可靠的底层数据存储系统，利用廉价集群提供海量数据存储能力。当然，HBase 也可以直接使用本地文件系统而不用 HDFS 作为底层数据存储系统。不过，为了提高数据可靠性和系统的健壮性，发挥 HBase 处理大数据量等功能，一般都使用 HDFS 作为 HBase 的底层数据存储系统。此外，为了方便在 HBase 上进行数据处理，Sqoop 为 HBase 提供了高效、便捷的关系数据库管理系统（Relational Database Management System, RDBMS）数据导入功能，Pig 和 Hive 为 HBase 提供了高层语言支持。

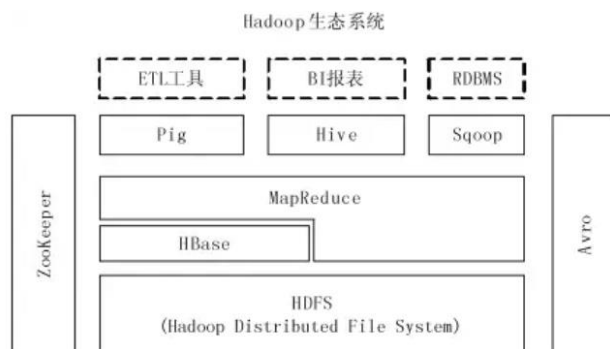


图 6-8 Hadoop 生态系统中 HBase 与其他部分的关系

### 6.7.3 HBase 数据模型

HBase 实际上就是一个稀疏、多维、持久化存储的映射表，它采用行键（Row Key）、列族（Column Family）、列限定符（Column Qualifier）和时间戳（Timestamp）进行索引，每个值都是未经解释的字节数组 `byte[]`。下面具体介绍 HBase 数据模型的相关概念。

（1）表。HBase 采用表来组织数据，表由行和列组成，列划分为若干个列族。

（2）行键。每个 HBase 表都由若干行组成，每个行由行键来标识。访问表中的行只有 3 种方式：通过单个行键访问、通过一个行键的区间来访问、全表扫描。行键可以是任意字符串（最大长度是 64 KB，实际应用中长度一般为 10~100Byte），在 HBase 内部，行键保存为字节数组。存储时，数据按照行键的字典序存储。在设计行键时，要充分考虑这个特性，将经常一起读取的行存储在一起。

（3）列族。一个 HBase 表被分组成许多“列族”的集合，它是基本的访问控制单元。列族需要在表创建时就定义好，数量不能太多（HBase 的一些缺陷使得列族的数量只限于几十个），而且不能频繁修改。存储在一个列族当中的所有数据，通常都属于同一种数据类型，这意味着具有更高的压缩率。表中的每个列都归属于某个列族，数据可以被存放到列族的某个列下面，但是在把数据存放到这个列族的某个列下面之前，必须首先创建这个列族。在创建完列族以后，就可以使用同一个列族当中的列。列名都以列族作为前缀。例如，`courses:history` 和 `courses:math` 这两个列都属于 `courses` 这个列族。在 HBase 中，访问控制、磁盘和内存的使用统计都是在列族层面进行的。实际应用中，我们可以借助列族上的控制权限帮助实现特定的目的。比如，我们可以允许一些应用能够向表中添加新的数据，而另一些应用只允许浏览数据。HBase 列族还可以被配置成支持不同类型的访问模式。比如，一个列族也可以被设置成放入内存，以消耗内存为代价，换取更好的响应性能。

（4）列限定符。列族里的数据通过列限定符（或列）来定位。列限定符不用事先定义，也不需要不同行之间保持一致。列限定符没有数据类型，总被视为字节数组 `byte[]`。

（5）单元格。在 HBase 表中，通过行键、列族和列限定符确定一个“单元格”（Cell）。单元格中存储的数据没有数据类型，总被视为字节数组 `byte[]`。每个单元格中可以保存一个数据的多个版本，每个版本对应一个不同的时间戳。HBase 是以单元格为单位来写入和读取数据的，这一点和关系数据库具有很大的区别。在关系数据库中，数据以行为单位，一行一行写入，一行一行读取。而在 HBase 数据库中，写入数据时，是一个单元格一个单元格地写入，读取数据时，也是一个单元格一个单元格地读取。

（6）时间戳。每个单元格都保存着同一份数据的多个版本，这些版本采用时间戳进行索引。每次对一个单元格执行操作（新建、修改、删除）时，HBase 都会隐式地自动生成并存储一个时间戳。时间戳一般是 64 位整型数据，可以由用户自己赋值（自己生成唯一时间戳可以避免应用程序中出现数据版本冲突），也可以由 HBase 在数据写入时自动赋值。一个单元格的不同版本根据时间戳降序存储，这样，最新的版本可以被最先读取。

下面以一个实例来阐释 HBase 的数据模型。图 6-9 是一张用来存储学生信息的 HBase 表，学

号作为行键来唯一标识每个学生，表中设计了列族 Info 来保存学生相关信息，列族 Info 中包含 3 个列——name、major 和 email，分别用来保存学生的姓名、专业和电子邮件信息。学号为“201505003”的学生存在两个版本的电子邮件地址，时间戳分别为  $ts1=1174184619081$  和  $ts2=1174184620720$ ，时间戳较大的版本的数据是最新的数据。再来看一下 HBase 和关系数据库在读写数据方面的差别。在关系数据库中，数据是逐行写入的，比如，先写入第一行数据（“201505001”，“Luo Min”，“Math”，“luo@qq.com”），再写入第 2 行数据，依此类推。而在 HBase 数据库中，数据是逐个单元格写入的，比如，对于行键“201505001”而言，先写入第 1 个单元格的值“Luo Min”，再写入第 2 个单元格的值“Math”，然后写入第 3 个单元格的值“luo@qq.com”。在读取数据时，也是类似的。



图 6-9 一张用来存储学生信息的 HBase 表

## 6.7.4 HBase 系统架构

HBase 的系统架构如图 6-10 所示，包括客户端、ZooKeeper 服务器、Master 主服务器、Region 服务器。需要说明的是，HBase 一般采用 HDFS 作为底层数据存储系统，因此图 6-10 中加入了 HDFS 和 Hadoop。

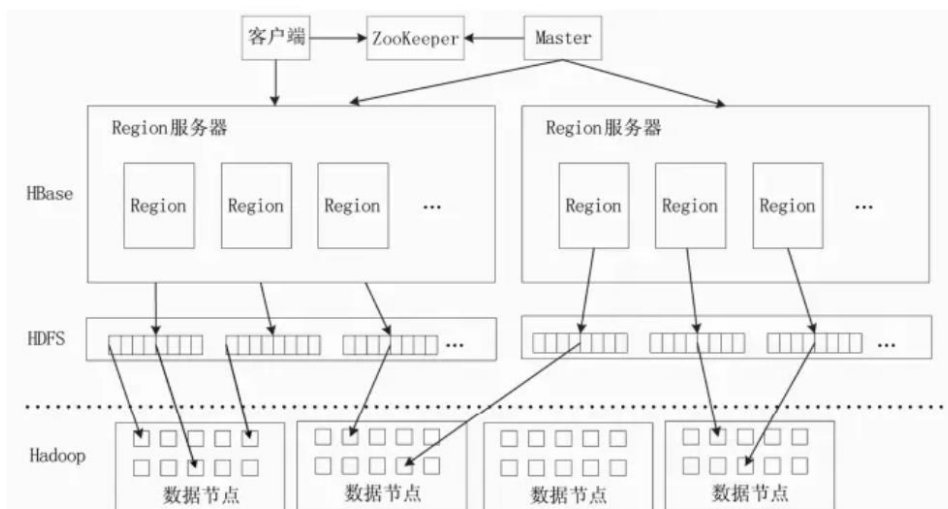


图 6-10 HBase 的系统架构

在一个 HBase 中, 存储了许多表。对于每个 HBase 表而言, 表中的行根据行键的值的字典序进行维护, 表中包含的行的数量可能非常庞大, 无法存储在一台机器上, 需要分布存储到多台机器上。因此, 需要根据行键的值对表中的行进行分区, 每个行区间构成一个分区, 被称为“Region”, 包含了位于某个值域的所有数据, 它是负载均衡和数据分发的基本单位, 这些 Region 会被分发到不同的 Region 服务器上。

客户端包含访问 HBase 的接口, 同时在缓存中维护着已经访问过的 Region 位置信息, 用来加快后续数据访问过程。HBase 客户端使用 HBase 的 RPC (Remote Process Call) 机制与 Master 和 Region 服务器进行通信。其中, 对于管理类操作, 客户端与 Master 进行 RPC; 而对于数据读写类操作, 客户端会与 Region 服务器进行 RPC。

在 HBase 服务器集群中, 包含了一个 Master 和多个 Region 服务器。Master 主要负责表和 Region 的管理工作, Region 服务器负责维护分配给自己的 Region, 并响应用户的读写请求。Master 就是这个 HBase 集群的“总管”, 它必须知道 Region 服务器的状态。ZooKeeper 就可以轻松做到这一点, 每个 Region 服务器都需要到 ZooKeeper 中进行注册, ZooKeeper 会实时监控每个 Region 服务器的状态并通知 Master, 这样, Master 就可以通过 ZooKeeper 随时感知到各个 Region 服务器的工作状态。

## 6.8 Google Spanner

Spanner 是一个可扩展的、全球分布式的数据库, 由 Google 设计、开发和部署。在最高抽象层面, Spanner 就是一个数据库, 把数据分片存储在许多 Paxos 状态机上, 这些机器遍布全球的数据中心。复制技术可以用来服务于全球可用性和地理局部性。客户端会自动在副本之间进行失败恢复。随着数据的变化和服务器的变化, Spanner 会自动把数据进行重新分片, 从而有效应对负载变化和失败。Spanner 被设计成可以扩展到几百万个机器节点, 跨越成百上千个数据中心, 具备几万亿数据库行的规模。应用可以借助于 Spanner 来实现高可用性, 在一个地区的内部或跨越不同的地区复制数据, 保证即使面对大范围的自然灾害时数据依然可用。

作为一个全球分布式数据库, Spanner 提供了很好的特性。第一, 在数据的副本配置方面, 应用可以在一个很细的粒度上进行动态控制。应用可以详细规定, 哪些数据中心包含哪些数据, 数据距离用户有多远 (控制用户读取数据的延迟), 不同数据副本之间的距离有多远 (控制写操作的延迟), 以及需要维护多少个副本 (控制可用性和读操作性能)。数据也可以动态和透明地在数据中心之间移动, 从而平衡不同数据中心内资源的使用。第二, Spanner 提供了读和写操作的外部一致性, 以及在一个时间戳下面的跨越数据库的全球一致性的读操作。这些特性使得 Spanner 可以支持一致的备份、一致的 MapReduce 执行和原子模式变更, 所有都是在全球范围内实现, 即使存在正在处理中的事务也支持。

一个 Spanner 部署称为一个 Universe。Spanner 被组织成许多个 Zone 的集合, 每个 Zone 都大

概像一个 Bigtable 服务器的部署。Zone 是管理部署的基本单元，Zone 的集合也是数据可以被复制到的位置的集合。当新的数据中心加入服务，或者老的数据中心被关闭时，Zone 可以被加入一个运行的系统，或者从中移除。Zone 也是物理隔离的单元，在一个数据中心中，可能有一个或者多个 Zone，例如，属于不同应用的数据可能必须被分区存储到同一个数据中心的不同服务器集合中。

图 6-11 显示了 Spanner 服务器的组织方式。一个 Zone 包括一个 Zonemaster 和一百至几千个 Spanserver。Zonemaster 把数据分配给 Spanserver，Spanserver 把数据提供给客户端。客户端使用每个 Zone 上面的 Locationproxy 来定位可以为自己提供数据的 Spanserver。Universe master 是一个控制台，它显示了关于 Zone 的各种状态信息，可以用于相互之间的调试。Placement driver 会周期性地与 Spanserver 进行交互，来发现那些需要被转移的数据，或者是为了满足新的副本约束条件，或者是为了进行负载均衡。

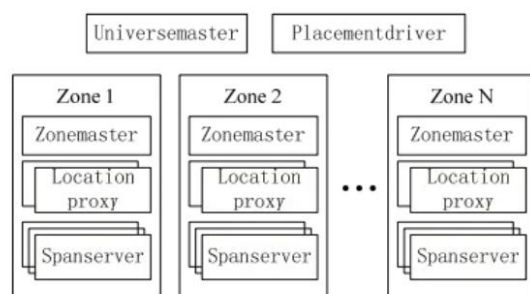


图 6-11 Spanner 服务器的组织方式

## 6.9 本章小结

随着计算机技术的发展，数据存储与管理经历了人工管理、文件系统、数据库系统 3 个发展阶段。在数据库方面，经历了网状数据库、层次数据库、关系数据库、NoSQL 数据库；在文件系统方面，则由单机文件系统发展到了现在的分布式文件系统。数据存储与管理技术的不断发展，使人类能够管理的数据越来越多，效率越来越高，对后续的大数据处理分析环节起到了很好的支撑作用。

本章首先介绍了文件系统、关系数据库、数据仓库、并行数据库等传统的数据存储与管理技术，然后介绍了分布式文件和分布式数据库等大数据时代的数据存储和管理技术。需要说明的是，虽然大数据时代的新技术“大行其道”，但是，一些传统的数据存储与管理技术仍然在发挥着“余热”，不会立即退出大数据的“江湖”。

## 6.10 习题

1. 试述传统的数据存储与管理技术有哪些。

2. 试述关系数据库有哪些特性。
3. 试述数据仓库有哪些特性。
4. 试述 Hadoop 具有哪些特性。
5. 试述 Hadoop 生态系统及其每个部分的具体功能。
6. 试述 HDFS 的设计要实现哪些目标。
7. 试述 HDFS 中的名称节点和数据节点的具体功能。
8. 试述键值数据库、列族数据库、文档数据库和图数据库的适用场合和优缺点。
9. 试述云数据库的概念。
10. 云数据库有哪些特性?
11. 试述云数据库与其他数据库的关系。
12. 举例说明云数据库厂商及其代表性产品。
13. 试述在 Hadoop 体系架构中 HBase 与其他组成部分的相互关系。
14. 请以实例说明 HBase 数据模型。
15. 分别解释 HBase 中行键、列键和时间戳的概念。
16. 试述 HBase 的系统架构及其每个组件的功能。
17. 试述 Spanner 服务器的组织方式。