HZ BOOKS
华章科技

畅销书全新升级，第1版广获好评；资深MySQL专家撰写，全球知名MySQL数据库服务提供商Percona公司CTO作序推荐，国内多位数据库专家联袂推荐

基于MySQL 5.6，结合源代码，从存储引擎内核角度对InnoDB的整体架构、核心实现和工作机制进行深入剖析

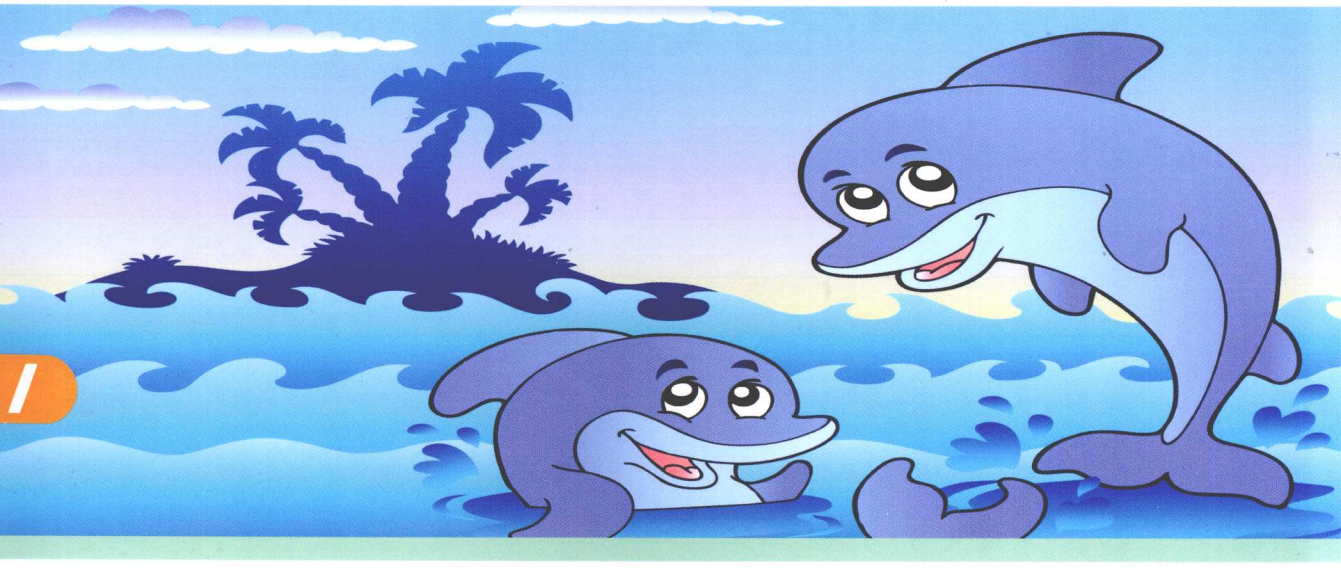
数据库技术丛书

Inside MySQL: InnoDB Storage Engine, Second Edition

MySQL技术内幕

InnoDB存储引擎

第2版



姜承尧◎著

机械工业出版社
China Machine Press

Inside MySQL: InnoDB Storage Engine, Second Edition

MySQL技术内幕

InnoDB存储引擎

第2版

姜承尧◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

MySQL 技术内幕：InnoDB 存储引擎 / 姜承尧著 .—2 版 . —北京：机械工业出版社，2013.6
（数据库技术丛书）

ISBN 978-7-111-42206-8

I. M… II. 姜… III. 关系数据库系统 IV. TP311.138

中国版本图书馆 CIP 数据核字（2013）第 079001 号

本书由国内资深 MySQL 专家亲自执笔，国内外多位数据库专家联袂推荐。作为国内唯一一本关于 InnoDB 的专著，本书的第 1 版广受好评，第 2 版不仅针对最新的 MySQL 5.6 对相关内容进行了全面的补充，还根据广大读者的反馈意见对第 1 版中存在的不足进行了完善，全书大约重写了 50%的内容。本书从源代码的角度深度解析了 InnoDB 的体系结构、实现原理、工作机制，并给出了大量最佳实践，能帮助你系统而深入地掌握 InnoDB，更重要的是，它能为你设计管理高性能、高可用的数据库系统提供绝佳的指导。

全书一共 10 章，首先宏观地介绍了 MySQL 的体系结构和各种常见的存储引擎以及它们之间的比较；接着以 InnoDB 的内部实现为切入点，逐一详细讲解了 InnoDB 存储引擎内部的各个功能模块的实现原理，包括 InnoDB 存储引擎的体系结构、内存中的数据结构、基于 InnoDB 存储引擎的表和页的物理存储、索引与算法、文件、锁、事务、备份与恢复，以及 InnoDB 的性能调优等重要的知识；最后对 InnoDB 存储引擎源代码的编译和调试做了介绍，对大家阅读和理解 InnoDB 的源代码有重要的指导意义。

本书适合所有希望构建和管理高性能、高可用性的 MySQL 数据库系统的开发者和 DBA 阅读。

机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码 100037）
责任编辑：姜 影
北京市荣盛彩色印刷有限公司印刷
2013 年 5 月第 2 版第 1 次印刷
186mm×240mm • 27.25 印张
标准书号：ISBN 978-7-111-42206-8
定 价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

推 荐 序

It's fair to say that MySQL is the most popular open source database. It has a very large installed base and number of users. Let's see what are the reasons MySQL is so popular, where it stands currently, and maybe touch on some of its future (although predicting the future is rarely successful).

Looking at the customer area of MySQL, which includes Facebook, Flickr, Adobe (in Creative Suite 3), Drupal, Digg, LinkedIn, Wikipedia, eBay, YouTube, Google AdSense (source http://mysql.com/customers/ and public resources), it's obvious that MySQL is everywhere. When you log in to your popular forum (powered by Bulleting) or blog (powered by WordPress), most likely it has MySQL as its backend database. Traditionally, two MySQL's characteristics, simplicity of use and performance, were what allowed it to gain such popularity. In addition to that, availability on a very wide range of platforms (including Windows) and built-in replication, which provides an easy scale-out solution for read-only clients, gave more user attractions and production deployments. There is simple evidence of MySQL's simplicity: In 15 minutes or less, you really can get installed, have a working database, and start running queries and store data. From its early stages MySQL had a good interface to most popular languages for Web development - PHP and Perl, and also Java and ODBC connectors.

There are two best known storage engines in MySQL: MyISAM and InnoDB (I don't cover NDB cluster here; it's a totally different story). MyISAM comes as the default storage engine and historically it is the oldest, but InnoDB is ACID compliant and provides transactions, row-level locking, MVCC, automatic recovery and data corruption detection. This makes it the storage engine you want to choose for your application. Also, there is the third-party transaction storage engine PBXT, with characteristics similar to InnoDB, which is included in the MariaDB distribution.

MySQL's simplicity has its own drawback. Just as it is very easy to start working with it, it is very easy to start getting into trouble with it. As soon as your website or forum gets popular, you may figure out that the database is a bottleneck, and that you need special skills and tools to fix it.

The author of this book is a MySQL expert, especially in InnoDB storage engineB. Hence, I highly recommend this book to new users of InnoDB as well as uers who already have well-tuned InnoDB-based applications but need to get internal out of them.

Vadim Tkachenko

全球知名 MySQL 数据库服务提供商 Percona 公司 CTO

知名 MySQL 数据库博客 MySQLPerformanceBlog.com 作者

《高性能 MySQL（第 2 版）》作者之一

前　言

为什么要写这本书

过去这些年我一直在和各种不同的数据库打交道，见证了 MySQL 从一个小型的关系型数据库发展为各大企业的核心数据库系统的过程，并且参与了一些大大小小的项目的开发工作，成功地帮助开发人员构建了可靠的、健壮的应用程序。在这个过程中积累了一些经验，正是这些不断累积的经验赋予了我灵感，于是有了这本书。这本书实际上反映了这些年来我做了哪些事情，其中汇集了很多同行每天可能都会遇到的一些问题，并给出了解决方案。

MySQL 数据库独有的插件式存储引擎架构使其和其他任何数据库都不同。不同的存储引擎有着完全不同的功能，而 InnoDB 存储引擎的存在使得 MySQL 跃入了企业级数据库领域。本书完整地讲解了 InnoDB 存储引擎中最重要的一些内容，即 InnoDB 的体系结构和工作原理，并结合 InnoDB 的源代码讲解了它的内部实现机制。

本书不仅讲述了 InnoDB 存储引擎的诸多功能和特性，还阐述了如何正确地使用这些功能和特性，更重要的是，还尝试了教我们如何 Think Different。Think Different 是 20 世纪 90 年代苹果公司在其旷日持久的宣传活动中提出的一个口号，借此来重振公司的品牌，更重要的是，这个口号改变了人们对技术在日常生活中的作用的看法。需要注意的是，苹果的口号不是 Think Differently，是 Think Different，Different 在这里做名词，意味该思考些什么。

很多 DBA 和开发人员都相信某些“神话”，然而这些“神话”往往都是错误的。无论计算机技术发展的速度变得多快，数据库的使用变得多么简单，任何时候 Why 都比 What 重要。只有真正理解了内部实现原理、体系结构，才能更好地去使用。这正是人类正确思考问题的原则。因此，对于当前出现的技术，尽管学习其应用很重要，但更重要的是，应当正确地理解和使用这些技术。

关于本书，我的头脑里有很多个目标，但最重要的是想告诉大家如下几个简单的观点：

- ❑ 不要相信任何的“神话”，学会自己思考；
- ❑ 不要墨守成规，大部分人都知道的事情可能是错误的；
- ❑ 不要相信网上的传言，去测试，根据自己的实践做出决定；
- ❑ 花时间充分地思考，敢于提出质疑。

当前有关 MySQL 的书籍大部分都集中在教读者如何使用 MySQL，例如 SQL 语句的使用、复制的搭建的、数据的切分等。没错，这对快速掌握和使用 MySQL 数据库非常有好处，但是真正的数据库工作者需要了解的不仅仅是应用，更多的是内部的具体实现。

MySQL 数据库独有的插件式存储引擎使得想要在一本书内完整地讲解各个存储引擎变得十分困难，有的书可能偏重对 MyISAM 的介绍，有的可能偏重对 InnoDB 存储引擎的介绍。对于初级的 DBA 来说，这可能会使他们的理解变得更困难。对于大多数 MySQL DBA 和开发人员来说，他们往往更希望了解作为 MySQL 企业级数据库应用的第一存储引擎的 InnoDB，我想在本书中，他们完全可以找到他们希望了解的内容。

再强调一遍，任何时候 Why 都比 What 重要，本书从源代码的角度对 InnoDB 的存储引擎的整个体系架构的各个组成部分进行了系统的分析和讲解，剖析了 InnoDB 存储引擎的核心实现和工作机制，相信这在其他书中是很难找到的。

第 1 版与第 2 版的区别

本书是第 2 版，在写作中吸收了读者对上一版内容的许多意见和建议，同时对于最新 MySQL 5.6 中许多关于 InnoDB 存储引擎的部分进行了详细的解析与介绍。希望通过这些改进，给读者一个从应用到设计再到实现的完整理解，弥补上一版中深度有余，内容层次不够丰富、分析手法单一等诸多不足。

较第 1 版而言，第 2 版的改动非常大，基本上重写了 50% 的内容。其主要体现在以下几个方面，希望读者能够在阅读中体会到。

- 本书增加了对最新 MySQL 5.6 中的 InnoDB 存储引擎特性的介绍。MySQL 5.6 版本是有史以来最大的一次更新，InnoDB 存储引擎更是添加了许多功能，如多线程清理线程、全文索引、在线索引添加、独立回滚段、非递归死锁检测、新的刷新算法、新的元数据表等。读者通过本书可以知道如何使用这些特性、新特性存在的局限性，并明白新功能与老版本 InnoDB 存储引擎之间实现的区别，从而在实际应用中充分利用这些特性。
- 根据读者的要求对于 InnoDB 存储引擎的 redo 日志和 undo 日志进行了详细的分析。读者应该能更好地理解 InnoDB 存储引擎事务的实现。在 undo 日志分析中，通过 InnoSQL 自带的元数据表，用户终于可对 undo 日志进行统计和分析，极大提高了 DBA 对于 InnoDB 存储引擎内部的认知。

- 对第 6 章进行大幅度的重写，读者可以更好地理解 InnoDB 存储引擎特有的 next-key locking 算法，并且通过分析锁的实现来了解死锁可能产生的情况，以及 InnoDB 存储引擎内部是如何来避免死锁问题的产生的。
- 根据读者的反馈，对 InnoDB 存储引擎的 insert buffer 模块实现进行了更为详细的介绍，读者可以了解其使用方法以及其内部的实现原理。此外还增加了对 insert buffer 的升级版本功能——change buffer 的介绍。

读者对象

本书不是一本面向应用的数据库类书籍，也不是一本参考手册，更不会教你如何在 MySQL 中使用 SQL 语句。**本书面向那些使用 MySQL InnoDB 存储引擎作为数据库后端开发应用程序的开发者和有一定经验的 MySQL DBA**。书中的大部分例子都是用 SQL 语句来展示关键特性的，如果想通过本书来了解如何启动 MySQL、如何配置 Replication 环境，可能并不能如愿。不过，在本书中，你将知道 InnoDB 存储引擎是如何工作的，它的关键特性的功能和作用是什么，以及如何正确配置和使用这些特性。

如果你想更好地使用 InnoDB 存储引擎，如果你想让你的数据库应用获得更好的性能，就请阅读本书。从某种程度上讲，技术经理或总监也要非常了解数据库，要知道数据库对于企业的重要性。如果技术经理或总监想安排员工参加 MySQL 数据库技术方面的培训，完全可以利用本书来“充电”，相信你一定不会失望的。

要想更好地学习本书的内容，要求具备以下条件：

- 掌握 SQL。
- 掌握基本的 MySQL 操作。
- 接触过一些高级语言，如 C、C++、Python 或 Java。
- 对一些基本算法有所了解，因为本书会分析 InnoDB 存储引擎的部分源代码，如果你能看懂这些算法，这会对你的理解非常有帮助。

如何阅读本书

本书一共有 10 章，每一章都像一本“迷你书”，可以单独成册，也就说你完全可以从书中任何一章开始阅读。例如，要了解第 10 章中的 InnoDB 源代码编译和调试的知识，就不必先去阅读第 3 章有关文件的知识。当然，如果你不太确定自己是否已经对本书所涉及的内容

完全掌握了，建议你系统性地阅读本书。

本书不是一本入门书籍，不会一步步引导你去如何操作。倘若你尚不了解 InnoDB 存储引擎，本书对你来说可能就显得沉重一些，建议你先查阅官方的 API 文档，大致掌握 InnoDB 的基础知识，然后再来学习本书，相信你会领略到不同的风景。

为了便于大家阅读，本书在提供源代码下载（下载地址：www.hzbook.com）的同时也将源代码附在了书中，因此占去了一些篇幅，还请大家理解。

勘误和支持

由于作者对 InnoDB 存储引擎的认知水平有限，再加上写作时可能存在疏漏，书中还存在许多需要改进的地方。在此，欢迎读者朋友们指出书中存在的问题，并提出指导性意见，不甚感谢。如果大家有任何与本书相关的内容需要与我探讨，请发邮件到 jiangchengyao@gmail.com，或者通过新浪微博 @insidemysql 与我联系，我会及时给予回复。最后，衷心地希望本书能给大家带来帮助，并祝大家阅读愉快！

致谢

在编写本书的过程中，我得到了很多朋友的热心帮助。首先要感谢 Pecona 公司的 CEO Peter Zaitsev 和 CTO Vadim Tkachenko，通过和他们的不断交流，使我对 InnoDB 存储引擎有了更进一步的了解，同时知道了怎样才能正确地将 InnoDB 存储引擎的补丁应用到生产环境。

其次，要感谢网易公司的各位同事们，能在才华横溢、充满创意的团队中工作我感到非常荣幸和兴奋。也因为在这个开放的工作环境中，我可以不断进行研究和创新。

此外，我还要感谢我的母亲，写本书不是一件容易的事，特别是这本书还想传达一些思想，在这个过程中我遇到了很多的困难，感谢她在这个过程中给予我的支持和鼓励。

最后，一份特别的感谢要送给本书的策划编辑杨福川和姜影，他们使得本书变得生动和更具有灵魂。此外还要感谢出版社的其他默默工作的同事们。

姜承尧

目　录

第 1 章　MySQL 体系结构和存储引擎

MySQL 被设计为一个可移植的数据库，几乎在当前所有系统上都能运行，如 Linux，Solaris、FreeBSD、Mac 和 Windows。尽管各平台在底层（如线程）实现方面都各有不同，但是 MySQL 基本上能保证在各平台上的物理体系结构的一致性。因此，用户应该能很好地理解 MySQL 数据库在所有这些平台上是如何运作的。

1.1　定义数据库和实例

在数据库领域中有两个词很容易混淆，这就是“数据库”（database）和“实例”（instance）。作为常见的数据库术语，这两个词的定义如下。

- **数据库**：物理操作系统文件或其他形式文件类型的集合。在 MySQL 数据库中，数据库文件可以是 frm、MYD、MYI、ibd 结尾的文件。当使用 NDB 引擎时，数据库的文件可能不是操作系统上的文件，而是存放于内存之中的文件，但是定义仍然不变。
- **实例**：MySQL 数据库由后台线程以及一个共享内存区组成。共享内存可以被运行的后台线程所共享。需要牢记的是，数据库实例才是真正用于操作数据库文件的。

这两个词有时可以互换使用，不过两者的概念完全不同。在 MySQL 数据库中，实例与数据库的关通常系是一一对应的，即一个实例对应一个数据库，一个数据库对应一个实例。但是，在集群情况下可能存在一个数据库被多个数据实例使用的情况。

MySQL 被设计为一个单进程多线程架构的数据库，这点与 SQL Server 比较类似，但与 Oracle 多进程的架构有所不同（Oracle 的 Windows 版本也是单进程多线程架构的）。这也就是说，**MySQL 数据库实例在系统上的表现就是一个进程。**

在 Linux 操作系统中通过以下命令启动 MySQL 数据库实例，并通过命令 ps 观察 MySQL 数据库启动后的进程情况：

```
[root@xen-server bin]# ./mysqld_safe&

[root@xen-server bin]# ps -ef | grep mysqld
```

```
root      3441  3258  0 10:23 pts/3    00:00:00 /bin/sh ./mysqld_safe
mysql 3578 3441 0 10:23 pts/3 00:00:00
/usr/local/mysql/libexec/mysqld --basedir=/usr/local/mysql
--datadir=/usr/local/mysql/var --user=mysql
--log-error=/usr/local/mysql/var/xen-server.err
--pid-file=/usr/local/mysql/var/xen-server.pid
--socket=/tmp/mysql.sock --port=3306
root      3616  3258  0 10:27 pts/3    00:00:00 grep mysqld
```

注意进程号为3578的进程，该进程就是MySQL实例。在上述例子中使用了mysqld_safe命令来启动数据库，当然启动MySQL实例的方法还有很多，在各种平台下的方式可能又会有所不同。在这里不一一赘述。

当启动实例时，MySQL数据库会去读取配置文件，根据配置文件的参数来启动数据库实例。这与Oracle的参数文件（spfile）相似，不同的是，Oracle中如果没有参数文件，在启动实例时会提示找不到该参数文件，数据库启动失败。而在MySQL数据库中，可以没有配置文件，在这种情况下，MySQL会按照编译时的默认参数设置启动实例。用以下命令可以查看当MySQL数据库实例启动时，会在哪些位置查找配置文件。

```
[root@xen-server bin]# mysql --help | grep my.cnf
order of preference, my.cnf, $MYSQL_TCP_PORT,
/etc/my.cnf /etc/mysql/my.cnf /usr/local/mysql/etc/my.cnf ~/.my.cnf
```

可以看到，MySQL数据库是按/etc/my.cnf → /etc/mysql/my.cnf → /usr/local/mysql/etc/my.cnf →～/.my.cnf的顺序读取配置文件的。可能有读者会问："如果几个配置文件中都有同一个参数，MySQL数据库以哪个配置文件为准？"答案很简单，MySQL数据库会以读取到的最后一个配置文件中的参数为准。在Linux环境下，配置文件一般放在/etc/my.cnf下。在Windows平台下，配置文件的后缀名可能是.cnf，也可能是.ini。例如在Windows操作系统下运行mysql--help，可以找到如下类似内容：

```
Default options are read from the following files in the given order:
C:\Windows\my.ini C:\Windows\my.cnf C:\my.ini C:\my.cnf C:\Program Files\
MySQL\M
\MySQL Server 5.1\my.cnf
```

配置文件中有一个参数datadir，该参数指定了数据库所在的路径。在Linux操作系统下默认datadir为/usr/local/mysql/data，用户可以修改该参数，当然也可以使用该路径，不过该路径只是一个链接，具体如下：

```
mysql>SHOW VARIABLES LIKE 'datadir'\G;
*************************** 1. row ***************************
Variable_name: datadir
        Value: /usr/local/mysql/data/
1 row in set (0.00 sec)1 row in set (0.00 sec)

mysql>system ls-lh /usr/local/mysql/data
total 32K
drwxr-xr-x  2 root mysql 4.0K Aug  6 16:23 bin
drwxr-xr-x  2 root mysql 4.0K Aug  6 16:23 docs
drwxr-xr-x  3 root mysql 4.0K Aug  6 16:04 include
drwxr-xr-x  3 root mysql 4.0K Aug  6 16:04 lib
drwxr-xr-x  2 root mysql 4.0K Aug  6 16:23 libexec
drwxr-xr-x 10 root mysql 4.0K Aug  6 16:23 mysql-test
drwxr-xr-x  5 root mysql 4.0K Aug  6 16:04 share
drwxr-xr-x  5 root mysql 4.0K Aug  6 16:23 sql-bench
lrwxrwxrwx  1 root mysql   16 Aug  6 16:05 data -> /opt/mysql_data/
```

从上面可以看到，其实 data 目录是一个链接，该链接指向了 /opt/mysql_data 目录。当然，用户必须保证 /opt/mysql_data 的用户和权限，使得只有 mysql 用户和组可以访问（通常 MySQL 数据库的权限为 mysql：mysql）。

1.2 MySQL 体系结构

由于工作的缘故，笔者的大部分时间需要与开发人员进行数据库方面的沟通，并对他们进行培训。不论他们是 DBA，还是开发人员，似乎都对 MySQL 的体系结构了解得不够透彻。很多人喜欢把 MySQL 与他们以前使用的 SQL Server、Oracle、DB2 作比较。因此笔者常常会听到这样的疑问：

- 为什么 MySQL 不支持全文索引？
- MySQL 速度快是因为它不支持事务吗？
- 数据量大于 1000 万时 MySQL 的性能会急剧下降吗？
- ……

对于 MySQL 数据库的疑问有很多很多，在解释这些问题之前，笔者认为不管对于使用哪种数据库的开发人员，了解数据库的体系结构都是最为重要的内容。

在给出体系结构图之前，用户应该理解了前一节提出的两个概念：数据库和数据库实例。很多人会把这两个概念混淆，即 MySQL 是数据库，MySQL 也是数据库实例。

这样来理解 Oracle 和 Microsoft SQL Server 数据库可能是正确的，但是这会给以后理解 MySQL 体系结构中的存储引擎带来问题。从概念上来说，数据库是文件的集合，是依照某种数据模型组织起来并存放于二级存储器中的数据集合；数据库实例是程序，是位于用户与操作系统之间的一层数据管理软件，用户对数据库数据的任何操作，包括数据库定义、数据查询、数据维护、数据库运行控制等都是在数据库实例下进行的，应用程序只有通过数据库实例才能和数据库打交道。

如果这样讲解后读者还是不明白，那这里再换一种更为直白的方式来解释：数据库是由一个个文件组成（一般来说都是二进制的文件）的，要对这些文件执行诸如 SELECT、INSERT、UPDATE 和 DELETE 之类的数据库操作是不能通过简单的操作文件来更改数据库的内容，需要通过数据库实例来完成对数据库的操作。所以，用户把 Oracle、SQL Server、MySQL 简单地理解成数据库可能是有失偏颇的，虽然在实际使用中并不会这么强调两者之间的区别。

好了，在给出上述这些复杂枯燥的定义后，现在可以来看看 MySQL 数据库的体系结构了，其结构如图 1-1 所示（摘自 MySQL 官方手册）。

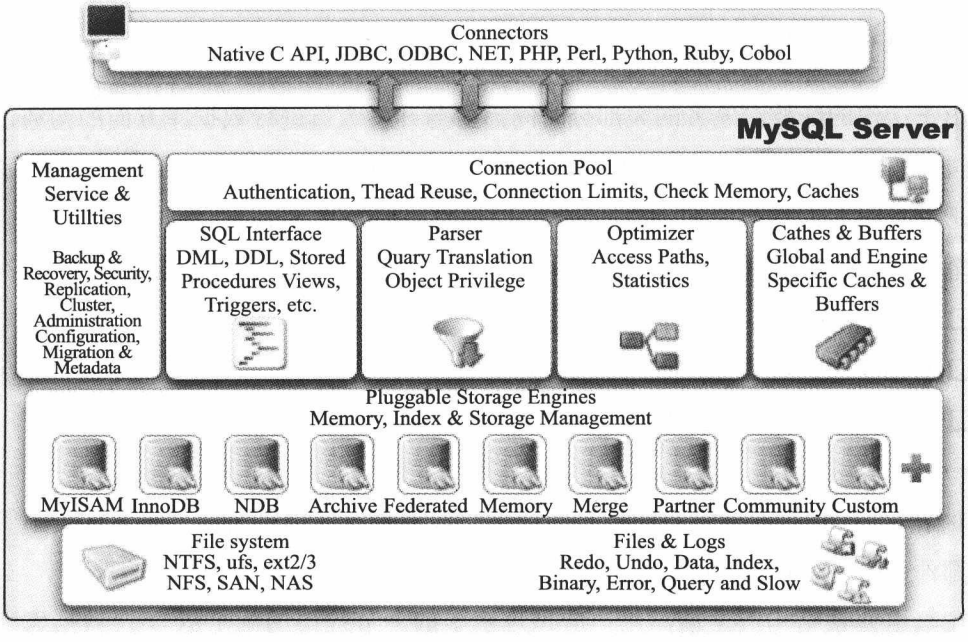


图 1-1 MySQL 体系结构