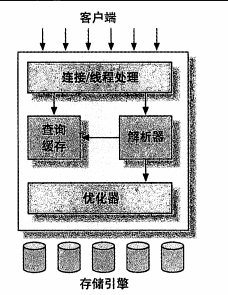
# MySQL架构与历史

MySQL足够灵活，能适应要求高的环境如Web类应用，也可以嵌入到应用程序中，也可以支持数据仓库、内容索引和部署软件、高可用的冗余系统、在线会务处理系统（OLTP）等各种应用类型。

MySQL最重要、最与众不同的特性是它的存储引擎架构，这种架构的设计将查询处理（Query Processing）以及其他系统任务（Server Task）和数据的存储\提取相分离。

## MySQL逻辑架构



最上层服务。比如连接处理、授权认证、安全等。

第二层架构。核心服务，包括查询解析、分析、优化、缓存以及所有的内置函数（例如，日期、时间、数学和加密函数）。所有跨存储的引擎功能也都在这一层实现：存储过程、触发器、视图等。

第三层包含了存储引擎。存储引擎负责MySQL中数据的存储和提取。服务器通过API与存储引擎进行通信。这些接口屏蔽了不同存储引擎之间的差异，使得这些差异对上层的查询过程透明。

### 1.1.1 连接和安全

每个客户端连接购汇在服务器进程中拥有一个线程。连接验证与查询都会在这个线程中进行。

MySQL会解析查询，并创建内部数据结构（解析数），然后对其进行各种优化，包括重写查询、决定表的读取顺序，以及选择合适的索引等。

## 1.2 并发控制

### 1.2.1读写锁

读锁（read lock）= 共享锁（shared lock）

写锁（write lock） = 排他锁（exclusive lock）

读锁是共享的，互相之间不阻塞。

写锁是排他的，一个写锁会阻塞其他写锁和读锁。

### 1.2.2 颗粒度

太频繁的使用锁会影响性能。

表锁（table lock）

是Mysql最基本的锁策略，并且是开销最小的策略。一个用户对表进行写操作（插入、删除、更新等）需要获得写锁，锁住整张表。

服务器会为alter table之类的语句使用表锁，而忽略存储引擎的锁机制。

行级锁（row lock）

行级锁可以最大程度的支持并发处理（同时也带来了最大的锁开销）。

行级锁只在存储引擎层实现，而mysql服务层没有实现。

## 1.3事务

事务就是一组原子性的SQL查询，或者说一个独立的工作单元。如果数据库引擎能够成功的对数据库应用该组查询的全部语句，那么就执行改组查询。

事务内的语句，要么全部执行成功，要么全部执行失败。

ACID

原子性 atomicity

一致性 consistency

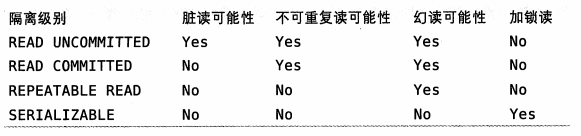
隔离性 isolation

持久性durability

### 1.3.1 隔离级别

sql标准中定义了四种隔离级别，每一种级别都规定了一个事务中所作的修改，那些在事务内和事务间是可见的，那些是不可见的。

较低级别的隔离通常可以执行更高的并发，系统的开销也更低。



### 1.3.2 死锁

### 1.3.4 mysql中的事务

两个存储引擎提供了事务功能InnoDB和NDB Cluster。还有一些第三方的存储引擎也支持事务，XtraDb和PBXT.

自动提交（AUTOCOMMIT）

mysql默认使用自动提交模式。不是显式的开始一个事务，则是每个查询都被当做一个事务执行提交操作。

在事务中混合使用存储引擎

## 1.5 mysql的存储引擎

### 1.5.1 InnoDB存储引擎

InnoDB是mysql的默认事务引擎，也是最重要、使用最广泛的存储引擎。

InnoDB的数据存储在表空间（tablespace）中，表空间是由InnoDB管理的一个黑盒子，由一系列的数据文件组成。在4.1之后的版本，InnoDB可以将每个表的数据和索引存放在单独的文件中。

### 1.5.2 MyISAM存储引擎

在mysql5.1及之前的版本，myisam是默认存储引擎。myisam提供了大量的特性，包括全文索引、压缩、空间函数（GIS）等，但MyISAM不支持事务和行级锁，而且有一个毫无疑问的缺陷就是崩溃后无法安全回复。

对于只读的数据，或者表比较小，可以忍受修复（repair）操作，则依然可以继续使用MyIsam。

### 1.5.5 选择合适的引擎

除非需要用到某些InnoDB不具备的特性，而且没有其他办法可以替代，否则都应该优先选择InnoDB引擎。

除非万不得已，否则建议不要混合使用更多中存储引擎，负责可能带来一些列复杂的问题，以及一些潜在的bug和边界问题。对一致性备份和服务器参数配置都带啦了一些困难。

### 1.5.6 转换表的引擎

方法1：alter

mysql>alter table mytable engine = InnoDB;

执行时间长。

方法2：导入与导出

将原表的数据导出，再重新创建表，导入数据。

方法3：创建于查询

mysql>create table innodb\_table like myisam\_table;

mysql>alter table innodb\_table engine=InnoDB;

mysql>insert into innodb\_table select \* from myisam\_table;

这种方法结合第一种方法的高效和第二种方法的安全。

# MySQL基准测试

简单的说，基准测试是针对系统设计的一种压力测试。通常的目标是为了掌握系统的行为。但也有其他的原因，如重新某个系统状态，或者做新硬件的可靠性测试。

# 服务器性能剖析

最常碰到的三个性能相关的服务请求是：

如何确认服务器是否达到了性能的最佳状态。

找出某条语句为什么执行不够快。

诊断被用户描述成“卡顿”、“堆积”和“卡死”的某些简写行疑难故障。

# 第4章 Schema与数据类型优化

# 第5章 创建高性能的索引

# 第6章 查询性能优化