MySQL 的存储引擎可能是所有关系型数据库产品中最具有特色的了，不仅可以同时使用多种存储引擎，而且每种存储引擎和MySQL之间使用插件方式这种非常松的耦合关系

MySQL 的存储引擎可能是所有关系型数据库产品中最具有特色的了，不仅可以同时使用多种存储引擎，而且每种存储引擎和MySQL之间使用插件方式这种非常松的耦合关系。

由于各存储引擎功能特性差异较大，这篇文章主要是介绍如何来选择合适的存储引擎来应对不同的业务场景。

MyISAM

特性

不支持事务：MyISAM存储引擎不支持事务，所以对事务有要求的业务场景不能使用

表级锁定：其锁定机制是表级索引，这虽然可以让锁定的实现成本很小但是也同时大大降低了其并发性能

读写互相阻塞：不仅会在写入的时候阻塞读取，MyISAM还会在读取的时候阻塞写入，但读本身并不会阻塞另外的读

只会缓存索引：MyISAM可以通过key_buffer缓存以大大提高访问性能减少磁盘IO，但是这个缓存区只会缓存索引，而不会缓存数据

适用场景

不需要事务支持（不支持）

并发相对较低（锁定机制问题）

数据修改相对较少（阻塞问题）

以读为主

数据一致性要求不是非常高

最佳实践

尽量索引（缓存机制）

调整读写优先级，根据实际需求确保重要操作更优先

启用延迟插入改善大批量写入性能

尽量顺序操作让insert数据都写入到尾部，减少阻塞

分解大的操作，降低单个操作的阻塞时间

降低并发数，某些高并发场景通过应用来进行排队机制

对于相对静态的数据，充分利用Query Cache可以极大的提高访问效率

MyISAM的Count只有在全表扫描的时候特别高效，带有其他条件的count都需要进行实际的数据访问

InnoDB

特性

具有较好的事务支持：支持4个事务隔离级别，支持多版本读

行级锁定：通过索引实现，全表扫描仍然会是表锁，注意间隙锁的影响

读写阻塞与事务隔离级别相关

具有非常高效的缓存特性：能缓存索引，也能缓存数据

整个表和主键以Cluster方式存储，组成一颗平衡树

所有Secondary Index都会保存主键信息

适用场景

需要事务支持（具有较好的事务特性）

行级锁定对高并发有很好的适应能力，但需要确保查询是通过索引完成

数据更新较为频繁的场景

数据一致性要求较高

硬件设备内存较大，可以利用InnoDB较好的缓存能力来提高内存利用率，尽可能减少磁盘 IO

最佳实践

主键尽可能小，避免给Secondary index带来过大的空间负担

避免全表扫描，因为会使用表锁

尽可能缓存所有的索引和数据，提高响应速度

在大批量小插入的时候，尽量自己控制事务而不要使用autocommit自动提交

合理设置innodb_flush_log_at_trx_commit参数值，不要过度追求安全性

避免主键更新，因为这会带来大量的数据移动