**一、MySQL架构与历史**

A.并发控制

1.共享锁（shared lock，读锁）：共享的，相互不阻塞的

2.排他锁（exclusive lock，写锁）：排他的，一个写锁会阻塞其他的写锁和读锁

**B.事务**

1.事务ACID

\* 原子性（atomicity）一个事务必须被视为一个不可分割的最小工作单元，整个事务中所有操作要么全部提交成功，要么全部失败回滚，对于一个事务来说，不可能只执行其中的一部分操作

\* 一致性（consistency）数据库总是从一个一致性的状态转换到另外一个一致性的状态

\* 隔离性（isolation）一个事务所做的修改在最终提交以前，对其他事务是不可见的

\* 持久性（durability）一旦事务提交，则其所做的修改就会永久保存到数据库中

2.四种隔离级别

\* READ UNCOMMITTED（未提交读），事务中的修改，即使没有提交，对其他事务也都是可见的，事务可以读取未提交的数据，也被称为脏读（Dirty Read），这个级别会导致很多问题

\* READ COMMITTED（提交读），大多数数据库系统的默认隔离级别，一个事务开始时，只能“看见”已经提交的事务所做的修改，一个事务从开始直到提交之前，所做的任何修改对其他事务都是不可见的，也叫不可重复读（nonrepeatable read），有可能出现幻读（Phantom Read），指的是当某个事务在读取某个范围内的记录时，另外一个事务又在该范围内插入了新的记录，当之前的事务再次读取该范围的记录时，会产生幻行（Phantom Row）

\* REPEATABLE READ（可重复读），通过InnoDB和XtraDB存储引擎，是MySQL的默认事务隔离级别

\* SERIALIZABLE（可串行化）最高级别，通过强制事务串行执行，避免了幻读问题，会在读取的每一行数据上都加锁，可能导致大量的超时和锁争用的问题

3.死锁：指两个或多个事务在同一资源上相互占用，并请求锁定对方占用的资源，从而导致恶性循环的现象

4.事务日志：存储引擎在修改表的数据时只需要修改其内存拷贝，再把该修改行为记录到持久在硬盘上的事务日志中，而不用每次都将修改的数据本身持久到磁盘。事务日志持久以后，内存中被修改的数据在后台可以慢慢地刷回到磁盘，称为预写式日志（Write-Ahead Logging）

**C.多版本并发控制**

1.多版本并发控制（MVCC）是行级锁的一个变种，但是它在很多情况下避免了加锁操作，因此开销更低。虽然实现机制有所不同，但大都实现了非阻塞的读操作，写操作也只锁定必要的行

2.MVCC的实现，是通过保存数据在某个时间点的快照来实现的，有乐观和悲观两种，只在REPEATABLE READ和READ COMMITTED两个隔离级别下工作

**D.MySQL的存储引擎**

1.MySQL的.frm文件保存表的定义，SHOW TABLE STATUS显示表的相关信息

2.除非有非常特别的原因需要使用其他的存储引擎，否则应该优先考虑InnoDB引擎

3.不要轻易相信MyISAM比InnoDB快之类的经验之谈，这个结论并不是绝对的

**二、MySQL基准测试**

A.为什么需要基准测试

1.基准测试可以观察系统在不同压力下的行为，评估系统的容量，掌握哪些是重要的变化，或者观察系统如何处理不同的数据

**B.基准测试的策略**

1.两种主要的策略：

\* 针对整个系统的整体测试（集成式full-stack）

\* 单独测试MySQL（单组件式single-component）

2.测试何种指标：

\* 吞吐量，指单位时间内的事务处理数，常用的测试单位是每秒事务数（TPS），或每分钟事务数（TPM）

\* 响应时间或者延迟，用于测试任务所需的整体时间，根据具体的应用，测试的时间单位可能是微秒、毫秒、秒或者分钟。通常使用百分比响应时间（percentile response time）来替代最大响应时间

\* 并发性，需要关注的是正在工作中的并发操作，或者是同时工作中的线程数或者连接数，在测试期间记录MySQL数据库的Threads\_running状态值

\* 可扩展性，给系统增加一倍的工作，在理想情况下就能获得两倍的效果（即吞吐量增加一倍），对于容量规范非常有用，可以提供其他测试无法提供的信息，来帮助发现应用的瓶颈

**C.基准测试方法**

1.需要避免的一些常见错误：

\* 使用真实数据的子集而不是全集

\* 使用错误的数据分布

\* 使用不真实的分布参数

\* 在多用户场景中，只做单用户测试

\* 在单服务器上测试分布式应用

\* 与真实用户行为不匹配

\* 反复执行同一个查询

\* 没有检查错误

\* 忽略了系统预热（warm up）的过程

\* 使用默认的服务器配置

\* 测试时间太短

2.应该建立将参数和结果文档化的规范，每一轮测试都必须进行详细记录

3.基准测试应该运行足够长的时间，需要在稳定状态下测试并观察

4.在执行基准测试时，需要尽可能多地收集被测试系统的信息

5.自动化基准测试可以防止测试人员偶尔遗漏某些步骤，或者误操作，另外也有助于归档整个测试过程，可以选择shell、php、perl等，要尽可能使所有测试过程都自动化，包括装载数据、系统预热、执行测试、记录结果等