1、和行锁相比，表锁有什么优势

（1）表锁粒度大，占用的内存小，消耗的资源就少

（2）如果业务中经常读写表中较多数据的话，表锁会更快

（2）如果业务中经常使用group by的话，表锁也会更快

注：所以在这种场景下，mysql使用myisam比innodb更优

2、为什么在并发插入量比较大的时候，比较适合用myisam？不会因为表锁频繁冲突而导致吞吐量降低吗？

（1）myisam的索引记录与存储数据分离，是非聚集索引

（2）myisam表，如果数据文件紧密存储，中间没有空闲块，数据总是插入到文件的尾部，就如同追加日志一样，性能很高，此时并发的insert和select是不加锁的（insert插入队列互斥，select共享读锁）

（3）如果数据文件之间有空洞，上述机制就会失效，直到空洞数据被填满，又会启用不加锁机制（删除和修改数据都可能导致空洞）

3、innodb为什么用b+tree作为索引，而不是b-tree（二叉树、平衡二叉树、b-tree、b+tree）

在innodb的存储引擎中，存储是以页为单位的（16k，操作系统中也有页的概念，通常为4k），b+树除叶子节点外，其余节点不存储数据，而b树所有节点都存储数据，所以相同数据量的情况下，同一页能存储的b树的数据量比b+树的要少，这样就会导致b树的深度更大，查找数据就会经历更多次的磁盘io，进而影响查询效率

4、mysql中innodb的锁，锁的是什么？

Record-lock（记录锁，加在索引记录上的锁）、gap-lock（间隙锁，加在索引记录之间的锁）、next-key-lock（记录锁和间隙锁的组合，mysql加锁的基本单位）

5、innodb的缓存池有什么作用？

缓存表数据和表索引数据，把磁盘的数据加载到缓存池中，避免每次查找都进行磁盘io，提高查询效率

6、如何管理和淘汰缓存池，使得性能最大化？

（1）预读：磁盘读取并不是按需读取，而是按页读取，一次最少读取一页的数据（4k），如果未来要读取的数据就在这页中，就能减少磁盘io，提高效率

（2）innodb以什么算法来管理这些缓存页：LRU（最近最少使用算法，一般由链表实现），将放入缓存池的页放入LRU的头部，作为最近访问的元素，从而最晚淘汰；但会出现两个问题：预读失效，缓存池污染

7、预读为什么有效？

局部性原理，一个数据被使用时，它附近的数据也很可能被使用

8、什么是预读失效以及如何优化？

（1）预读失效：由于预读提前把页的数据放到了缓存池中，但是没有被MySQL使用

（2）优化思路：让预读失败的页，停留在缓存池中的时间尽可能的短；让真正被读取的页才移到LRU的头部

（3）具体方法：将LRU分为两个部分（新生代，老年代）；新生代和老年代首尾相连，新生代的尾部连接着老年代的头部；当新页（被预读的页）加入缓存池中，在LRU中只会加入到老年代的头部，如果数据预读成功，则将其再移到新生代的头部

9、什么是缓存池污染及优化解决？

（1）缓存池污染：当一条sql语句需要扫描大量的数据时，可能就会把缓存池中所有的页都刷下去，导致大量的热点数据失效，mysql性能急剧下降

（2）优化方法：加入老年代停留时间窗口的机制，插入到老年代头部的页，即使被立马访也不会移动到新生代头部，只有满足被访问和在老年代停留时间大于设置的时间才会被移动到新生代头部

10、什么是mysql回表查询？

先通过普通索引（非聚集索引）定位主键值，然后再通过主键索引（聚集索引）定位到行记录，扫描两遍索引树。

11、mysql主从复制流程

主：binlog线程——记录下所有改变了数据库数据的语句，放进master上的binlog 中；

从：io线程——在使用start slave 之后，负责从master上拉取 binlog 内容，放进 自己 的relay log中；

从：sql执行线程——执行relay log中的语句；

12、自增主键最大id记录，innodb和myisam是如何存储的?

Innodb:把最大id记录在内存中，重启后会导致最大id重置（重新查询当前记录最大值），8.0后会保存到最大id到表中

myisam:把最大id记录的数据文件中

13、说一说数据库mvcc

Mvcc-多版本并发控制，一般是为了实现对数据库的并发访问。在mysql的inndo引擎中就是指读已提交和和重复读这两种隔离级别下的事务对于查询操作会访问版本链中的记录的过程。它采用了乐观锁的方式，在每行的记录后保存在两个隐藏列来实现trx\_id：事务id，roll\_pointer：指向这条聚簇索引记录的上一个版本的位置，通过它来获取上一个版本的记录信息。

<https://baijiahao.baidu.com/s?id=1629409989970483292&wfr=spider&for=pc>

14、为什么innodb一定会生成主键？

因为innodb的数据结构是通过聚簇索引组织起来的，如果没有主键的话，通过其它索引回表的时候没法查找到对应的数据行。

15、mysql中undolog、redolog、binlog

Undolog：也就是我们常说的回滚日志文件 主要用于事务中执行失败，进行回滚，以及MVCC中对于数据历史版本的查看。

Redolog：是重做日志文件，是记录数据修改之后的值，用于持久化到磁盘中。redo log包括两部分：一是内存中的日志缓冲(redo log buffer)，该部分日志是易失性的；二是磁盘上的重做日志文件(redo log file)，该部分日志是持久的。

Binlog：由Mysql的Server层实现,是逻辑日志,记录的是sql语句的原始逻辑

16、innodb中主键索引和非主键索引是如何进行数据存储的

主键索引：叶子节点存储的是整行数据，也叫聚簇索引

非主键索引：叶子节点存储的是主键的值，也叫二级索引

17、主键采用自增id还是uuid好

因为在InnoDB存储引擎中,主键索引是作为聚簇索引存在的,也就是说,主键索引的B+树叶子节 点上存储了主键索引以及全部的数据(按照顺序),如果主键索引是自增ID,那么只需要不断向后 排列即可,如果是UUID,由于到来的ID与原来的大小不确定,会造成非常多的数据插入,数据移动, 然后导致产生很多的内存碎片,进而造成插入性能的下降.

总之,在数据量大一些的情况下,用自增主键性能会好一些.

18、mysql三范式

第一范式: 每个列都不可以再拆分.

第二范式: 非主键列完全依赖于主键,而不能是依赖于主键的 一部分.

第三范式: 非主键列只依赖于主键,不依赖于其他非主键。

在设计数据库结构的时候,要尽量遵守三范式,如果不遵守,必须有足够的理由.比如性能. 事实上 我们经常会为了性能而妥协数据库的设计。