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。