# 事务隔离级别

1：读未提交 一个事务可以读取另一个未提交事务的数据 **脏读**

2：读提交 就是一个事务要等另一个事务提交后才能读取数据 **不可重复读**

3：重复读 就是在开始读取数据（事务开启）时，不再允许修改操作 **幻读**

4：Serializable 事务串行化顺序执行，可以避免脏读、不可重复读与幻读

# 缓存

本地缓存:耗时低 占用内存 多台机器会有冗余 同步不便

远程缓存:例如Redis等 基于tcp传输

什么样的文件需要缓存

热点数据,查多改少,频繁加载的..

### 缓存更新

经典的缓存+数据库读写的模式：

读的时候，先读缓存，缓存没有的话，那么就读数据库，然后取出数据后放入缓存，同时返回响应

更新的时候，更新数据库，删除缓存

### 缓存更新算法

FIFO先进先出队列:根据缓存生成的顺序

LRU 最近最少使用:淘汰最少使用的缓存 维护一个有效队列 最近被使用的放到队列头部

LFU最近最常使用:

### 启动方式

冷启动:第一次启动无缓存 请求过多或者缓存丢失可能会导致数据库崩溃

缓存预热:提前放入部分需求高的缓存

### 缓存穿透

通过统计总调用数、缓存层命中数、存储层命中数等数据，来监测缓存命中率及穿透情况

查询没有命中各级缓存 直接访问数据存储层 数据层压力过大导致异常

比如查询没有的数据 过期数据等

解决方法:

1:缓存空对象

存储层不命中后,仍然将空对象保留到缓存层,保护后端数据源

同时空值作缓存 无效数据占用大量存储空间 可以设置较短的有效期 过期删除

并且存储层数据更新 需要做额外的同步处理

2:过滤器拦截

将存储层所有存在的key放到一个容器中 对存储层中不存在的key进行过滤拦截

### 缓存雪崩

缓存服务器重启或者某一时间段大量缓存失效(缓存同时过期) 存储层也会压力过大

雪崩原因有很多

1:穿透导致 设置高可用的缓存层 个别节点,服务器 宕机 其他的正常工作

2:依赖隔离组件为后端限流并降级 比如个人推荐功能 服务挂了可以推送热点数据

3:设计多重缓存

### 缓存击穿key重建优化

击穿是大量请求相同的key,但是key刚好过期 正在重构缓存

缓存+过期时间策略比较常用 有两个问题

1:当前key热点度高 并发量会很大

2:重建缓存需要时间 可能是复杂SQL 多次IO等

缓存失效瞬间 大量线程重建缓存 后端复杂过大

解决方案:

1:互斥锁

只允许一个线程重建缓存，其他线程等待重建缓存的线程执行完，重新从缓存获取数据

2:缓存不过期

将缓存设置不过期,但是设置一个逻辑上的过期时间,定期的去更新他

重构缓存期间可能会出现不一致状况

### Redis锁

Redis锁分为INCR,SETNX,SET

#### INCR锁

1表示获取锁成功 2表示失败

如果key不存在,key的值首先初始化为0,然后执行INCR操作进行+1

然后其他用户在执行INCR操作+1,返回的数大于1,则说明锁正在被使用

#### SETNX锁

如果key不存在,将key设置为value,如果key已经存在,则不作任何处理

A请求服务器设置key值,设置成功加锁成功

B请求服务器设置key值,返回失败则加锁失败

#### SET锁

上述两种锁需要将key设置一个过期时间,以防异常退出锁未删除 缓存不能更新的情况

SET锁通过事务保证原子性完成上述操作

#### 拓展问题

1:Redis发现锁失败了如何处理?中断请求还是循环请求?

使用循环请求去获取锁

2:循环请求的话,如果有一个获取了锁,其他的再去获取锁,会发生抢锁吗?

会,可以在循环请求的时候,加入睡眠功能,等待几毫秒

3:锁提前过期后,A还没执行完,B获取到了锁,然后A执行完删锁,B的锁会受影响吗?

加锁存的key是随机的,每次删除key的时候判断存入的key的数据是否相同再删

# mysql索引

5.5之前使用SAM引擎 5.5之后使用DB引擎

SAM 主键,唯一,非主键索引都是B-Tree