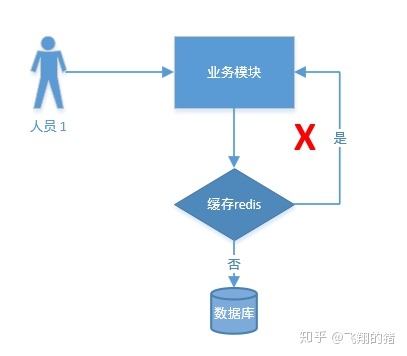
**缓存穿透和缓存雪崩**

缓存穿透和缓存雪崩

**缓存穿透**

概念

所谓缓存穿透就是说在缓存中不存在，然后直接在数据库中查询的现象，图例如下：



**场景**

一般来说，缓存穿透的场景发生在故意攻击的场景下；比如说，本来查询意见商品的序号是正数，但是请求方总是请求大量的负数过来，导致缓存无效，全部流量都打在了数据库中，如果某一时刻流量过大，则会导致数据库崩溃；

**解决方案**

**方案一：布隆过滤器**

布隆过滤器会判断某个元素是否在某个几个中，相关的知识点会专门去介绍。

**方案二：缓存空值**

可以把一些不符合要求的数据的key值设置为NULL，这样就不会一直去查数据库了但是需要设置过期时间；

最好就是在redis前面加一个布隆过滤器，这样很降低缓存穿透的概率；

**伪代码**

////方案一

//伪码

string penetrate1(int ID,string str)

{

//使用布隆过滤器，后续专门写一篇博客

if(存在)

{

redis操作

}

else

{

return NULL;

}

}

////方案二

//伪码

string penetrate1(int ID,string str)

{

//假设可以就是ID

int CacheTime = 30;

string CacheValue;

char cmd[128];

char sql[128];

memset(cmd,0,sizeof(cmd));

memset(sql,0,sizeof(sql));

sprintf(cmd,"get %d",ID);

redisReply\* replay = (redisReply\*)redisCommand(conn,cmd);

if(replay->type == REDIS\_REPLY\_ERROR)

{

//查询数据库

mysql\_query(&mysql,sql);

MYSQL\_RES \*result= mysql\_store\_result(&mysql);

int row = mysql\_num\_rows(result);

//查询为空

if(row < 0)

{

//设置默认值

memset(cmd,0,sizeof(cmd));

sprintf(cmd,"set %d ''",ID);

redisReply\* replay = (redisReply\*)redisCommand(conn,cmd);

}

//数据库数据同步到redis中

//set XX XX

redisReply\* replay = (redisReply\*)redisCommand(conn,cmd);

//返回数据

return 结果；

}

else

{

return replay->str;

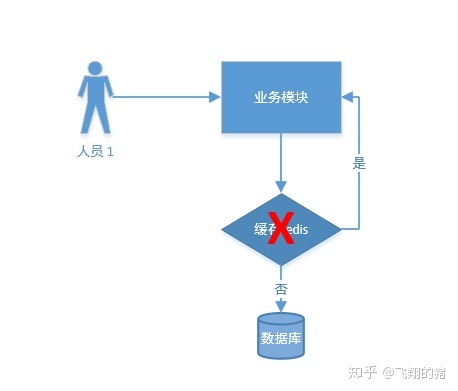
}

}

**缓存雪崩**

概念

所谓缓存雪崩就是在某一个时刻，缓存集大量失效。所有流量直接打到数据库上，对数据库造成巨大压力；



**场景**

基本我们能想到的场景就是一些电商抢购的现象，一般就是比如12点开始，在1点的时候大量同时失效，这个时候就会造成缓存雪崩的现象；

解决方案

方案一：加锁/队列

加锁/队列；这样虽然能降低数据库压力，但是同时，响应也很慢

方案二：缓存标记

给每一个缓存数据增加相应的缓存标记，记录缓存的是否失效，如果缓存标记失效，则更新数据缓存

方案三：缓存过期时间错开

设置缓存时间错开，可以在设置过期时间的时候，加一个一定范围内的随机值来错开