点赞数和评论数的去标准化，虽然有可能导致获取的点赞数和评论数会不一致，但可以加快速度，在社交系统中这很重要，比一致性重要，而金融系统一致性最重要。

点赞数和评论数标准的取法是从数据库表中采用SELECT COUNT(\*) FROM Table，例如TweetSerializer中包含点赞数和评论数，前端渲染tweets列表时，显示时有10个tweet，那么点赞数和评论数就分别需要10次Query，这会导致N次Queries，而且sql语句中的COUNT(\*)也比较耗时，导致总的查询非常耗时。

解决办法是点赞数和评论数进行denormalization去标准化，都存储在tweet表中，相当于冗余存储。

所谓denormalization就是之前标准化获取点赞数和评论数只能通过SELECT COUNT(\*) FROM table来获取，现在可以直接从tweet的记录中直接查询出来。好处是查询tweet记录时就可以查询到点赞数和评论数，不用额外的查询，坏处是可能会产生一致性的问题，这时需要明确一个source of truth，即绝对准确的来源，点赞数和评论数的source of truth显然是SELECT查询表。

这种不一致的问题，可以不处理，例如点赞数和评论数就不需要处理，如果处理，可以采用随机的方式，在一定概率下取SELECT查询表来获取点赞数和评论数，大致代码如下：





mysql中这种操作是原子操作，它有row lock的功能，但不能先取出来，加上1，再写会去，这样无法保证是原子操作。

点赞操作时不要尝试把cache中的tweet和tweets列表失效掉，以保证likes\_count会被更新，这样做如果用户时明星用户，那么就会频繁的失效cache中的tweet，会降低访问速度。同理评论操作也不要失效cache。

点赞数和评论数会经常改变，要单独缓存到Redis。一般来说经常容易改变的对象不应该缓存在memcached或Redis中，除非像点赞数和评论数这样cache提供了相应的+1和-1操作，而且要单独缓存，不要和其他数据一起缓存。