关注推送

使用feed流

Feed 流产品有两种常见模式：

Timeline：不做内容筛选，简单的按照内容发布时间排序，常用于好友或关注。

例如朋友圈

优点：信息全面，不会有缺失。并且实现也相对简单

缺点：信息噪音较多，用户不一定感兴趣，内容获取效率低

智能排序：利用智能算法屏蔽掉违规的、用户不感兴趣的内容。推送用户感兴趣信息来吸引用户

例如抖音，快手

优点：投喂用户感兴趣信息，用户粘度很高，容易沉迷

缺点：如果算法不精准，可能起到反作用

本例中的个人页面，是基于关注的好友来做 Feed 流，因此采用 Timeline 的模式。

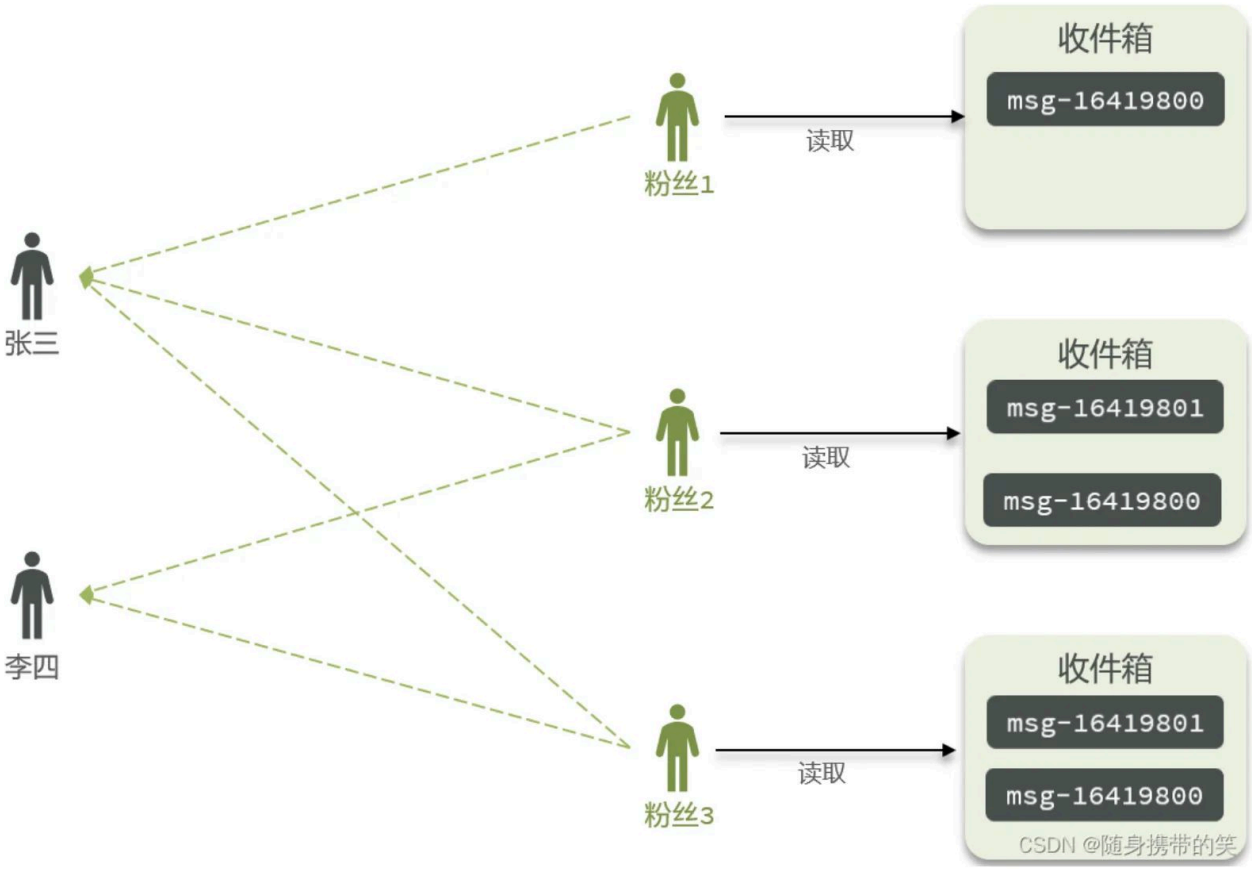
该模式的实现方案有三种：拉模式、推模式、推拉结合

拉模式：也叫做读扩散

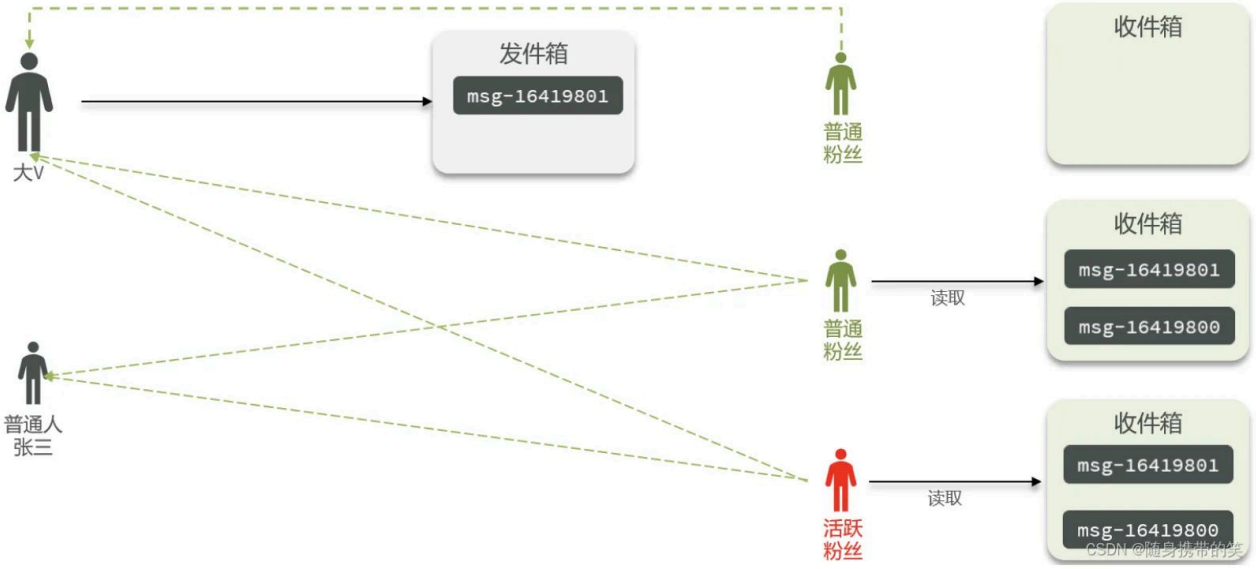
每次读的时候获取消息，内存消耗小，但读操作过于频繁，若用户关注了许多博主，一次要读的消息也是十分多，造成延迟较高

推模式：也叫做写扩散。

发消息时写入粉丝收件箱，内存占用更高，写操作频繁，若博主有许多粉丝，写操作更加繁重



推拉结合模式：也叫做读写混合，兼具推和拉两种模式的优点。
普通博主，粉丝少，可以采用推模式，写操作并不是很繁重
大v博主，粉丝多；分两种粉丝，活跃粉，普通粉；活跃粉，数量少，可以采用推模式；普通粉，数量多，但上线查看少，采用拉模式，什么时候看什么时候拉取。



基于推模式实现关注推送功能

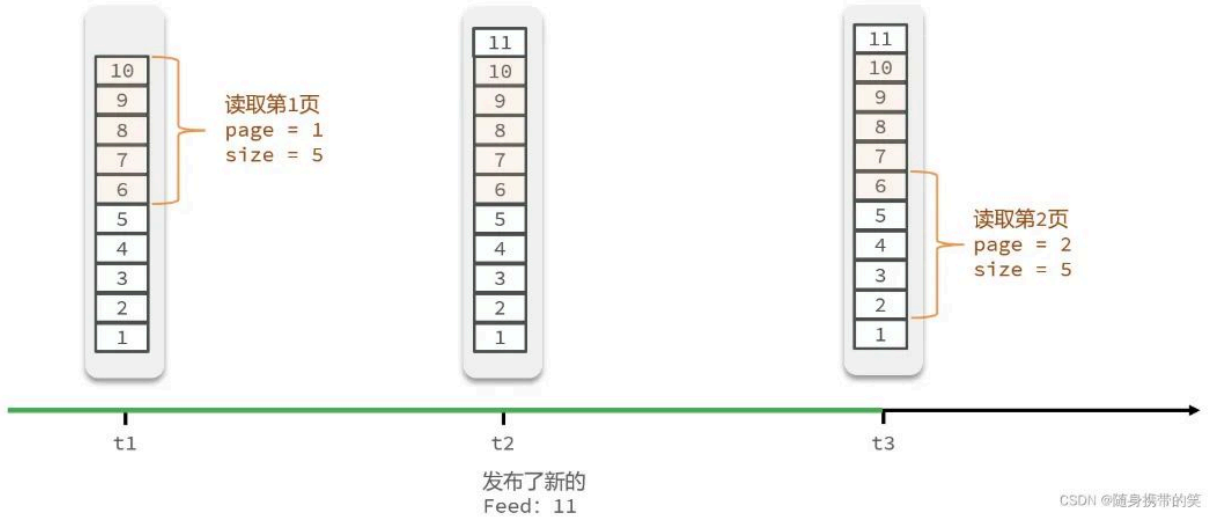
需求：

①修改新增探店笔记的业务，在保存 blog 到数据库的同时，推送到粉丝的收件箱

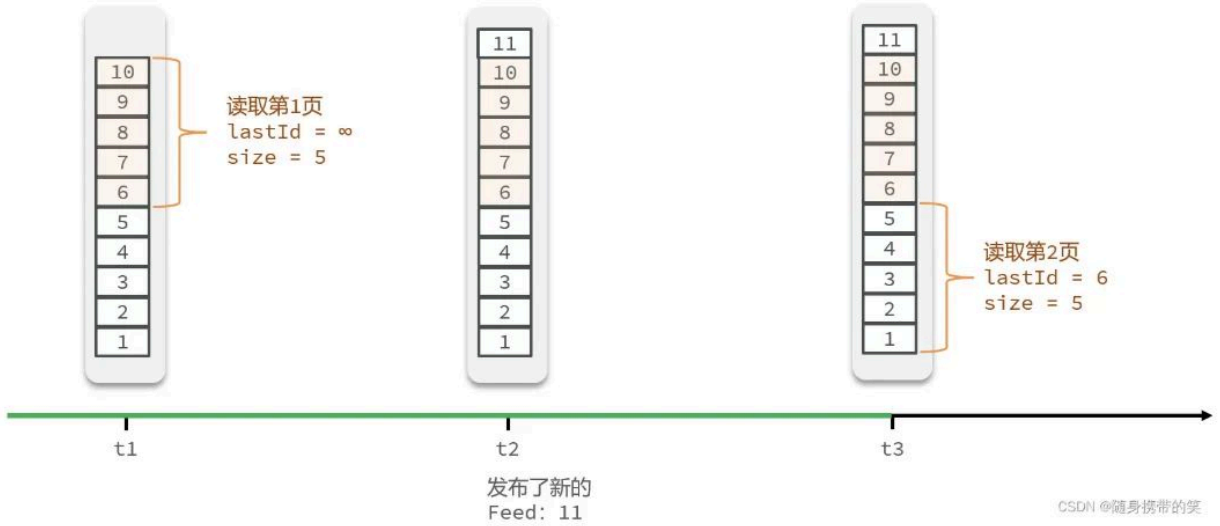
②收件箱满足可以根据时间戳排序，必须用 Redis 的数据结构实现

③查询收件箱数据时，可以实现分页查询

- Feed 流中的数据会不断更新，所以数据的角标也在变化，因此不能采用传统的分页模式。



- 滚动分页



Java

```java
public Result saveBlog(Blog blog) {
    // 获取登录用户
    UserDTO user = UserHolder.getUser();
    blog.setUserId(user.getId());
    // 保存探店博文
    boolean isSuccess = save(blog);
    if (!isSuccess){
        return Result.fail("发布失败，请检查重试");
    }
    // 查询博文作者的所有粉丝
    List&lt;Follow> follows = followService.query().eq("follow_user_id", user.getId()).list();
    for (Follow follow : follows) {
```

```
13          // 获取粉丝id
14          Long userId = follow.getUserId();
15          // 推送笔记id给所有粉丝
16          String key = "feed:" + userId;
17          stringRedisTemplate.opsForZSet().add(key, blog.getId().toString(),
    System.currentTimeMillis());
18      }
19      // 返回id
20      return Result.ok(blog.getId());
21  }
```