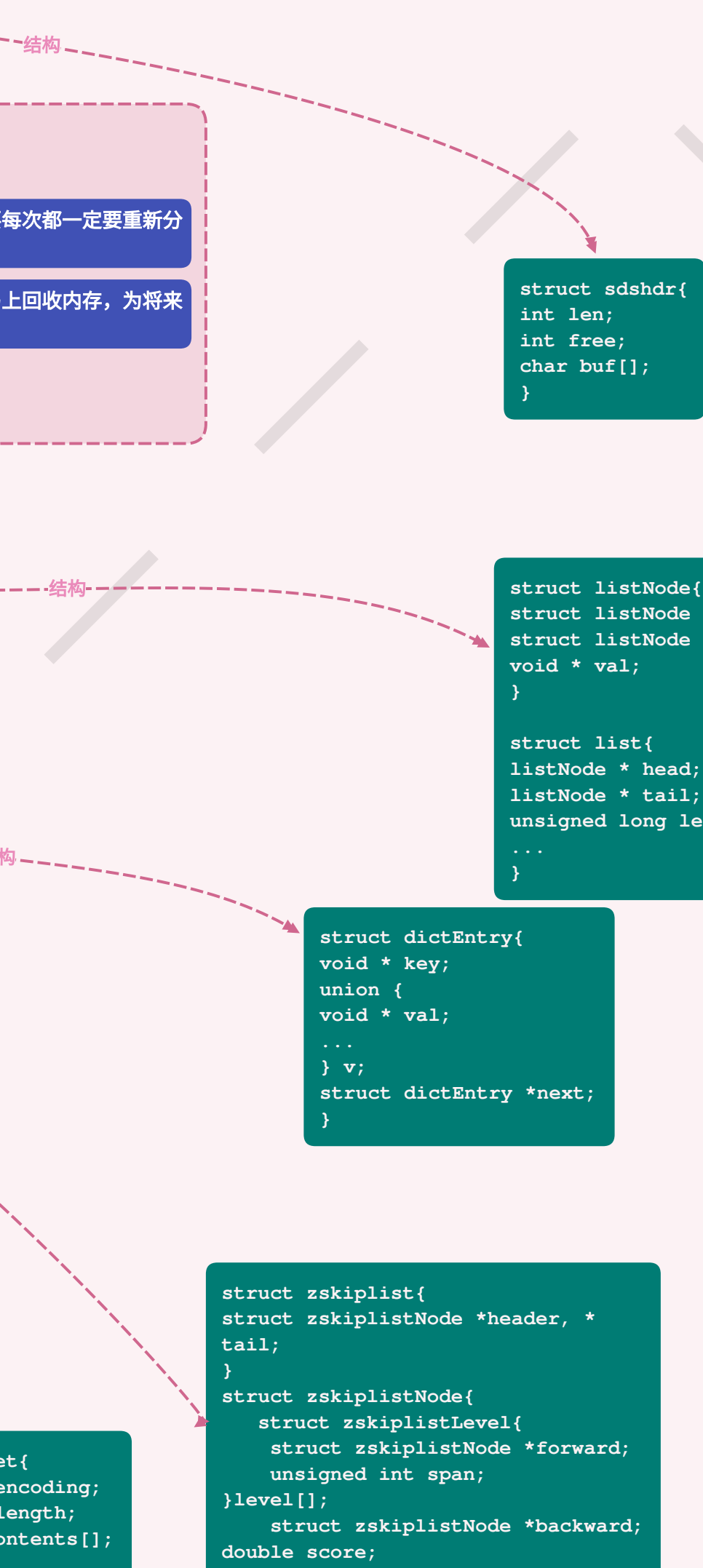
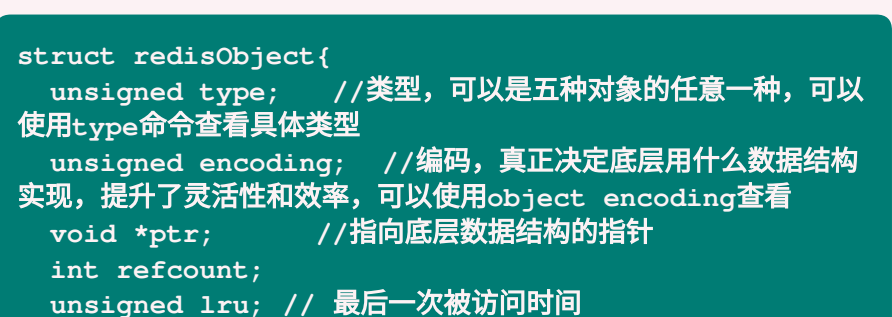
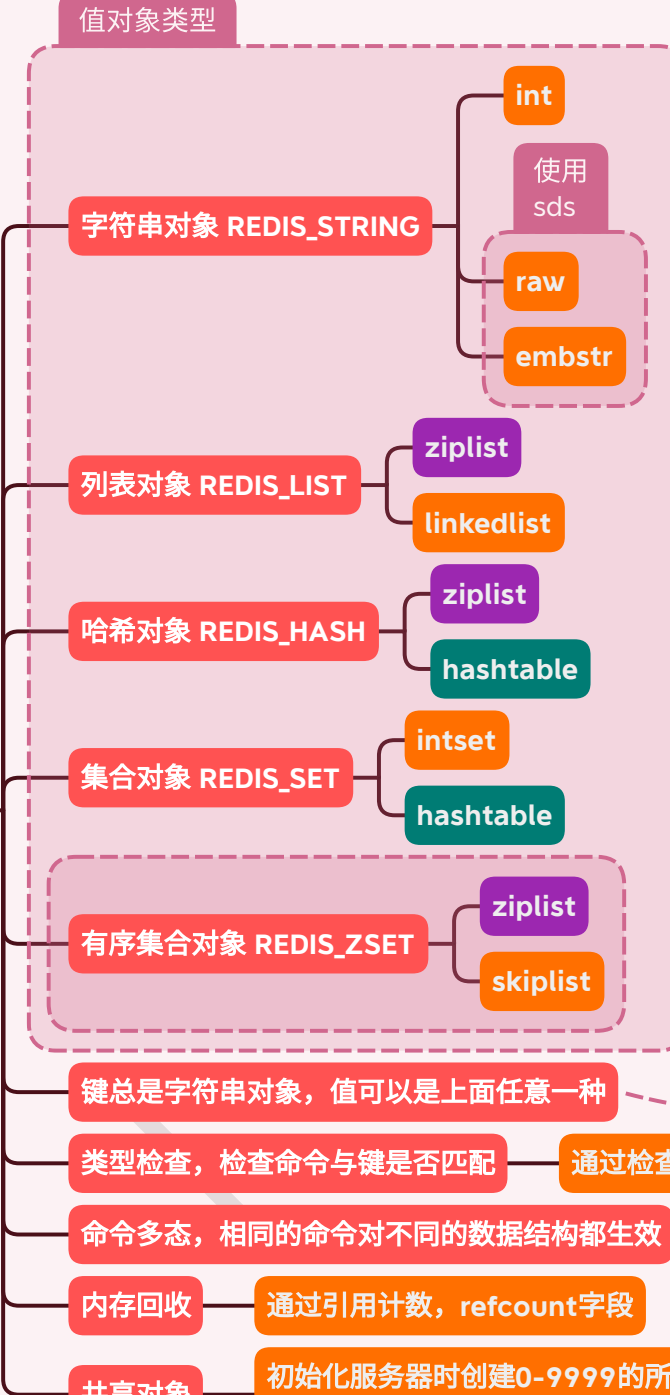


Redis

底层数据结构

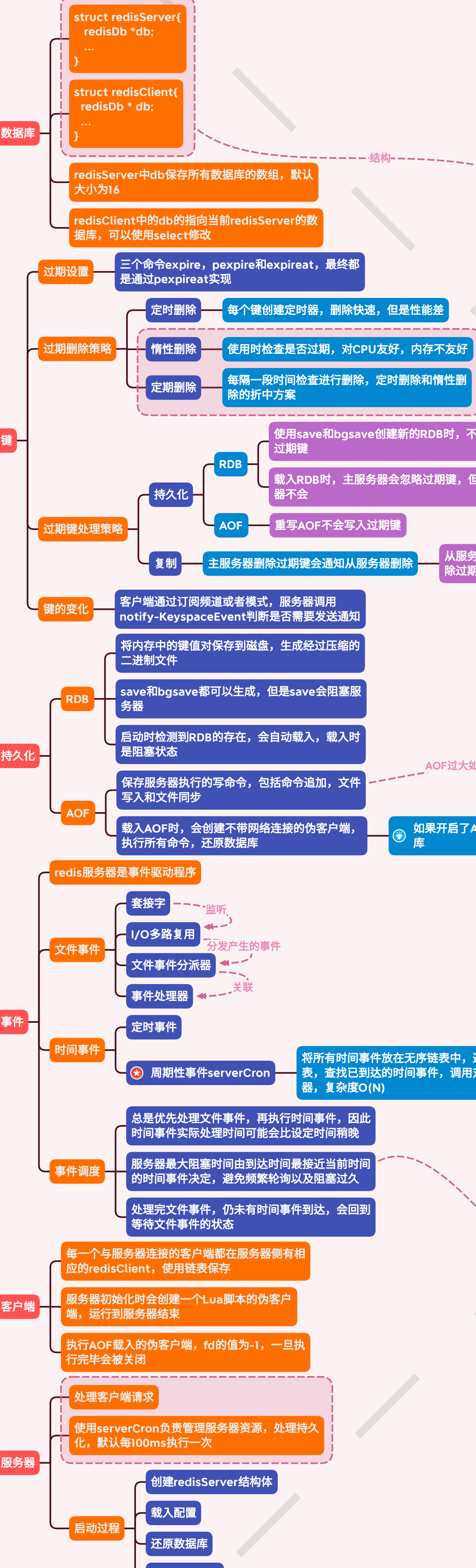


对象

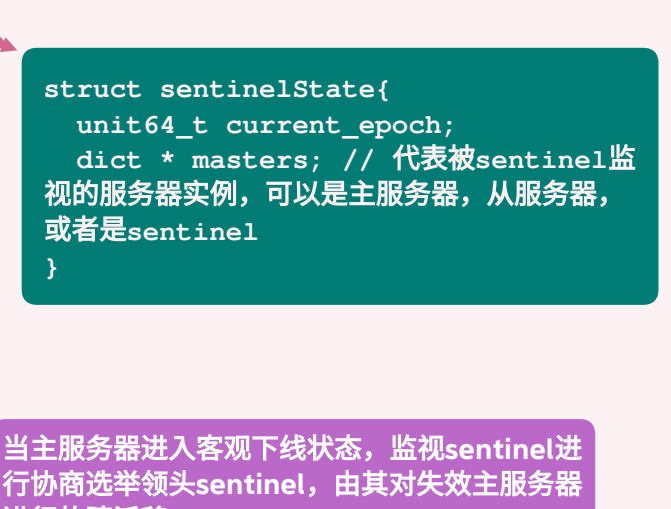
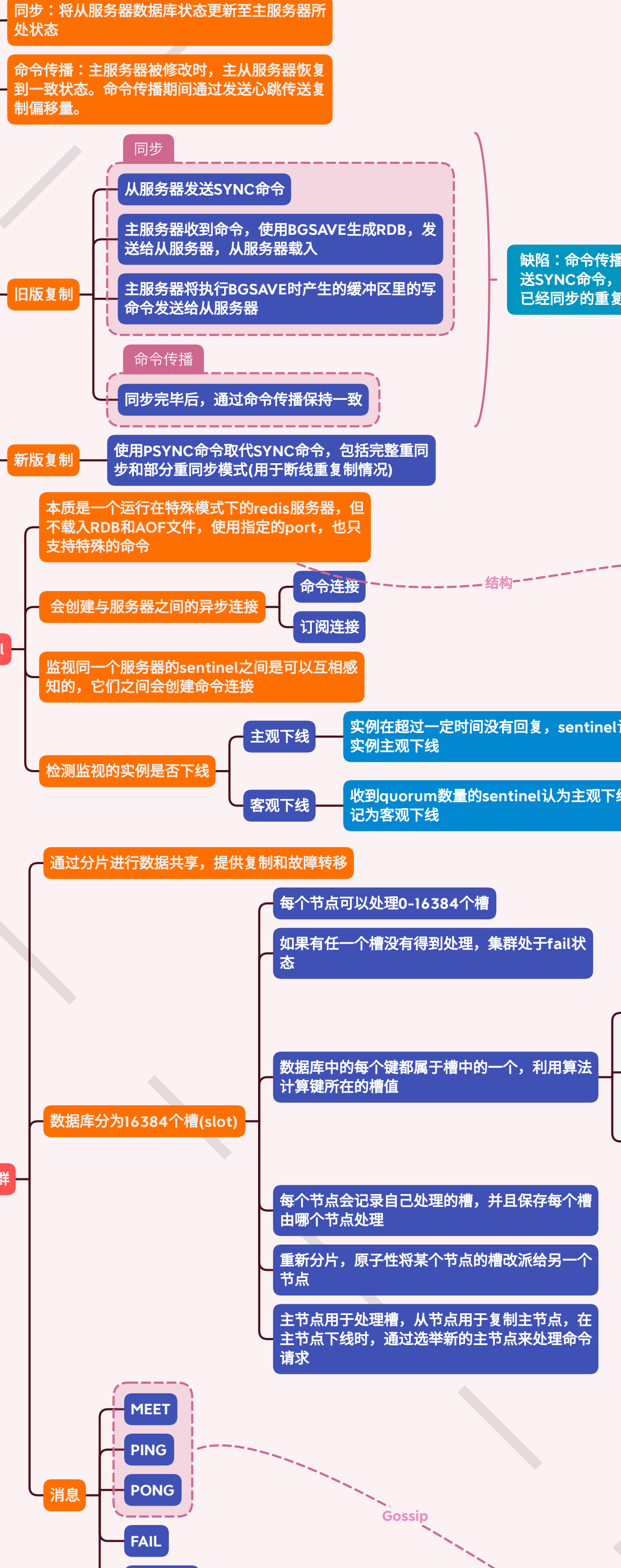


- 键总是字符串对象, 值可以是上面任意一种
- 类型检查, 检查命令与键是否匹配 通过检查type实现
- 命令多态, 相同的命令对不同的数据结构都生效 根据encoding实现, 调用指定数据结构API
- 内存回收 通过引用计数, refcount字段
- 共享对象 初始化服务器时创建0-9999的所有整数值作为共享对象

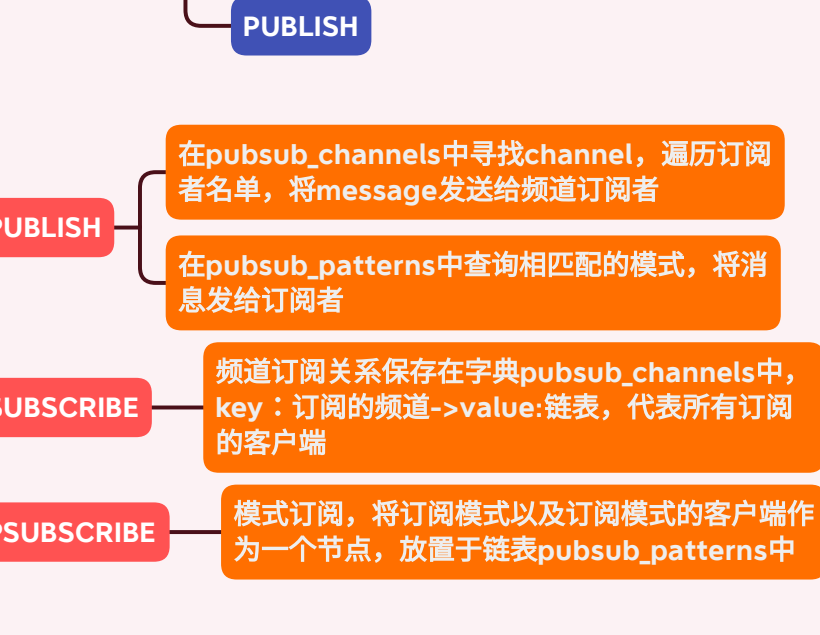
单机数据库



集群



发布/订阅



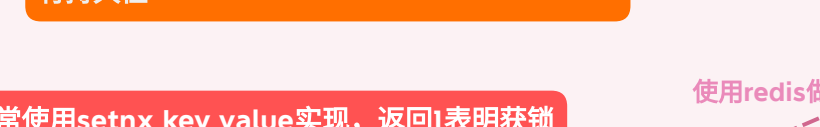
消息正文一样, 根据type来判断是哪种消息。每次发送时, 都从已知节点中选取两个节点

事务



watch命令监视键, 当键被修改时, 会将所有监视该键的客户端的redis\_dirtyCAS打开, 打开了该标志位的客户端, 服务器将拒绝执行其提交的事务

分布式锁



使用redis做分布式锁真的可靠吗?

- 1. 没有设置超时时间, 可能导致锁永远没有被释放。而追加expire语句, 会导致所有操作不是原子性的
- redis 2.8后可以使用ex和xx来实现原子操作
- 2. 如果设置了超时时间, A执行太久导致超时释放, B可能获取锁, 最后A执行完毕将B的锁释放
- 可以通过守护进程, 发现即将超时的就延长时间
- 3. 使用主从机制, 保证redis锁高可用, 但是可能导致从master获得的锁还没有同步到slave上, 就有新的客户端从slave上获取到了锁
- 针对多redis节点, 只能使用redlock解决, 从大部分节点中获取锁成功, 才能视为获取成功