面经 -1

作者：hantiaotiao
原文链接：https://ld246.com/article/1677735565212
来源网站：链滴

<h2 id="电话面试">电话面试</h2>
<h3 id="简历存在问题">简历存在问题</h3>
项目逻辑复杂、冷门，很难给对方表述清楚
改进：项目修改为 OJ 平台
<h3 id="相关问题">相关问题</h3>
<h4 id="1-MySQL分库分表">1、MySQL 分库分表</h4>

为什么要分库分表？
当一张表的数据达到几千万时，你查询一次所花的时间会变多，如果再有联合查询的话，我想有能会死在那儿了。而分库主要是为了面对高并发的读写访问。总的来说目的都是减小数据库的负担，短查询时间。

具体是如何对数据库如何进行垂直拆分或水平拆分的？
水平拆分的意思，就是把一个表的数据给弄到多个库的多个表里去，但是每个库的表结构都一样只不过每个库表放的数据是不同的，所有库表的数据加起来就是全部数据。水平拆分的意义，就是将据均匀放更多的库里，然后用多个库来扛更高的并发，还有就是用多个库的存储容量来进行扩容。 </
>

<img src="https://ld246.com/images/img-loading.svg" alt="" data-src="https://b3logfile
com/file/2023/03/solo-fetchupload-5466394884620228882-eAZ0Tnn.png?imageView2/2/inte
lace/1/format/jpg">
垂直拆分的意思，就是把一个有很多字段的表给拆分成多个表，或者是多个库上去。每个库表的结构不一样，每个库表都包含部分字段。一般来说，会将较少的访问频率很高的字段放到一个表里去，然将较多的访问频率很低的字段放到另外一个表里去。因为数据库是有缓存的，你访问频率高的行字段少，就可以在缓存里缓存更多的行，性能就越好。

分库分表后，id 主键如何处理？
snowflake 算法是 twitter 开源的分布式 id 生成算法，一个 64 位的 long 型的 id，1 个 bit 是不用，用其中的 41 bit 作为毫秒数，用 10 bit 作为工作机器 id，12 bit 作为序列号。
<blockquote>

1 bit：不用，为啥呢？因为二进制里第一个 bit 为如果是 1，那么都是负数，但是我们生成的 id 都是正数，所以第一个 bit 统一都是 0。
41 bit：表示的是时间戳，单位是毫秒。41 bit 可以表示的数字多达 2^41 - 1，也就是可以标识 ^41 - 1 个毫秒值，换算成年就是表示 69 年的时间。
10 bit：记录工作机器 id，代表的是这个服务最多可以部署在 2^10 台机器上哪，也就是 1024 机器。但是 10 bit 里 5 个 bit 代表机房 id，5 个 bit 代表机器 id。意思就是最多代表 2^5 个机房（3 个机房），每个机房里可以代表 2^5 个机器（32 台机器）。
12 bit：这个是用来记录同一个毫秒内产生的不同 id，12 bit 可以代表的最大正整数是 2^12 - 1 = 4096，也就是说可以用这个 12 bit 代表的数字来区分同一个毫秒内的 4096 个不同的 id。

</blockquote>
算法可见：<a href="https://ld246.com/forward?goto=https%3A%2F%2Fmp.weixin.qq.co
%2Fs%3F__biz%3DMzU5NTAzNjM0Mw%3D%3D%26mid%3D2247486846%26idx%3D1%26s
%3Da2de9e617899df46e9e5352b1c2d7af9%26chksm%3Dfe795ca6c90ed5b0991a770fa1d43f
04140821503cccbfa720b1791a27f40a16704ca43cd4e%26scene%3D21%23wechat_redirect" ta
get="_blank" rel="nofollow ugc">面试官：分库分表之后，id 主键如何处理？ (qq.com)</p

<blockquote>
怎么说呢，大概这个意思吧，就是说 41 bit 是当前毫秒单位的一个时间戳，就这意思；然后 5 bit

是你传递进来的一个机房 id（但是最大只能是 32 以内），另外 5 bit 是你传递进来的机器 id（但是大只能是 32 以内），剩下的那个 12 bit 序列号，就是如果跟你上次生成 id 的时间还在一个毫秒内那么会把顺序给你累加，最多在 4096 个序号以内。
所以你自己利用这个工具类，自己搞一个服务，然后对每个机房的每个机器都初始化这么一个东，刚开始这个机房的这个机器的序号就是 0。然后每次接收到一个请求，说这个机房的这个机器要生一个 id，你就找到对应的 Worker 生成。
利用这个 snowflake 算法，你可以开发自己公司的服务，甚至对于机房 id 和机器 id，反正给你留了 5 bit + 5 bit，你换成别的有业务含义的东西也可以的。
这个 snowflake 算法相对来说还是比较靠谱的，所以你要是真搞分布式 id 生成，如果是高并发的，那么用这个应该性能比较好，一般每秒几万并发的场景，也足够你用了。
</blockquote>

<h4 id="2-创建表的时候需要注意什么">2、创建表的时候需要注意什么</h4>

尽量做到冷热数据分离，减小表的宽度。更有效的利用缓存，避免读入无用的冷数据
经常一起使用的列放到一个表中 <code>（避免更多的关联操作）</code>
消除冗余，减少存储成本。不过有时为了提高运行效率，适当保留冗余数据
命名规范
注释

<h4 id="3-外键相关">3、外键相关</h4>

为什么要用外键使用外键
使用外键可以让数据库自身保证数据一致性，完整性，更可靠。
但是外键也有很多弊端

外键会降低数据库性能。总是需要对外键相对应的表查询有无对应的数据
复杂的外键关联导致维护困难

<h4 id="4-如何去优化一条SQL语句-explain">4、如何去优化一条 SQL 语句？explain</h4>

<blockquote>
详情可见：mysql explain 的用法 - 简书 (jianshu.com)
</blockquote>
<h4 id="5-redis什么时候把内存中的数据同步到数据库">5、redis 什么时候把内存中的数据同步到据库</h4>

AOF：通过保存数据库执行的命令来记录数据库的状态
RDB：实现方式是将存在 Redis 内存中的数据写入到 RDB 文件中保存到磁盘上从而实现持久化的
<blockquote>
详情：(2 条消息) Redis 实战：服务器断电，内存上面的数据就会丢失了。这个问题显然怎么解决？_php redis 服务断开重连数据会丢麽_卡卡的 Java 架构笔记的博客-CSDN 博客
</blockquote>

<h4 id="6-简单描述一下项目的业务流程">6、简单描述一下项目的业务流程</h4>