进来偷学一招，数据归档二三事儿

原创 楼下小黑哥 小黑十一点半 2021年07月10日 18:07

随着业务的快速增长，业务体量变得越来越大，这个过程我们会碰到各种问题，倒逼着我们进行技术升级。

那今天我们来聊下，这个过程将会碰到关于数据的问题。

数据增长带来的烦恼

业务快速增长，业务表数据记录不断在增加，这就会带来两个问题。

第一，数据库数据最终将会保存在本地磁盘中，数据记录越多，磁盘占用空间就会越多，对应剩余可用空间就会越少。

剩余空间到达一定的阈值之后，将会引发磁盘空间的持续报警，消耗宝贵的数据库生产服务器的资源。

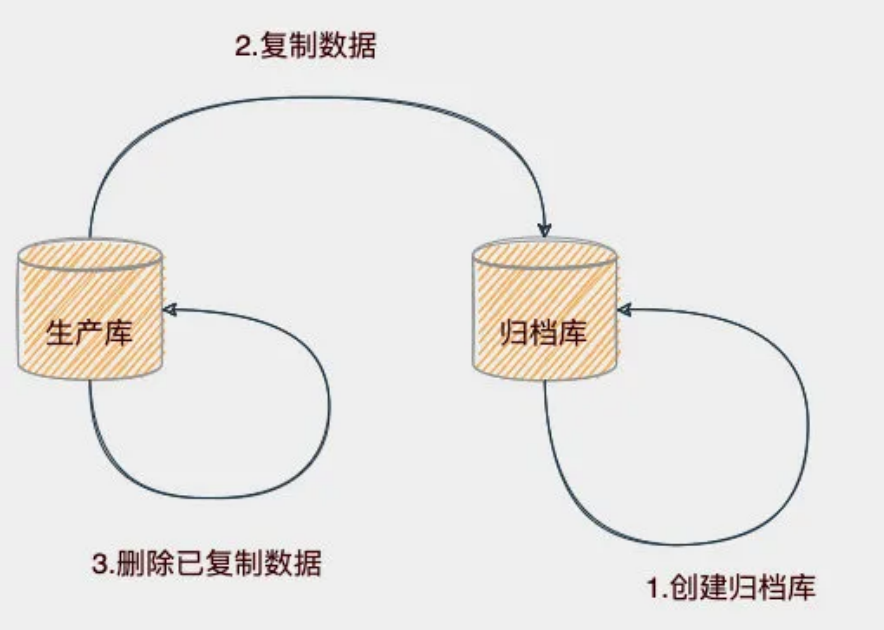
第二，业务表记录越多，表查询的效率就会相应变低，另外表变更也会变的很麻烦。

那解决这个问题，解决办法有很多，那 今天介绍其中一种方式，数据归档。

数据归档

数据归档的解决思路非常简单，就是将生产库的数据转移到拥有相同表结构的数据库中，通过减少生产库记录数量，从而提高数据查询等操作的效率。

数据归档的流程如图所示：



数据归档

数据归档分为三个流程

- 创建一个新的数据库-归档库，然后在归档库创建与生产库相同的表
- 不断查询生产库数据记录，同步复制到归档库
- 生产库删除已经复制的数据记录

虽然数据数据归档流程非常简单，但是设计数据归档的方案，我们必须想清楚以下几个问题：

- 归档前：那些数据可以归档？归档库如何选型？
- 归档中：数据归档的执行方案
- 归档后：数据归档带来问题预案

归档之前

首先我们需要思考第一个问题，那些数据可以归档？

表数据量很大，并且存在很明显的冷热数据，冷数据几乎很少访问。

一个非常典型的例子，订单数据。我们通常访问最近的历史订单，而一年前的历史订单查看就会很少。

又比如优惠券数据，还未过期的优惠券，可能需要被查询使用。而一年前过期的，或者说已经被使用的优惠券，就会被很少访问。

所以，设计数据归档的之前，我们需要思考，我们归档的数据是否适合被归档。

第二，我们需要思考归档数据库的选型。

数据归档主要目的是为了节约宝贵的生产服务器存储，数据需要经过压缩后才会存到数据库。

所以，归档库需要选择那些支持高压缩比的存储引擎。

我们目前归档库选型使用 TokuDB 引擎的 MySQL数据库，压缩比大约为6:1。

归档之中

上面流程我们看到，数据归档无非就是查询数据，插入数据，然后删除数据。

那我们可以基于这个流程开发一个通用的归档脚本。

Google 搜索了一圈，在 Github 上找到了一个归档小工具，基本上实现了数据归档的自动运转，统一的归档任务调度管理、自动监控和预警、自动生成报表。在一定程度上节约了生产力，提高了运维效率。

github地址：https://github.com/dbarun/mysql_archiver

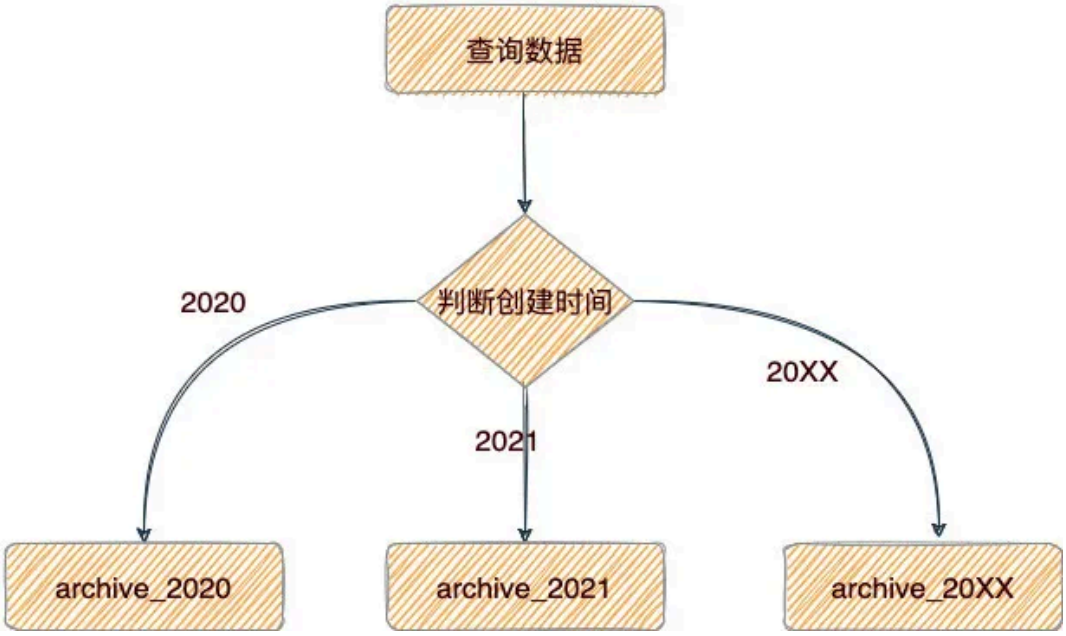
特殊归档需求

一般来说，通用数据归档脚本可以满足大部分情况。但是如果数据归档有一些特殊的要求的话，那就需要自己开发。

比如说，我们有一个项目，数据归档的需求是批量查询 90 天前的数据，然后将这些数据插入到当年的归档表中。

举个例子，如果当前数据记录创建时间为 2020-12-31，这个记录将会归档到 archive_2020 表中。

那如果这个数据记录创建时间为 2021-01-01，那这个记录就会被归档到 archive_2021 表中。



数据归档

那像这种有明显业务需求数据归档方式，那就需要我们在项目中自己开发。

归档注意事项

第一，数据归档过程需要不断的读写生产库，这个过程将会大量使用的网络、IO。那为了防止对线上业务造成压力，数据归档一般只在业务低峰期执行。

另外我们需要尽可能调优数据，尽量降低对线上业务的影响。

第二，数据归档之后，将会删除生产库的数据，这些数据删除之后，将会造成数据空洞。即数据删除之后，表空间并未及时的释放，当长时间没有新的数据填充，会造成空间浪费的情况。

所以数据删除之后，我们需要及时优化数据空洞，释放这些被浪费的空间。

第三，如果数据归档中，影响了线上业务，那一定要及时止损，结束数据归档，然后复盘问题，及时找到问题。

归档之后

数据归档之后，将会带来一些问题，我们需要及时想好这些的预案。

数据幂等被破坏

生产数据库，我们可以使用唯一索引，防止插入重复数据。

但是数据归档之后，部分数据被归档到归档库，这样生产库就又可以插入这些数据库，这就会造成业务上插入重复的数据。

那这个问题，我们可以使用 ID 发号器解决。生产数据库唯一索引存储 ID 发号生成的 ID，ID 发号器每天单调递增，那理论上就不会重复的 ID。

归档查询库 **RT** 较高

由于归档数据库使用高压缩比的存储引擎，这就会导致归档库查询 RT 变高，例如生产库查询是1ms 的rt，用 tokudb 会变成2ms。

那这个问题，我们就需要从业务上去思考，是否可以接受。

如果你是后台类查询业务，可以接受高 RT 的查询，那我们完全可以使用归档库。

那如果你是前台类用户侧查询，查询 RT 要低，那就不能接受查询归档库。

但是从另一方面来讲，如果业务上要求查询 RT 一定要比较低，那这些数据真的适合被归档吗？

归档数据缺失，造成业务影响

数据归档之后，生产库就会缺失这部分数据，那如果业务上正好需要使用这些数据，那就会造成业务上异常。

比如说，支付业务中，退款一般需要支持一年以内的订单。那如果退款的时候，正交易支付的数据正好被归档，那就会造成退款的时候找不到对应的支付数据，造成退款失败。

那这个问题解决办法有两种，第一个解决办法，双重查询。如果生产数据库找不到业务数据，那就去归档库查找。

这个解决办法适合离线的业务。

第二个解决办法，设计一个兼容方案，提供数据逆向接口，反向将归档数据库的记录重新还原到生产库。

那这种解决办法，只适合少量因为归档数据造成业务异常的业务场景。

总结

数据归档可以解决生产数据库因为数据量过多，从而引发磁盘空间预警，表查询、变更效率变低等问题。

但是任务方案都存在双面性，数据归档可能引发数据幂等被破坏、归档查询库 RT 较高、归档数据缺失，造成业务影响等问题。

所以我们设计数据归档的方案时，需要全面考虑，提前准备预案，解决可能造成的业务问题

小黑十一点半

双非本科，支付行业打工人，自学转行 Java。在这里我会分享支付行业相关技术文章...

96篇原创内容

公众号

系统架构设计 5

系统架构设计 · 目录

下一篇 Â· 支付对账系统序章：千万级数据对账怎么这么难？