07｜MySQLHA：如何将“删库跑路”的损失降到最低？

你好，我是李玥。

对于任何一个企业来说，数据安全的重要性是不言而喻的。我在开篇词中也曾经强调过，凡是涉及到数据的问题，都是损失惨重的大问题。

能够影响数据安全的事件，都是极小概率的事件，比如说：数据库宕机、磁盘损坏甚至机房着火，还有最近频繁出现在段子中“程序员不满老板删库跑路”的梗儿，但这些事儿一旦发生了，我们的业务就会损失惨重。

一般来说，存储系统导致的比较严重的损失主要有两种情况，一是数据丢失造成的直接财产损失，比如大量的坏账；二是由于存储系统损坏，造成整个业务系统停止服务而带来的损失。

所谓防患于未然，你从设计一个系统的第一天起，就需要考虑在出现各种问题的时候，如何来保证这个系统的数据安全性。今天我们来聊一聊，如何提前预防，将“删库跑路”等这类问题导致的损失尽量降到最低。

如何更安全地做数据备份和恢复？

保证数据安全，最简单而且有效的手段就是定期备份数据，这样出现任何问题导致的数据损失，都可以通过备份来恢复数据。但是，如何备份，才能最大程度地保证数据安全，并不是一个简单的事儿。

2018年还出现过某个著名的云服务商因为硬盘损坏，导致多个客户数据全部丢失的重大故障。这么大的云服务商，数据是不可能没有备份的，按说硬盘损坏，不会导致数据丢失的，但是因为各种各样的原因，最终的结果是数据的三个副本都被删除，数据丢失无法找回。

所以说，不是简单地定期把数据备份一下就可以高枕无忧了。接下来我们还是以大家最常用的MySQL为例来说一下，如何更安全地来做数据备份和恢复。

最简单的备份方式就是全量备份。备份的时候，把所有的数据复制一份，存放到文件中，恢复的时候再把文件中的数据复制回去，这样可以保证恢复之后数据库中的数据和备份时是完全一样的。在MySQL中，你可以使用mysqldump命令来执行全量备份。

比如我们要全量备份数据库test：

```
$mysqldump -uroot -p test > test.sql
```

备份出来的文件就是一个SQL文件，就是创建数据库、表，写入数据等等这些SQL，如果要恢复数据，直接执行这个备份的SQL文件就可以了：

```
$mysql -uroot test < test.sql
```

不过，全量备份的代价非常高，为什么这么说呢？

首先，备份文件包含数据库中的所有数据，占用的磁盘空间非常大；其次，每次备份操作都要拷贝大量数据，备份过程中会占用数据库服务器大量的CPU、磁盘IO资源，并且为了保证数据一致性，还有可能会锁表，这些都会导致备份期间，数据库本身的性能严重下降。所以，我们不能经常对数据库执行全量备份。

一般来说，每天执行一次全量备份已经是非常频繁了。那这就意味着，如果数据库中的数据丢了，那只能恢复到最近一次全量备份的那个时间点，这个时间点之后的数据还是丢了。也就是说，全量备份不能做到完全无损地恢复。

既然全量备份代价太高，不能频繁执行，那有没有代价低一点儿的备份方法，能让我们少丢甚至不丢数据呢？还真有，那就是**增量备份**。相比于全量备份，增量备份每次只备份相对于上一次备份变化的那部分数据，所以每次增量备份速度更快。

MySQL自带了Binlog，就是一种实时的增量备份。Binlog里面记录的就是MySQL数据的变更的操作日志，开启Binlog之后，我们对MySQL中的每次更新数据操作，都会被记录到Binlog中。

Binlog是可以回放的，回放Binlog，就相当于把之前对数据库所有数据更新操作按照顺序重新执行了一遍，回放完成之后数据自然就恢复了。这就是Binlog增量备份的基本原理。很多数据库都有类似于MySQL Binlog的日志，原理和Binlog是一样的，备份和恢复方法也是类似的。

下面通过一个例子看一下如何使用Binlog进行备份和恢复。首先使用“show variables like ‘%log_bin%’ ”命令确认一下是否开启了Binlog功能：

```
mysql> show variables like '%log_bin%';
+---------------------------------+-----------------------------------+
| Variable_name                   | Value                             |
+---------------------------------+-----------------------------------+
| log_bin                         | ON                                |
| log_bin_basename                | /usr/local/var/mysql/binlog       |
+---------------------------------+-----------------------------------+
mysql> show master status;
+---------------+----------+--------------+------------------+-------------------+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+---------------+----------+--------------+------------------+-------------------+
| binlog.000001 |    18745 |              |                  |                   |
+---------------+----------+--------------+------------------+-------------------+
```

可以看到当前这个数据库已经开启了Binlog，log_bin_basename表示Binlog文件在服务器磁盘上的具体位置。然后用“show master status”命令可查看当前Binlog的状态，显示正在写入的Binlog文件，及当前的位置。假设我们每天凌晨用mysqldump做一个全量备份，然后开启了Binlog，有了这些，我们就可以把数据恢复到全量备份之后的任何一个时刻。

下面我们做一个简单的备份恢复演示。我们先模拟一次“删库跑路”的场景，直接把账户余额表清空：

```
mysql> truncate table account_balance;
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> select * from  account_balance;
Empty set (0.00 sec)
```

然后我们来进行数据恢复，首先执行一次全量恢复，把数据库恢复到今天凌晨的状态。

```
$mysql -uroot test < dump.sql

mysql> select * from  account_balance;
+---------+---------+---------------------+--------+
| user_id | balance | timestamp           | log_id |
+---------+---------+---------------------+--------+
|       0 |     100 | 2020-02-13 20:24:33 |      3 |
+---------+---------+---------------------+--------+
```

可以看到，表里面的数据已经恢复了，但还是比较旧的数据。然后我们再用Binlog把数据恢复到删库跑路之前的那个时刻：

```
$mysqlbinlog --start-datetime "2020-02-20 00:00:00" --stop-datetime "2020-02-20 15:09:00" /usr/local/var/my

mysql> select * from  account_balance;
+---------+---------+---------------------+--------+
| user_id | balance | timestamp           | log_id |
+---------+---------+---------------------+--------+
|       0 |     200 | 2020-02-20 15:08:12 |      0 |
+---------+---------+---------------------+--------+
```

这时候，数据已经恢复到当天的15点了。

通过定期的全量备份，配合Binlog，我们就可以把数据恢复到任意一个时间点，再也不怕程序员删库跑路了。详细的命令你可以参考MySQL的官方文档中“备份和恢复”这一章。

在执行备份和恢复的时候，有几个要点你需要特别的注意。

第一，也是最重要的，“不要把所有的鸡蛋放在同一个篮子中”，无论是全量备份还是Binlog，都不要和数据库存放在同一个服务器上。最好能做到不同机房，甚至不同城市，离得越远越好。这样即使出现机房着火、光缆被挖断甚至地震也不怕。

第二，在回放Binlog的时候，指定的起始时间可以比全量备份的时间稍微提前一点儿，确保全量备份之后的所有操作都在恢复的Binlog范围内，这样可以保证恢复的数据的完整性。

因为回放Binlog的操作是具备幂等性的（为了确保回放幂等，需要设置Binlog的格式为ROW格式），关于幂等性，我们在《01 | 创建和更新订单时，如何保证数据准确无误？》这节课中讲到过，多次操作和一次操作

对系统的影响是一样的，所以重复回放的那部分Binlog并不会影响数据的准确性。

配置MySQL HA实现高可用

通过全量备份加上Binlog，我们可以将数据库恢复到任何一个时间点，这样至少不会丢数据了。如果说，数据库服务器宕机了，因为我们有备份数据，完全可以启动一个新的数据库服务器，把备份数据恢复到新的数据库上，这样新的数据库就可以替代宕机的数据库，继续提供服务。

但是，这个恢复数据的时间是很长的，如果数据量比较大的话，有可能需要恢复几个小时。这几个小时，我们的系统是一直不可用的，这样肯定不行。

这个问题怎么解决？很简单，你不要等着数据库宕机了，才开始做恢复，我们完全可以提前来做恢复这些事儿。

我们准备一台备用的数据库，把它的数据恢复成主库一样，然后实时地在主备数据库之间来同步Binlog，主库做了一次数据变更，生成一条Binlog，我们就把这一条Binlog复制到备用库并立即回放，这样就可以让备用库里面的数据和主库中的数据一直保持是一样的。一旦主库宕机，就可以立即切换到备用库上继续提供服务。这就是MySQL的高可用方案，也叫MySQL HA。

MySQL自身就提供了主从复制的功能，通过配置就可以让一主一备两台MySQL的数据库保持数据同步，具体的配置方法你可以参考MySQ官方文档中“复制”这一章。

接下来我们说这个方案的问题。当我们对主库执行一次更新操作的时候，主从两个数据库更新数据实际的时序是这样的：

1. 在主库的磁盘上写入Binlog；
2. 主库更新存储引擎中的数据；
3. 给客户端返回成功响应；
4. 主库把Binlog复制到从库；
5. 从库回放Binlog，更新存储引擎中的数据。

也就是说，从库的数据是有可能比主库上的数据旧一些的，这个主从之间复制数据的延迟，称为“主从延迟”。正常情况下，主从延迟基本都是毫秒级别，你可以认为主从就是实时保持同步的。麻烦的是不正常的情况，一旦主库或者从库繁忙的时候，有可能会出现明显的主从延迟。

而很多情况下，数据库都不是突然宕机的，而是先繁忙，性能下降，最终宕机。这种情况下，很有可能主从延迟很大，如果我们把业务直接切到从库上继续读写，主从延迟这部分数据就丢了，并且这个数据丢失是不可逆的。即使事后你找回了当时主库的Binlog也是没法做到自动恢复的，因为它和从库的数据是冲突的。

简单地说，如果主库宕机并且主从存在延迟的情况下，切换到从库继续读写，可以保证业务的可用性，但是主从延迟这部分数据就丢失了。

这个时候你就需要做一个选择题了，第一个选项是，保证不丢数据，牺牲可用性，暂时停止服务，想办法把主库的Binlog恢复到从库上之后再提供服务。第二个选项就是，冒着丢一些数据的风险，保证可用性，第一时间切换到从库继续提供服务。

那能不能既保证数据不丢，还能做到高可用呢？也是可以的，那你就要牺牲一些性能。MySQL也支持同步

复制，开启同步复制时，MySQL主库会等待数据成功复制到从库之后，再给客户端返回响应。

如果说，牺牲的这点儿性能我不在乎，这个方案是不是就完美了呢？也不是，新的问题又来了！你想一下，这种情况下从库宕机了怎么办？本来从库宕机对主库是完全没影响的，因为现在主库要等待从库写入成功再返回，从库宕机，主库就会一直等待从库，主库也卡死了。

这个问题也有解决办法，那就是再加一个从库，把主库配置成：成功复制到任意一个从库就返回，只要有一个从库还活着，就不会影响主库写入数据，这样就解决了从库宕机阻塞主库的问题。如果主库发生宕机，在两个从库中，至少有一个从库中的数据是和主库完全一样的，可以把这个库作为新的主库，继续提供服务。为此你需要付出的代价是，你要至少用三台数据库服务器，并且这三台服务器提供的服务性能，还不如一台服务器高。

我把上面这三种典型的HA方案总结成下面这个表格，便于你对比选择：

方案	高可用	可能丢数据	性能
一主一从 异步复制，手动切换	否	可控	好
一主一从 异步复制，自动切换	是	是	好
一主二从 同步复制，自动切换	是	否	差

小结

今天这节课讲了两件事儿，一是如何备份和恢复数据库中的数据，确保数据安全；二是如何来实现数据库的高可用，避免宕机停服。

虽然这是两个不同的问题，但你要知道，解决这两个问题背后的实现原理是一样的。**高可用依赖的是数据复制，数据复制的本质就是从一个库备份数据，然后恢复到另外一个库中去。**

数据备份时，使用低频度的全量备份配合Binlog增量备份是一种常用而且非常实用的方法，使用这种备份方法，我们可以把数据库的数据精确地恢复到历史上任意一个时刻，不仅能解决数据损坏的问题，也不用怕误操作、删库跑路这些事儿了。特别要注意的是，让备份数据尽量地远离数据库。

我们今天讲到的几种MySQL典型的HA方案，在数据可靠性、数据库可用性、性能和成本几个方面，各有利弊，你需要根据业务情况，做一个最优的选择，并且为可能存在的风险做好准备。

思考题

课后也请你在留言区分享一下，你现在负责系统的数据库是如何来实现高可用的，有什么风险和问题，学习了这节课之后，你会如何来改进这个高可用方案？欢迎你在留言区与我讨论。

感谢阅读，如果你觉得今天的内容对你有帮助，也欢迎把它分享给你的朋友。

精选留言：

- 李玥 2020-03-12 16:25:08
  Hi，我是李玥。

  照例说一下上节课思考题：

  我们在电商的搜索框中搜索商品时，它都有一个搜索提示的功能，比如我输入“苹果”还没有点击搜索按钮的时候，搜索框下面会提示“苹果手机”、“苹果11、苹果电脑”这些建议的搜索关键字，请你课后看一下ES的文档，想一下，如何用ES快速地实现这个搜索提示功能？

  在课后留言中，Geek_c76e2d同学给出的答案非常赞，我在这里就直接“盗用”了，以下是Geek_c76e2d同学的答案：

  因为用户每输入一个字都可能会发请求查询搜索框中的搜索推荐。所以搜索推荐的请求量远高于搜索框中的搜索。es针对这种情况提供了suggestion api，并提供的专门的数据结构应对搜索推荐，性能高于match，但它应用起来也有局限性，就是只能做前缀匹配。再结合pinyin分词器可以做到输入拼音字母就提示中文。如果想做非前缀匹配，可以考虑Ngram。不过Ngram有些复杂，需要开发者自定义分析器。比如有个网址www.geekbang.com，用户可能记不清具体网址了，只记得网址中有2个e，此时用户输入ee两个字母也是可以在搜索框提示出这个网址的。以上是我在工作中针对前缀搜索推荐和非前缀搜索推荐的实现方案。

- skyline 2020-03-12 01:04:13
  除了技术方面，我觉得删库跑路也是一个管理机制上的问题，要当成不可抗因素去对待。

  为防止地震我们需要异地备份，距离越远越好，为防止跑路我们需要完善的权限管理。

  不能让一个人有能接触到所有备份的权限，否则就跟单机故障一样出现"单人故障"😷 [3赞]

- 滴流乱转小胖子 2020-03-12 08:53:43
  老师好，开篇的某云场景叙述，好皮啊！优秀！ [1赞]

- qbit 2020-03-12 15:00:45
  SHOW VARIABLES LIKE '%log_bin%';

  "log_bin" "OFF"
  "log_bin_basename" ""
  "log_bin_index" ""
  "log_bin_trust_function_creators" "OFF"
  "sql_log_bin" "ON"

  请问 sql_log_bin 和 log_bin 有什么区别和联系？

- qbit 2020-03-12 14:05:40
  腾讯云是一家著名的云服务商:-D

- 刘楠 2020-03-12 09:05:39
  binlog 日志中也是有删除库的SQL的，难道，备库或者从库不会执行吗？感觉会执行，所以数据在几个库都删除了。怎么保证备库或者从的数据？

- 发条橙子 。 2020-03-12 07:40:47
  老师 binlog中不会包含删除表的那行记录么 还是说虽然包含 但是我们可以不去执行那条命令