分库设计中的主键选择

April 11, 2011

在先前的文章《又拍网架构中的分库设计》中，我有提到过MySQL分库设计中的主键选择问题。在这篇文章里我想对这个问题进行展开讨论，以此作为对上一篇文章的一个补充。

前面提到又拍网采用了全局唯一的字段作为主键。比如拿照片表为例，虽然不同用户的照片数据存放在不同的Shard（或者说MySQL节点/实例, 请参考《又拍网架构中的分库设计》）上，但是每一张照片拥有整个站点唯一的ID作为标示。

为什么要全局唯一？

我们在对数据库集群作扩容时，为了保证负载的平衡，需要在不同的Shard之间进行数据的移动，如果主键不唯一，我们就没办法这样随意的移动数据。起初，我们考虑采用组合主键来解决这个问题。一般会以`user_id`和一个自增的`photo_id`来作为主键，这的确能解决移动数据可能带来的主键冲突问题，但是就像在“又拍网架构中的分库设计”中描述的那样当Shard之间的数据发生关系后，我们需要用更多的字段来组成主键以保证唯一性，因此主键的索引会变的很大，从而影响查询性能，同时也会影响写入性能。

其次，每个Shard由两台MySQL服务器组成，而这两台服务器采用master-master的复制方式，以保证每个Shard一直可写。master-master复制方式必须保证在两台服务器上各自插入的数据有不同的主键，不然当复制到另外一台时就会出现主键重复错误。如果我们保证主键全局唯一，就自然的解决了这个问题。在没有采用数据拆分的设计当中，如果要用自增字段，可以参考这篇文章里的解决办法。

可能的解决方案

- UUID

或许可以采用UUID作为主键，但是UUID好长的一串，放在URL里好难看啊，有木有?

当然这个不是关键所在，更重要的原因还是性能。UUID的生成没有顺序性，所以在写入时， 需要随机更改索引的不同位置，这就需要更多的IO操作，如果索引太大而不能存放在内存中的话就更是如此。 而UUID索引时，一个key需要32个字节(当然如果采用二进制形式存储的话可以压缩到16个字节)， 因此整个索引也会相对比较大。

- MySQL自增字段

在单个MySQL数据库的应用中一般设置一个自增的字段就可以了，而在水平分库的设计当中，这种方法显然不能保证全局唯一。 那么我们可以单独建立一个库用来生成ID，在Shard中的每张表在这个ID库中都有一个对应的表，而这个对应的表只有一个字段，这个字段是自增的。当我们需要插入新的数据，我们首先在ID库中的相应表中插入一条记录，以此得到一个新的ID， 然后将这个ID作为插入到Shard中的数据的主键。这个方法的缺点就是需要额外的插入操作，如果ID库变的很大， 性能也会随之降低。所以一定要保证ID库的数据集不要太大，一个办法是定期清理前面的记录。

- 引入其它工具

Redis、Memcached等都支持原子性的increment操作，而且因为它们的优秀性能可以减少写入时的额外开销， 也许我们可以拿它们当作序列生成器。Memcached的问题在于不持久性，所以我们不会考虑。 而Redis也不是实时持久的，当然也可以配置成实时的，但那样怪怪的。当然也有一些持久的工具， 比如Kyoto Cabinet、Tokyo Cabinet、MongoDB等等，传说中性能都不错，但是引入其它工具会增加架构的复杂程度， 也会增加维护成本。我们的团队很小，精力有限，我们奉行够用就好的原则，也就是没有特别的原因， 在可以接受的情况下，尽量用我们熟悉的工具解决问题。所以，我们还是来考虑一下怎么样用MySQL来解决这个问题吧。

更好的方案

我们一开始就是采用了上面所描述的MySQL自增字段的方法， 后来看到《Ticket Servers: Distributed Unique Primary Keys on the Cheap》 这篇文章里所描述的方法，豁然开朗。我经常这样想：如果没有那些开源产品、没有那些无私分享经验的人， 光凭我们自己的能力能做到什么程度。很感谢那些人，所以我也尽量多的分享一些自己的经验。

我先描述一下Flickr那篇文章里所描述的方法，他们使用了REPLACE INTO这个MySQL的扩展功能。 REPLACE INTO和INSERT的功能一样，但是当使用REPLACE

INTO插入新数据行时， 如果新插入的行的主键或唯一键(UNIQUE Key)已有的行重复时，已有的行会先被删除，然后再将新数据行插入。 你可以放心，这是原子操作。

建立类似下面的表:

```
CREATE TABLE `tickets64` (
    `id` bigint(20) unsigned NOT NULL auto_increment,
    `stub` char(1) NOT NULL default '',
    PRIMARY KEY  (`id`),
    UNIQUE KEY `stub` (`stub`)
) ENGINE=MyISAM;
```

当需要获得全局唯一ID时，执行下面的SQL语句:

```
REPLACE INTO `tickets64` (`stub`) VALUES ('a');
SELECT LAST_INSERT_ID();
```

第一次执行这个语句后，ticket64表将包含以下数据:

```
+--------+------+
| id     | stub |
+--------+------+
| 1      |    a |
+--------+------+
```

以后再次执行前面的语句，stub字段值为'a'的行已经存在，所以MySQL会先删除这一行，再插入。 因此，第二次执行后，ticket64表还是只有一行数据，只是id字段的值为2。 这个表将一直只有一行数据。

Flickr为Photo, Group, Account, Task各自建立了一张ticket表以保持各自的ID的连续性。其它业务表的ID都使用同一个ticket表产生。

不错吧，其实还可以更棒。比如，只需要一张ticket表就可以为所有的业务表提供各自连续的ID。 下面，来看一下我们的方法。首先来看一下表结构:

```
CREATE TABLE `sequence` (
    `name` varchar(50) NOT NULL,
    `id` bigint(20) unsigned NOT NULL DEFAULT '0',
    PRIMARY KEY (`name`)
) ENGINE=InnoDB;
```

注意区别，id字段不是自增的，也不是主键。在使用前，我们需要先插入一些初始化数据:

```
INSERT INTO `sequence` (`name`) VALUES
('users'), ('photos'), ('albums'), ('comments');
```

接下来，我们可以通过执行下面的SQL语句来获得新的照片ID：

```
UPDATE `sequence` SET `id` = LAST_INSERT_ID(`id` + 1) WHERE `name` =
'photos';
SELECT LAST_INSERT_ID();
```

我们执行了一个更新操作，将id字段增加1，并将增加后的值传递到LAST_INSERT_ID函数，从而指定了LAST_INSERT_ID的返回值。

实际上，我们不一定需要预先指定序列的名字。如果我们现在需要一种新的序列，我们可以直接执行下面的SQL语句：

```
INSERT INTO `sequence` (`name`) VALUES('new_business') ON DUPLICATE
KEY UPDATE `id` = LAST_INSERT_ID(`id` + 1);
SELECT LAST_INSERT_ID();
```

这里，我们采用了INSERT ... ON DUPLICATE KEY UPDATE这个MySQL扩展，这个扩展的功能也和INSERT一样插入一行新的记录，但是当新插入的行的主键或唯一键(UNIQUE Key)和已有的行重复时，会对已有行进行UPDATE操作。

需要注意的是，当我们第一次执行上面的语句时，因为还没有name为'new_business'的字段，所以正常的执行了插入操作，没有执行UPDATE，所以也没有为LAST_INSERT_ID传递值。所以之后执行SELECT LAST_INSERT_ID()返回的值不可确定，要看当前连接在此之前执行过什么操作，如果没有执行过会影响LAST_INSERT_ID值的操作，那么返回值将是0，不然就是该操作产生的值。所以，我们应该尽量避免使用这种方式。

UPDATE: 这个方法更容易解决单点问题，也不局限于两个服务器，只要对不同的服务器设置不同的初始值（但必须是连续的），然后将增量变为服务器数就行了。

总结一下

我还是那句话，够用就好。当然，也不是说就不要去了解其它产品、方案了。又拍网也在使用一些新兴的产品，比如Redis（在10年3月就开始在正式环境下使用了，算是比较早的使用者），因为它的引入的确能够更好、更方便、更高效的解决我们的某些问题。

关键还是需要在使用前对其进行足够的了解。我会在后面的文章中介绍一下Redis的使用情况。

所以建议还是事先初始化好要生成的序列名称和初始值。

11 月前 in reply to BeerBubble 喜欢 回复

国恺 韩

这个bug最好原文更新一下，因为第一次生成的时候 LAST_INSERT_ID() 返回了上次别的类型的最后id，而不是从0开始的。而之后这个类型的id生成到第一个错误的 id 的时候就会导致生成的主键不唯一，最终出现主键重复错误！

我今天就遇到了这个错误，调试了半天才明白是这里的问题。所以想过来指出，原来前几天就有人指出了，可惜没来看评论，呵呵。

11 月前 in reply to zolazhou 喜欢 回复

zolazhou

好的，已更新

11 月前 in reply to 国恺 韩 喜欢 回复

tianshengpan alex

UPDATE `sequence` SET `id` = LAST_INSERT_ID(`id` + 1) WHERE `name` = 'photos';
SELECT LAST_INSERT_ID();

这是一个原子操作?

11 月前 喜欢 回复

zolazhou

UPDATE 是原子操作，
后面的SELECT加一起就不是了，不过没有关系，在同一个连接里，只要中间不执行会导致LAST_INSERT_ID值变更的操作，不管是什么时候SELECT LAST_INSERT_ID() 返回的都是刚才的值。

11 月前 in reply to tianshengpan alex 喜欢 回复

tianshengpan alex

在多个连接中，并发的时候，是不是就会导致SELECT LAST_INSERT_ID();返回的是同一个值了

11 月前 in reply to zolazhou 喜欢 回复

zolazhou

不会，其它连接不会影响当前连接的LAST_INSERT_ID

11 月前 in reply to tianshengpan alex 喜欢 回复

tianshengpan alex

SELECT LAST_INSERT_ID(); 只是跟当前的连接有关系？

11 月前 in reply to zolazhou 喜欢 回复

zolazhou

是的

11 月前 in reply to tianshengpan alex 喜欢 回复

Abioy Sun

但这样你就要保证每次NEW一个ID就要新起一个连接？另外如果不加其他保护的话，是有单点风险的。flickr有避免单点的方法，也可以一起参考。用ttserver之类的incr操作还是挺方便的，这个维护成长作为长远的设计来讲应该是值得。

11 月前 in reply to zolazhou 喜欢 回复

zolazhou

一般情况下，在同步应用（非异步的，或者说基于线程/进程模型的应用），处理一个业务请求时不管新建连接（如果是PHP,建议新建连接）或者重用连接，都不会有问题。

至于单点问题，我描述的方法更容易解决，只要在不同的服务器上变一下初始值和增量就行了。

引入其它工具在管理上带来的麻烦是不这样做的一方面原因，另一个原因是我们需要将复杂的ID逻辑封装到我们开发框架里，我们不希望这个框架依赖其它工具。

11 月前 喜欢 回复
in reply to Abioy Sun

Ramd Wang

你好，zolazhou：
这个问题在php这种线程模型中

确实没有问题，但是在java环境下，由于是多线程的，同一个连接可能有多个并发访问操作，会不会造成一些隐患？例如SELECT LAST_INSERT_ID();得到的并不是上次执行update操作的id

10 月前　喜欢　回复
in reply to zolazhou

zolazhou

不需要担心这个问题，我在例子里面用的都是SQL，分成两个语句执行，所以会有这样的顾虑。实际上当我们使用客户端实现时，这个ID会在执行INSERT/UPDATE语句的返回包中就包含，客户端实现通常解析这个包后就将这个值保持在连接对象里。也就是说当你在调用mysql_insert_id()时（C客户端的方法，PHP基于它），是不需要再请求服务器的。Java的客户端也一样。

10 月前　喜欢　回复
in reply to Ram Wan

Ramd...

我还不是很明白。mysql_insert_id()如果说是把last_insert_id放在了返回值

里，那么如果我在java里执行这两句sql又是什么情况呢？
因为这个last_insert_id毕竟是上一句sql赋值才能得到，所以必须使用这两句sql才能完成操作呀。

喜欢 回复

zol

不，一个MySQL Command就可以了，当执行了INSERT 或UPDATE语句，MySQL的返回包里就包含了这个ID。请参考http://forge.mysql.com/

wiki/My...

喜欢 回复

Liukaixuan

lilo?

11 月前 喜欢 回复

anhong wang

这个方法很巧，

11 月前 喜欢 回复

Iamxjb

学习

11 月前 喜欢 回复

chinalu

Good

11 月前 喜欢 回复

M 通过邮件订阅 S RSS