



下载APP



05 | 主键：如何正确设置主键？

2021-03-18 朱晓峰

MySQL 必知必会

[进入课程 >](#)**讲述：朱晓峰**

时长 12:18 大小 11.28M



你好，我是朱晓峰，今天我们来聊一聊如何用好 MySQL 的主键。

前面在讲存储的时候，我提到过主键，它可以唯一标识表中的某一条记录，对数据表来说非常重要。当我们需要查询和引用表中的一条记录的时候，最好的办法就是通过主键。只有合理地设置主键，才能确保我们准确、快速地找到所需要的数据记录。今天我就借助咱们的超市项目的实际需求，来给你讲一讲怎么正确设置主键。

在我们的项目中，客户要进行会员营销，相应的，我们就需要处理会员信息。会员信息表（demo.membermaster）的设计大体如下：



cardno (卡号)	membername (名称)	memberphone (电话)	memberpid (身份证号)	address (地址)	sex (性别)	birthday (生日)
10000001	张三	13812345678	110123200001017890	北京	男	2000-01-01
10000002	李四	13512312312	123123199001012356	上海	女	1990-01-01

为了能够唯一地标识一个会员的信息，我们需要为会员信息表设置一个主键。那么，怎么为这个表设置主键，才能达到我们理想的目标呢？


今天，我就带你在解决这个实际问题的过程中，学习下三种设置主键的思路：**业务字段做主键**、**自增字段做主键**和**手动赋值字段做主键**。

业务字段做主键

针对这个需求，我们最容易想到的，是选择表中已有的字段，也就是跟业务相关的字段做主键。那么，在这个表里，哪个字段比较合适呢？我们来分析一下。


会员卡号（cardno）看起来比较合适，因为会员卡号不能为空，而且有唯一性，可以用来标识一条会员记录。我们来尝试一下用会员卡号做主键。

我们可以用下面的代码，在创建表的时候，设置字段 cardno 为主键：

 复制代码

```
1 mysql> CREATE TABLE demo.membermaster
2 -> (
3 -> cardno CHAR(8) PRIMARY KEY, -- 会员卡号为主键
4 -> membername TEXT,
5 -> memberphone TEXT,
6 -> memberpid TEXT,
7 -> memberaddress TEXT,
8 -> sex TEXT,
9 -> birthday DATETIME
10 -> );
11 Query OK, 0 rows affected (0.06 sec)
```

我们来查询一下表的结构，确认下主键是否创建成功了：

 复制代码


```

1 mysql> DESCRIBE demo.membermaster;
2 +-----+-----+-----+-----+-----+-----+
3 | Field | Type | Null | Key | Default | Extra |
4 +-----+-----+-----+-----+-----+-----+
5 | cardno | char(8) | NO | PRI | NULL | |
6 | membername | text | YES | | NULL | |
7 | memberphone | text | YES | | NULL | |
8 | memberpid | text | YES | | NULL | |
9 | memberaddress | text | YES | | NULL | |
10 | sex | text | YES | | NULL | |
11 | birthday | datetime | YES | | NULL | |
12 +-----+-----+-----+-----+-----+-----+
13 7 rows in set (0.02 sec)

```

可以看到，字段 cardno 在表示键值的 key 这一列的值是“PRI”，意思是 PRIMARY KEY，这就表示它已经被设置成主键了。这里需要注意的一点是，除了字段 cardno，所有的字段都允许为空。这是因为，这些信息有可能当时不知道，要稍后补齐。

会员卡号做主键有没有什么问题呢？我们插入 2 条数据来验证下：

 复制代码


```

1 mysql> INSERT INTO demo.membermaster
2 -> (
3 -> cardno,
4 -> membername,
5 -> memberphone,
6 -> memberpid,
7 -> memberaddress,
8 -> sex,
9 -> birthday
10 -> )
11 -> VALUES
12 -> (
13 -> '10000001',
14 -> '张三',
15 -> '13812345678',
16 -> '110123200001017890',
17 -> '北京',
18 -> '男',
19 -> '2000-01-01'
20 -> );
21 Query OK, 1 row affected (0.01 sec)
22
23 mysql> INSERT INTO demo.membermaster
24 -> (
25 -> cardno,

```

```
26 -> membername,  
27 -> memberphone,  
28 -> memberpid,  
29 -> memberaddress,  
30 -> sex,  
31 -> birthday  
32 -> )  
33 -> VALUES  
34 -> (  
35 -> '100000002',  
36 -> '李四',  
37 -> '13512345678',  
38 -> '123123199001012356',  
39 -> '上海',  
40 -> '女',  
41 -> '1990-01-01'  
42 -> );  
43 Query OK 1 row affected (0.01 sec)
```

插入成功后，我们来看一下表的内容：

 复制代码

```
1 mysql> SELECT *  
2     -> FROM demo.membermaster;  
3 +-----+-----+-----+-----+-----+-----+  
4 | cardno | membername | memberphone | memberpid | memberaddress | s  
5 +-----+-----+-----+-----+-----+-----+  
6 | 10000001 | 张三      | 13812345678 | 110123200001017890 | 北京          | 男  
7 | 10000002 | 李四      | 13512345678 | 123123199001012356 | 上海          | 女  
8 +-----+-----+-----+-----+-----+-----+  
9 2 rows in set (0.00 sec)
```

我们发现，不同的会员卡号对应不同的会员，字段“cardno”唯一地标识某一个会员。如果都是这样，会员卡号与会员一一对应，系统是可以正常运行的。

但是实际情况是，上线不到一周，就发生了“cardno”无法唯一识别某一个会员的问题。原来，会员卡号存在重复使用的情况。

这也很好理解，比如，张三因为工作变动搬离了原来的地址，不再到商家的门店消费了（退还了会员卡），于是张三就不再是这个商家门店的会员了。但是，商家不想让这个会员卡空着，就把卡号是“10000001”的会员卡发给了王五。

从系统设计的角度看，这个变化只是修改了会员信息表中的卡号是“10000001”这个会员信息，并不会影响到数据一致性。也就是说，修改会员卡号是“10000001”的会员信息，系统的各个模块，都会获取到修改后的会员信息，不会出现“有的模块获取到修改之前的会员信息，有的模块获取到修改后的会员信息，而导致系统内部数据不一致”的情况。因此，从信息系统层面上看是没问题的。但是从使用系统的业务层面来看，就有很大的问题了，会对商家造成影响。

下面，我们就来看看这种修改，是如何影响到商家的。

比如，我们有一个销售流水表，记录了所有的销售流水明细。2020 年 12 月 01 日，张三在门店购买了一本书，消费了 89 元。那么，系统中就有了张三买书的流水记录，如下所示：

transactionno (流水单号)	itemnumber (商品编号)	quantity (销售数量)	price (价格)	salesvalue (销售金额)	cardno (会员卡号)	transdate (交易时间)
1	1	1	89	89	10000001	2020-12-01

我们可以用下面的代码创建销售流水表。因为需要引用会员信息和商品信息，所以表中要包括商品编号字段和会员卡号字段。

[复制代码](#)

```
1 mysql> CREATE table demo.trans
2 -> (
3 -> transactionno INT,
4 -> itemnumber INT, -- 为了引用商品信息
5 -> quantity DECIMAL(10,3),
6 -> price DECIMAL(10,2),
7 -> salesvalue DECIMAL(10,2),
8 -> cardno CHAR(8), -- 为了引用会员信息
9 -> transdate DATETIME
10 -> );
11 Query OK, 0 rows affected (0.10 sec)
```


创建好表以后，我们来插入一条销售流水：

[复制代码](#)

```
1 mysql> INSERT INTO demo.trans
```

```
2 -> (  
3 -> transactionno,  
4 -> itemnumber,  
5 -> quantity,  
6 -> price,  
7 -> salesvalue,  
8 -> cardno,  
9 -> transdate  
10 -> )  
11 -> VALUES  
12 -> (  
13 -> 1,  
14 -> 1,  
15 -> 1,  
16 -> 89,  
17 -> 89,  
18 -> '10000001',  
19 -> '2020-12-01'  
20 -> );  
21 Query OK, 1 row affected (0.01 sec)
```

接着，我们查询一下 2020 年 12 月 01 日的会员销售记录：


 复制代码

```
1 mysql> SELECT b.membername,c.goodsname,a.quantity,a.salesvalue,a.transdate  
2 -> FROM demo.trans AS a  
3 -> JOIN demo.membermaster AS b  
4 -> JOIN demo.goodsmaster AS c  
5 -> ON (a.cardno = b.cardno AND a.itemnumber=c.itemnumber);  
6 +-----+-----+-----+-----+-----+  
7 | membername | goodsname | quantity | salesvalue | transdate |  
8 +-----+-----+-----+-----+-----+  
9 | 张三 | 书 | 1.000 | 89.00 | 2020-12-01 00:00:00 |  
10 +-----+-----+-----+-----+-----+  
11 1 row in set (0.00 sec)
```

我们得到的查询结果是：张三，在 2020 年 12 月 01 日买了一本书，花了 89 元。


需要注意的是，这里我用到了 JOIN，也就是表的关联，目的是为了引用其他表的信息，包括会员信息表（demo.membermaster）和商品信息表（demo.goodsmaster）。有关关联表查询的具体细节，我会在下节课讲到，这里你只要知道，通过关联查询，可以从会员信息表中获取会员信息，从商品信息表中获取商品信息，就可以了。

下面，我们假设会员卡“10000001”又发给了王五，我们需要更改会员信息表：

 复制代码

```
1 mysql> UPDATE demo.membermaster
2 -> SET membername = '王五',
3 -> memberphone = '13698765432',
4 -> memberpid = '475145197001012356',
5 -> memberaddress='天津',
6 -> sex='女',
7 -> birthday = '1970-01-01'
8 -> WHERE cardno = '10000001';
9 Query OK, 1 row affected (0.02 sec)
10 Rows matched: 1 Changed: 1 Warnings: 0
```

会员记录改好了，我们再次运行之前的会员消费流水查询：

 复制代码

```
1 mysql> SELECT b.membername,c.goodsname,a.quantity,a.salesvalue,a.transdate
2 -> FROM demo.trans AS a
3 -> JOIN demo.membermaster AS b
4 -> JOIN demo.goodsmaster AS c
5 -> ON (a.cardno = b.cardno AND a.itemnumber=c.itemnumber);
6 +-----+-----+-----+-----+-----+
7 | membername | goodsname | quantity | salesvalue | transdate |
8 +-----+-----+-----+-----+-----+
9 | 王五 | 书 | 1.000 | 89.00 | 2020-12-01 00:00:00 |
10 +-----+-----+-----+-----+-----+
11 1 row in set (0.01 sec)
```

这次得到的结果是：王五在 2020 年 12 月 01 日，买了一本书，消费 89 元。

很明显，这个结果把张三的消费行为放到王五身上去了，肯定是不对的。这里的原因就是，我们把会员卡号是“10000001”的会员信息改了，而会员卡号是主键，会员消费查询通过会员卡号关联到会员信息，得到了完全错误的结果。

现在你知道了吧，千万不能把会员卡号当做主键。

那么，会员电话可以做主键吗？不行的。在实际操作中，手机号也存在被运营商收回，重新发给别人用的情况。

那身份证号行不行呢？好像可以。因为身份证决不会重复，身份证号与一个人存在一一对应的关系。可问题是，身份证号属于个人隐私，顾客不一定愿意给你。对门店来说，顾客就是上帝，要是强制要求会员必须登记身份证号，会把很多客人赶跑的。其实，客户电话也有这个问题，这也是我们在设计会员信息表的时候，允许身份证号和电话都为空的原因。

这样看来，任何一个现有的字段都不适合做主键。

所以，我建议你**尽量不要用业务字段，也就是跟业务有关的字段做主键**。毕竟，作为项目设计的技术人员，我们谁也无法预测在项目的整个生命周期中，哪个业务字段会因为项目的业务需求而有重复，或者重用之类的情況出现。


既然业务字段不可以，那我们再来试试自增字段。

使用自增字段做主键

我们来给会员信息表添加一个字段，比如叫 id，给这个字段定义自增约束，这样，我们就有了一个具备唯一性的，而且不为空的字段来做主键了。


接下来，我们就来修改一下会员信息表的结构，添加一个自增字段做主键。

第一步，修改会员信息表，删除表的主键约束，这样，原来的主键字段，就不再是主键了。不过需要注意的是，删除主键约束，并不会删除字段。

 复制代码

```
1 mysql> ALTER TABLE demo.membermaster
2 -> DROP PRIMARY KEY;
3 Query OK, 2 rows affected (0.12 sec)
4 Records: 2 Duplicates: 0 Warnings: 0
```

第二步，修改会员信息表，添加字段“id”为主键，并且给它定义自增约束：

 复制代码

```
1 mysql> ALTER TABLE demo.membermaster
2 -> ADD id INT PRIMARY KEY AUTO_INCREMENT;
3 Query OK, 0 rows affected (0.12 sec)
4 Records: 0 Duplicates: 0 Warnings: 0
```


第三步，修改销售流水表，添加新的字段 `memberid`，对应会员信息表中的主键：

[复制代码](#)

```
1 mysql> ALTER TABLE demo.trans
2 -> ADD memberid INT;
3 Query OK, 0 rows affected (0.04 sec)
4 Records: 0 Duplicates: 0 Warnings: 0
```

第四步，我们更新一下销售流水表，给新添加的字段 “`memberid`” 赋值，让它指向对应的会员信息：

[复制代码](#)

```
1 mysql> UPDATE demo.trans AS a,demo.membermaster AS b
2 -> SET a.memberid=b.id
3 -> WHERE a.transactionno > 0
4 --> AND a.cardno = b.cardno; -- 这样操作可以不用删除trans的内容，在实际工作中更适合
5 Query OK, 1 row affected (0.01 sec)
6 Rows matched: 1 Changed: 1 Warnings: 0
```

这个更新语句包含了 2 个关联的表，看上去有点复杂。其实，你完全可以通过删除表 `demo.trans`、重建表，再插入一条数据的操作，来达到同样的目的，但是我不建议你这么做。

在实际操作中，你不一定能删掉 `demo.trans` 这个表，因为这个表里面可能已经有了很多重要的数据。所以，你一定要认真学习一下我给你介绍的这个更新数据的方法，这种复杂一点的更新语句在实战中更有用。

好了，到这里，我们就完成了数据表的重新设计，让我们看一下新的数据表 `demo.membermaster` 和 `demo.trans` 的结构：

[复制代码](#)

```
1 mysql> DESCRIBE demo.membermaster;
2 +-----+-----+-----+-----+-----+-----+
3 | Field          | Type      | Null  | Key  | Default | Extra |
4 +-----+-----+-----+-----+-----+-----+
5 | cardno         | char(8)   | NO    |      | NULL    |       |
```

```

6 | membername | text | YES | | NULL | |
7 | memberphone | text | YES | | NULL | |
8 | memberpid | text | YES | | NULL | |
9 | memberaddress | text | YES | | NULL | |
10 | sex | text | YES | | NULL | |
11 | birthday | datetime | YES | | NULL | |
12 | id | int | NO | PRI | NULL | auto_increment |
13 +-----+-----+-----+-----+-----+
14 8 rows in set (0.02 sec)
15
16 mysql> DESCRIBE demo.trans;
17 +-----+-----+-----+-----+-----+
18 | Field | Type | Null | Key | Default | Extra |
19 +-----+-----+-----+-----+-----+
20 | transactionno | int | NO | PRI | NULL | |
21 | itemnumber | int | YES | | NULL | |
22 | quantity | decimal(10,3) | YES | | NULL | |
23 | price | decimal(10,2) | YES | | NULL | |
24 | salesvalue | decimal(10,2) | YES | | NULL | |
25 | cardno | char(8) | YES | | NULL | |
26 | transdate | datetime | YES | | NULL | |
27 | memberid | int | YES | | NULL | |
28 +-----+-----+-----+-----+-----+
29 8 rows in set (0.00 sec)

```

现在，如果我们再次面对卡号重用的情况，该如何应对呢（这里我们假设回到修改会员卡 10000001 为王五之前的状态）？

如果张三的会员卡 “10000001” 不再使用，发给了王五，我们就在会员信息表里面增加一条记录：

```

1 mysql> INSERT INTO demo.membermaster
2 -> (
3 -> cardno,
4 -> membername,
5 -> memberphone,
6 -> memberpid,
7 -> memberaddress,
8 -> sex,
9 -> birthday
10 -> )
11 -> VALUES
12 -> (
13 -> '10000001',
14 -> '王五',
15 -> '13698765432',

```


 复制代码

```

16 -> '475145197001012356',
17 -> '天津',
18 -> '女',
19 -> '1970-01-01'
20 -> );
21 Query OK, 1 row affected (0.02 sec)

```

下面我们看看现在的会员信息：

 复制代码


```

1 mysql> SELECT *
2     -> FROM demo.membermaster;
3 +-----+-----+-----+-----+-----+-----+
4 | cardno   | membername | memberphone | memberpid       | memberaddress | s |
5 +-----+-----+-----+-----+-----+-----+
6 | 10000001 | 张三       | 13812345678 | 110123200001017890 | 北京         | 男 |
7 | 10000002 | 李四       | 13512345678 | 123123199001012356 | 上海         | 女 |
8 | 10000001 | 王五       | 13698765432 | 475145197001012356 | 天津         | 女 |
9 +-----+-----+-----+-----+-----+-----+
10 3 rows in set (0.00 sec)

```

由于字段“cardno”不再是主键，可以允许重复，因此，我们可以在保留会员“张三”信息的同时，添加使用同一会员卡号的“王五”的信息。

现在再来查会员消费，就不会出问题了：

 复制代码

```

1 mysql> SELECT b.membername,c.goodsname,a.quantity,a.salesvalue,a.transdate
2     -> FROM demo.trans AS a
3     -> JOIN demo.membermaster AS b
4     -> JOIN demo.goodsmaster AS c
5     -> ON (a.memberid = b.id AND a.itemnumber=c.itemnumber);
6 +-----+-----+-----+-----+-----+-----+
7 | membername | goodsname | quantity | salesvalue | transdate |
8 +-----+-----+-----+-----+-----+-----+
9 | 张三       | 书       | 1.000    | 89.00      | 2020-12-01 00:00:00 |
10 +-----+-----+-----+-----+-----+-----+
11 1 row in set (0.01 sec)

```

可以看到，结果是 2020 年 12 月 01 日，张三买了一本书，消费 89 元，是正确的。

如果是一个小项目，只有一个 MySQL 数据库服务器，用添加自增字段作为主键的办法是可以的。不过，这并不意味着，在任何情况下你都可以这么做。

举个例子，用户要求把增加新会员的工作放到门店进行，因为发展新会员的工作一般是在门店进行的，毕竟，人们一般都是在购物的同时申请会员。解决的办法是，门店的信息系统添加新增会员的功能，把新的会员信息先存放到本地 MySQL 数据库中，再上传到总部，进行汇总。

可是问题来了，如果会员信息表的主键是自增的，那么各个门店新加的会员就会出现 “id” 冲突的可能。

比如，A 店的 MySQL 数据库中的 demo.membermaster 中，字段 “id” 的值是 100，这个时候，新增了一个会员，“id” 是 101。同时，B 店的字段 “id” 值也是 100，要加一个新会员，“id” 也是 101，毕竟，B 店的 MySQL 数据库与 A 店相互独立。等 A 店与 B 店都把新的会员上传到总部之后，就会出现两个 “id” 是 101，但却是不同会员的情况，这该如何处理呢？

手动赋值字段做主键

为了解决这个问题，我们想了一个办法：取消字段 “id” 的自增属性，改成信息系统在添加会员的时候对 “id” 进行赋值。

具体的操作是这样的：在总部 MySQL 数据库中，有一个管理信息表，里面的信息包括成本核算策略，支付方式等，还有总部的系统参数，我们可以在这个表中添加一个字段，专门用来记录当前会员编号的最大值。

门店在添加会员的时候，先到总部 MySQL 数据库中获取这个最大值，在这个基础上加 1，然后用这个值作为新会员的 “id”，同时，更新总部 MySQL 数据库管理信息表中的当前会员编号的最大值。

这样一来，各个门店添加会员的时候，都对同一个总部 MySQL 数据库中的数据表字段进行操作，就解决了各门店添加会员时会员编号冲突的问题，同时也避免了使用业务字段导致数据错误的问题。

总结

今天，我给你介绍了设置数据表主键的三种方式：数据表的业务字段做主键、添加自增字段做主键，以及添加手动赋值字段做主键。

用业务字段做主键，看起来很简单，但是我们应该尽量避免这样做。因为我们无法预测未来会不会因为业务需要，而出现业务字段重复或者重用的情况。

自增字段做主键，对于单机系统来说是没问题的。但是，如果有多台服务器，各自都可以录入数据，那就不一定适用了。因为如果每台机器各自产生的数据需要合并，就可能会出现主键重复的问题。

我们可以采用手动赋值的办法，通过一定的逻辑，确保字段值在全系统的唯一性，这样就可以规避主键重复的问题了。

刚开始使用 MySQL 时，很多人都很容易犯的错误是喜欢用业务字段做主键，想当然地认为了解业务需求，但实际情况往往出乎意料，而更改主键设置的成本非常高。所以，如果你的系统比较复杂，尽量给表加一个字段做主键，采用手动赋值的办法，虽然系统开发的时候麻烦一点，却可以避免后面出大问题。

思考题

在刚刚的例子中，如果我想把销售流水表 demo.trans 中，所有单位是“包”的商品的价格改成原来价格的 80%，该怎么实现呢？

欢迎在留言区写下你的思考和答案，我们一起交流讨论。如果你觉得今天的内容对你有所帮助，欢迎你把它分享给你的朋友或同事，我们下节课见。

提建议

12.12 大促

每日一课 VIP 年卡

10分钟，解决你的技术难题

¥159/年 ¥365/年

每日一课
VIP 年卡

仅3天，【点击】图片，立即抢购 >>>

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 04 | 增删改查：如何操作表中的数据？

下一篇 06 | 外键和连接：如何做关联查询？

精选留言 (8)

写留言

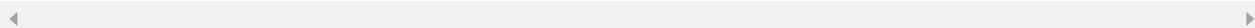


李鸣

2021-03-22

关于门店新加会员的id冲突，我们是不是可以通过加上门店编号来解决，不需要每次都去总部mysql去搜一遍

作者回复: 不建议通过给id增加与业务有关的信息解决问题，比如门店编号，原因是门店编号也是有可能会重复或者重用的



2



Harry

2021-03-18

交作业了：

```
update demo.trans set price = price * 0.8 where itemnumber in (select itemnumber from demo.goodsmaster where unit = '包');
```

展开 ∨



2



右耳朵猫咪

2021-03-18

如果是多个门店同时添加会员信息呢？那么这些门店查询的max_id是一样的，添加会员的id也是一样的了，即max_id+1。

展开 ∨

作者回复：需要用事物，防止别的连接读取到错误的信息



3

1



王建

2021-03-29

老师好，

1. 既然门店客户端能够更新总部数据库为啥还要把数据放到本地呢。
2. 每次插入数据都要去总部管理信息表获取ID，插入后还要更新总部管理信息表，这个管理信息表会导致资源争用，在数据并发请求高的时候肯定会出现阻塞或者更新失败的情况吧。...

展开 ∨



paul

2021-03-29

1. 主键不能用会员号或身份证，除了出于业务的考虑，没有其它原因吗？老师不提一下聚集索引？
2. 门店维护全局的max id的方式不好：总部不应该把这种全局控制交给门店，如果某个分店bug或者恶意非恶意修改了max id呢，影响了整个系统

展开 ∨



lesserror

2021-03-19

目前的业务都是使用自增主键的场景为主，偶有使用业务字段的老项目。采用自定义主键

的方式还没有尝试过。

另外，希望课程中的MySQL的执行语句和执行结果输出能够分开展示。或者去掉命令行中的一些特殊字符。复制代码的时候复制的是整个命令行的字符，体验不是太好。望采纳~...
展开

作者回复: 你的建议非常好，谢谢！



陈启年
2021-03-18

老师我在想，遇到这种id问题，可否简单一些：id TEXT
约定规则为 门店id+时间，这样也能满足要求，而且还有语义

作者回复: 不建议这样做，业务字段有重复和重用的可能，比如门店编号，系统时间也有错误的时候



Harry
2021-03-18

小结一下：

在创建表的时候，若要将某个字段设置为主键，直接为其添加主键约束即可。
...
展开