



下载APP



## 04 | 增删改查：如何操作表中的数据？

2021-03-16 朱晓峰

MySQL 必知必会

[进入课程 >](#)**讲述：朱晓峰**

时长 13:40 大小 12.53M



你好，我是朱晓峰。今天，我们来聊一聊如何操作数据表里的数据。

在咱们的超市项目中，我们给用户设计好了一个数据表 `demo.goodsmaster`，定义好了里面的字段，以及各种约束，如下所示：

```
1 mysql> DESCRIBE demo.goodsmaster;
```

```
2 +-----+-----+-----+-----+-----+-----+
```

```
3 | Field          | Type          | Null | Key | Default | Extra          |
```

```
4 +-----+-----+-----+-----+-----+-----+
```

```
5 | itemnumber     | int           | NO   | PRI | NULL     | auto_increment |
```

```
6 | barcode        | text          | NO   |     | NULL     |                |
```

```
7 | goodsname      | text          | NO   |     | NULL     |                |
```

```
8 | specification  | text          | YES  |     | NULL     |                |
```

```
9 | unit           | text          | YES  |     | NULL     |                |
```

[复制代码](#)

```
10 | price          | decimal(10,2) | NO    |          | NULL    |          |
11 +-----+-----+-----+-----+-----+-----+
12 6 rows in set (0.02 sec)
```

现在，我们需要使用这个表来存取数据。那么，如何对表里面的数据进行操作呢？接下来，我就给你讲讲操作表中的数据的方法，也就是常说的“增删改查”。

## 添加数据

我们先来看一下添加数据的语法结构：

```
1 INSERT INTO 表名 [(字段名 [, 字段名] ...)] VALUES (值的列表);
```

[复制代码](#)

这里方括号“[]”表示里面的内容可选，也就是说，根据 MySQL 的语法要求，写不写都可以。

添加数据分为两种情况：插入数据记录和插入查询结果。下面我分别来介绍下。

## 插入数据记录

其实，MySQL 支持的数据插入操作十分灵活。你既可以通过给表里面所有的字段赋值，完整地插入一条数据记录，也可以在插入记录的时候，只给部分字段赋值。

这是什么意思呢？我借助一个例子来给你讲解下。

比如，我们有一个叫 demo.goodsmaster 的表，包括了 itemnumber、barcode、goodsname、specification、unit 和 price 共 6 个字段。我想要插入一条数据记录，其中包含了所有字段的值，就可以这样操作：

```
1 INSERT INTO demo.goodsmaster
2 (
3   itemnumber,
4   barcode,
5   goodsname,
6   specification,
```


[复制代码](#)

```

7    unit,
8    price
9 )
10 VALUES
11 (
12     4,
13     '0003',
14     '尺子',
15     '三角型',
16     '把',
17     5
18 ):

```

运行这个 SQL 语句，然后对数据表进行查询，可以得到下面的结果：

 复制代码

```

1 mysql> SELECT *
2     -> FROM demo.goodsmaster;
3 +-----+-----+-----+-----+-----+-----+
4 | itemnumber | barcode | goodsname | specification | unit | price |
5 +-----+-----+-----+-----+-----+-----+
6 |          4 | 0003    | 尺子      | 三角型        | 把   | 5.00  |
7 +-----+-----+-----+-----+-----+-----+
8 1 row in set (0.01 sec)

```

如果我想插入一条记录，但是只给部分字段赋值，可不可以呢？比如说，客户有个商品，需要马上上线销售，目前只知道条码、名称和价格，其它的信息先不录入，等之后再补，可以实现吗？

我们来尝试一下只给 3 个字段赋值，看看实际操作的结果：

 复制代码


```

1 INSERT INTO demo.goodsmaster
2 (
3     -- 这里只给3个字段赋值，itemnumber、specification、unit不赋值
4     barcode,
5     goodsname,
6     price
7 )
8 VALUES
9 (
10    '0004',
11    '测试',
12    10

```

```
13 );
```


运行这条 SQL 语句，我们来查询表的内容，就会发现，显然是可以的。

 复制代码

```
1 mysql> SELECT *
2     -> FROM demo.goodsmaster;
3 +-----+-----+-----+-----+-----+-----+
4 | itemnumber | barcode | goodsname | specification | unit | price |
5 +-----+-----+-----+-----+-----+-----+
6 |          4 | 0003    | 尺子      | 三角型        | 把   | 5.00 |
7 |          5 | 0004    | 测试      | NULL          | NULL | 10.00 |
8 +-----+-----+-----+-----+-----+-----+
9 2 rows in set (0.00 sec)
```

我们之所以能够在插入一条数据记录的时候，只给部分字段赋值，原因就在于我们对字段的定义方式。

那字段是怎么定义的呢？我们来查看一下表的结构，看看各个字段的定义：

 复制代码

```
1 mysql> DESCRIBE demo.goodsmaster;
2 +-----+-----+-----+-----+-----+-----+
3 | Field          | Type          | Null | Key | Default | Extra          |
4 +-----+-----+-----+-----+-----+-----+
5 | itemnumber     | int           | NO   | PRI | NULL    | auto_increment |
6 | barcode        | text          | NO   |     | NULL    |                |
7 | goodsname      | text          | NO   |     | NULL    |                |
8 | specification   | text          | YES  |     | NULL    |                |
9 | unit           | text          | YES  |     | NULL    |                |
10 | price          | decimal(10,2) | NO   |     | NULL    |                |
11 +-----+-----+-----+-----+-----+-----+
12 6 rows in set (0.01 sec)
```

可以看到，我们在插入数据时没有明确赋值的 3 个字段，都有着各自的特点。“specification” 和 “unit” 都可以是空值，而 itemnumber 则定义了自增约束。

我们在插入一条数据记录的时候，必须要考虑字段约束的 3 种情况。


第一种情况是，如果字段允许为空，而我们没有给它赋值，那么 MySQL 会自动给它们赋予空值。在刚刚的代码中，“specification” “unit” 字段都允许为空，因此，如果数据插入语句没有指定这几个字段的值，MySQL 会自动插入空值。

第二种情况是，如果字段是主键，就不能为空，这个时候，MySQL 会按照我们添加的约束进行处理。比如字段 “itemnumber” 是主键，不能为空，而我们定义了自增约束，所以 MySQL 自动在之前的最大值基础上加了 1。因此，“itemnumber” 也有了自己该有的值。

第三种情况是，如果有一个字段定义不能为空，又不是主键，当你插入一条数据记录的时候，就必须给这个记录赋值。

如果我们的操作违反了字段的约束限制，会出现什么情况呢？


比如说，我们尝试把表 demo.goodsmaster 的字段 “speicification” 改为不能为空：

 复制代码

```
1 ALTER TABLE demo.goodsmaster
2 MODIFY specification TEXT NOT NULL;
```

运行这个 SQL 语句，系统会提示错误，原因就是我们刚才部分插入了一条数据记录，没有给字段 “specification” 赋值，这跟我们给字段 “specification” 添加非空约束的操作冲突了。


因此，我们要把字段 “speicification” 的值为空的数据记录删除，然后再修改字段约束：

 复制代码

```
1 DELETE
2 FROM demo.goodsmaster
3 WHERE itemnumber=5;
```

删除数据记录之后，再运行上面的语句，给字段 “specification” 添加非空约束，就成功了。

现在我们来验证一下非空约束。我们尝试部分插入一条数据记录，不给字段 “specification” 赋值：

 复制代码

```
1 INSERT INTO demo.goodsmaster
2 (
3     barcode,
4     goodsname,
5     price
6 )
7 VALUES
8 (
9     '0004',
10    '测试',
11    10
12 );
```

运行这个 SQL 语句，MySQL 报告错误，提示字段 “specification” 没有默认值。也就是说，这个字段不能为空，如果插入数据时不给它赋值，就必须给它一个默认值。

现在，你一定清楚了，**部分插入一条数据记录是可以的，但前提是，没有赋值的字段，一定要让 MySQL 知道如何处理，比如可以为空、有默认值，或者是自增约束字段，等等，否则，MySQL 会提示错误的。**

好了，到这里，我们就学会了给 MySQL 的数据表插入一条数据记录。但是，在实际工作中，一次只插入一条数据，有时候会不够用。

我们的项目就有这样的场景：门店每天的销售流水都很多，日积月累，流水表变得越来越大。如果就让它这么不断地增长，数据量甚至会达到数亿条，占据的存储空间达到几十个 G。虽然 MySQL 可以处理这样比较大的数据表，但是每次操作的响应时间就会延长，这会导致系统的整体效率下降。

刚开始的时候，我们开发了一个日结处理，当天算清所有账目。其中一个步骤就是，把当天流水表的数据全部转到历史流水表当中去。现在，我们就可以用上数据插入语句了。具体有 2 步：

1. 从流水表中取出一条数据；



## 2. 把这条数据插入到历史流水表中。

然后不断重复这两个步骤，一直到把今天流水表的数据全部插入到历史流水表当中去。

你肯定会说，这种做法的效率也太低了吧？有没有更好的办法呢？

当然是有的。这个时候，我们就要用到 MySQL 的另一种数据插入操作了：把查询结果插入到数据表中。

## 插入查询结果

MySQL 支持把查询的结果插入到数据表中，我们可以指定字段，甚至是数值，插入到数据表中。语法结构如下：

```
1 INSERT INTO 表名 (字段名)
2 SELECT 字段名或值
3 FROM 表名
4 WHERE 条件
```

[复制代码](#)

举个例子，在我们的超市信息系统的 MySQL 数据库中，历史流水表设计与流水表非常类似。不同的是，历史流水表增加了一些字段来标识历史流水的状态，比如日结时间字段，是用来记录日结操作是什么时候进行的。

用 INSERT 语句实现起来也很简单：

```
1 INSERT INTO 历史流水表 (日结时间字段, 其他字段)
2 SELECT 获取当前时间函数, 其他字段
3 FROM 流水表
```

[复制代码](#)

好了，添加数据的操作就讲完了，现在你知道了，我们给一张数据表插入一条数据记录的时候，可以给所有的字段赋值，也可以给部分字段赋值。这取决于字段的定义。如果字段不能为空并且没有默认值，就必须赋值。另外，我们还可以通过把一个查询结果插入数据表中的方式，提高添加数据的效率。

接下来，我们再来看看如何删除数据。

## 删除数据

数据删除的语法很简单，如下所示：

```
1 DELETE FROM 表名
2 WHERE 条件
```

[复制代码](#)

如果我们现在想把刚才用过的表 `demo.goodsmaster` 里的内容清理一下，删除全部数据，可以通过下面的 SQL 语句来实现：

```
1 DELETE FROM demo.goodsmaster;
```

[复制代码](#)

因为我们的查询主要是在 MySQL 的图形化管理工具中进行，现在，我们在 Workbench 中运行上面的 SQL 语句。这个时候，Workbench 会提示错误。

这是因为，Workbench 自动处于安全模式，它要求对数据的删除或修改操作中必须包含 WHERE 条件。而且，这个 WHERE 条件中，必须用到主键约束或者唯一性约束的字段。MySQL 的这种安全性设置，主要就是为了防止删除或者修改数据时出现误操作，导致删除或修改了不相关的数据。

因此，我们要习惯在删除数据的时候，添加条件语句 WHERE，防止误操作。

假设我们的数据表 `demo.goodsmaster` 中有如下数据记录：

```
1 mysql> SELECT *
2     -> FROM demo.goodsmaster;
3 +-----+-----+-----+-----+-----+-----+
4 | itemnumber | barcode | goodsname | specification | unit | price |
5 +-----+-----+-----+-----+-----+-----+
6 |          4 | 0003    | 尺子      | 三角型        | 把   | 5.00  |
7 |          5 | 0004    | 测试      | NULL          | NULL | 10.00 |
8 +-----+-----+-----+-----+-----+-----+
```

[复制代码](#)



```
9  2 rows in set (0.00 sec)
```

如果我们要删除所有数据，可以这样写：

```
1  DELETE FROM demo.goodsmaster
2  WHERE itemnumber > 1;
```

[复制代码](#)

好了，下面我们来学习一下修改数据。

## 修改数据

我们来看一下 MySQL 的数据修改语法：

```
1  UPDATE 表名
2  SET 字段名=值
3  WHERE 条件
```

[复制代码](#)

语法很简单，需要注意的一点是，**不要修改主键字段的值**。因为主键是数据记录的唯一标识，如果修改了主键的值，就有可能破坏数据的完整性。

下面我们来看一个通过主键查询商品信息的例子，看看修改主键的值，会产生什么样的结果。

```
1  mysql> SELECT *
2      -> FROM demo.goodsmaster
3      -> WHERE itemnumber = 3;
4  +-----+-----+-----+-----+-----+-----+
5  | itemnumber | barcode | goodsname | specification | unit | price |
6  +-----+-----+-----+-----+-----+-----+
7  |          3 | 0003    | 尺子      | 三角型        | 把   | 5.00  |
8  +-----+-----+-----+-----+-----+-----+
9  1 row in set (0.00 sec)
```

[复制代码](#)

我们能查询到一条商品编号是 3 的数据记录：条码是 “0003” ，名称是 “尺子” ，价格是 5 元。

如果我们修改了主键的值，就可能会改变刚才的查询结果：

[复制代码](#)

```
1 mysql> UPDATE demo.goodsmaster
2     -> SET itemnumber=2
3     -> WHERE itemnumber = 3;
4 Query OK, 1 row affected (0.02 sec)
5 Rows matched: 1  Changed: 1  Warnings: 0
6
7 mysql> SELECT *
8     -> FROM demo.goodsmaster
9     -> WHERE itemnumber = 3;
10 Empty set (0.00 sec)
```

可以看到，查询结果是空，因为商品编号是 3 的数据记录，已经不存在了。

当然，如果你必须要修改主键的值，那有可能就是主键设置得不合理。关于主键的设置问题，我会在下节课给你详细介绍。这里你只需要知道，不要随便修改表的主键值，就可以了。

## 查询数据

经过了前面的操作之后，现在，我们知道了把数据存入数据表里，以及删除和修改表里数据的方法。那么如何知道数据操作的结果呢？这就要用到数据查询了。

我们先来看下查询语句的语法结构：

[复制代码](#)

```
1 SELECT * | 字段列表
2 FROM 数据源
3 WHERE 条件
4 GROUP BY 字段
5 HAVING 条件
6 ORDER BY 字段
7 LIMIT 起始点, 行数
```

在这些字段中，SELECT、WHERE、GROUP BY 和 HAVING 比较好理解，你只要知道它们的含义就可以了。

**SELECT：**是查询关键字，表示我们要做一个查询。“\*” 是一个通配符，表示我们要查询表中所有的字段。你也可以把要查询的字段罗列出来，这样，查询的结果可以只显示你想要查询的字段内容。

**WHERE：**表示查询条件。你可以把你要查询的数据所要满足的条件，放在 WHERE 关键字之后。

**GROUP BY：**作用是告诉 MySQL，查询结果要如何分组，经常与 MySQL 的聚合函数一起使用。

**HAVING：**用于筛选查询结果，跟 WHERE 类似。

FROM、ORDER BY 和 LIMIT 相对来说比较复杂，需要注意的地方比较多，我来具体给你解释一下。

## FROM

FROM 关键字表示查询的数据源。我们现在只学习了单个数据表，你可以把你要查询的数据表名，直接写在 FROM 关键字之后。以后我们在学到关联表的时候，你就会知道，FROM 关键字后面，还可以跟着更复杂的数据表联接。

这里需要你注意的是，数据源也不一定是表，也可以是一个查询的结果。比如下面的查询：

```
mysql> SELECT a.goodsname, a.price
-> FROM (
-> SELECT *
-> FROM demo.goodsmaster
-> ) AS a;
```

派生表 (Derived Table) 或者子查询 (subquery)

goodsname	price
尺子	5.00
测试	10.00


2 rows in set (0.00 sec)

这里你要注意的，红色框里的部分叫做派生表（derived table），或者子查询（subquery），意思是我们把一个查询结果数据集当做一个虚拟的数据表来看待。MySQL 规定，必须要用 AS 关键字给这个派生表起一个别名。在这张图中，我给这个派生表起了个名字，叫做 “a”。

## ORDER BY


ORDER BY 的作用，是告诉 MySQL，查询结果如何排序。**ASC 表示升序，DESC 表示降序。**

我来举个简单的小例子，带你看看 ORDER BY 是怎么使用的（这里我们仍然假设字段 “specification” 和 “unit” 允许为空）。我们向表 demo.goodsmaster 中插入 2 条数据：

 复制代码

```
1 INSERT INTO demo.goodsmaster
2 (
3     barcode,
4     goodsname,
5     price
6 )
7 VALUES
8 (
9     '0003',
10    '尺子1',
11    15
12 );
13 INSERT INTO demo.goodsmaster
14 (
15     barcode,
16     goodsname,
17     price
18 )
19 VALUES
20 (
21     '0004',
22     '测试1',
23     20
24 );
```

如果我们不控制查询结果的顺序，就会得到这样的结果：


 复制代码

```

1 mysql> SELECT *
2     -> FROM demo.goodsmaster;
3 +-----+-----+-----+-----+-----+-----+
4 | itemnumber | barcode | goodsname | specification | unit | price |
5 +-----+-----+-----+-----+-----+-----+
6 |          4 | 0003    | 尺子      | 三角型        | 把   | 5.00  |
7 |          5 | 0004    | 测试      | NULL          | NULL | 10.00 |
8 |          6 | 0003    | 尺子1     | NULL          | NULL | 15.00 |
9 |          7 | 0004    | 测试1     | NULL          | NULL | 20.00 |
10 +-----+-----+-----+-----+-----+-----+
11 4 rows in set (0.00 sec)

```

如果我们使用 ORDER BY 对查询结果进行控制，结果就不一样了：

 复制代码

```


1 mysql> SELECT *
2     -> FROM demo.goodsmaster
3     -> ORDER BY barcode ASC,price DESC;
4 +-----+-----+-----+-----+-----+-----+
5 | itemnumber | barcode | goodsname | specification | unit | price |
6 +-----+-----+-----+-----+-----+-----+
7 |          6 | 0003    | 尺子1     | NULL          | NULL | 15.00 |
8 |          4 | 0003    | 尺子      | 三角型        | 把   | 5.00  |
9 |          7 | 0004    | 测试1     | NULL          | NULL | 20.00 |
10 |          5 | 0004    | 测试      | NULL          | NULL | 10.00 |
11 +-----+-----+-----+-----+-----+-----+
12 4 rows in set (0.00 sec)

```

可以看到，查询结果会先按照字段 barcode 的升序排序，相同 barcode 里面的字段，按照 price 的降序排序。

## LIMIT

LIMIT 的作用是告诉 MySQL 只显示部分查询的结果。比如，现在我们的数据表 demo.goodsmaster 中有 4 条数据，我们只想要显示第 2、3 条数据，就可以用 LIMIT 关键字来实现：

 复制代码

```

1 mysql> SELECT *
2     -> FROM demo.goodsmaster
3     -> LIMIT 1,2;
4

```

```


5  +-----+-----+-----+-----+-----+
6  | itemnumber | barcode | goodsname | specification | unit | price |
7  +-----+-----+-----+-----+-----+
8  |           5 | 0004    | 测试      | NULL          | NULL | 10.00 |
9  |           6 | 0003    | 尺子1     | NULL          | NULL | 15.00 |
10 +-----+-----+-----+-----+-----+
    query in test (0.00 sec)

```

这里的“LIMIT 1,2”中，“1”表示起始位置，MySQL 中，起始位置的起点是 0，1 表示从第 2 条记录开始；“2”表示 2 条数据。因此，“LIMIT 1,2”就表示从第 2 条数据开始，显示 2 条数据，也就是显示了第 2、3 条数据。

## 总结

今天，我们学习了添加、删除、修改和查询数据的方法，这些都是我们经常遇到的操作，你一定要好好学习。下面的这些 SQL 语句，是我总结的数据操作语法，你一定要重点掌握。

 复制代码

```

1  INSERT INTO 表名 [(字段名 [,字段名] ...)] VALUES (值的列表);
2
3  INSERT INTO 表名 (字段名)
4  SELECT 字段名或值
5  FROM 表名
6  WHERE 条件
7
8  DELETE FROM 表名
9  WHERE 条件
10
11 UPDATE 表名
12 SET 字段名=值
13 WHERE 条件
14
15 SELECT *|字段列表
16 FROM 数据源
17 WHERE 条件
18 GROUP BY 字段
19 HAVING 条件
20 ORDER BY 字段
21 LIMIT 起始点, 行数

```

最后，我再补充一点。如果我们把查询的结果插入到表中时，导致主键约束或者唯一性约束被破坏了，就可以用“ON DUPLICATE”关键字进行处理。这个关键字的作用是，告诉



MySQL，如果遇到重复的数据，该如何处理。

我来给你举个例子。


假设用户有 2 个各自独立的门店，分别有自己的系统。现在需要引入连锁经营的模式，把 2 个店用一套系统统一管理。那么首先遇到的问题就是，需要进行数据整合。下面我们就以商品信息表为例，来说明如何通过使用“ON DUPLICATE”关键字，把两个门店的商品信息数据整合到一起。

假设门店 A 的商品信息表是“demo.goodsmaster”，代码如下：

 复制代码

```
1 mysql> SELECT *
2     -> FROM demo.goodsmaster;
3 +-----+-----+-----+-----+-----+-----+
4 | itemnumber | barcode | goodsname | specification | unit | salesprice |
5 +-----+-----+-----+-----+-----+-----+
6 |          1 | 0001    | 书        | 16开          | 本   | 89.00    |
7 |          2 | 0002    | 笔        | 10支装        | 包   | 5.00     |
8 |          3 | 0003    | 橡皮      | NULL          | 个   | 3.00     |
9 +-----+-----+-----+-----+-----+-----+
10 3 rows in set (0.00 sec)
```

门店 B 的商品信息表是“demo.goodsmaster1”：

 复制代码

```
1 mysql> SELECT *
2     -> FROM demo.goodsmaster1;
3 +-----+-----+-----+-----+-----+-----+
4 | itemnumber | barcode | goodsname | specification | unit | salesprice |
5 +-----+-----+-----+-----+-----+-----+
6 |          1 | 0001    | 教科书    | NULL          | NULL | 89.00    |
7 |          4 | 0004    | 馒头      |               |      | 1.50     |
8 +-----+-----+-----+-----+-----+-----+
9 2 rows in set (0.00 sec)
```

假设我们要把门店 B 的商品数据，插入到门店 A 的商品表中去，如果有重复的商品编号，就用门店 B 的条码，替换门店 A 的条码，用门店 B 的商品名称，替换门店 A 的商品名

称；如果没有重复的编号，就直接把门店 B 的商品数据插入到门店 A 的商品表中。这个操作，就可以用下面的 SQL 语句实现：

[复制代码](#)

```
1 INSERT INTO demo.goodsmaster
2 SELECT *
3 FROM demo.goodsmaster1 as a
4 ON DUPLICATE KEY UPDATE barcode = a.barcode,goodsname=a.goodsname;
5 -- 运行结果如下
6 mysql> SELECT *
7       -> FROM demo.goodsmaster;
8 +-----+-----+-----+-----+-----+-----+
9 | itemnumber | barcode | goodsname | specification | unit | salesprice |
10 +-----+-----+-----+-----+-----+-----+
11 |          1 | 0001   | 教科书   | 16开         | 本   | 89.00   |
12 |          2 | 0002   | 笔       | 10支装       | 包   | 5.00    |
13 |          3 | 0003   | 橡皮     | NULL        | 个   | 3.00    |
14 |          4 | 0004   | 馒头     |              |      | 1.50    |
15 +-----+-----+-----+-----+-----+-----+
16 4 rows in set (0.00 sec)
```

最后，我再跟你分享 3 份资料，分别是 [MySQL 数据插入](#)、[MySQL 数据更新](#)和 [MySQL 数据查询](#)，如果你在工作中遇到了更加复杂的操作需求，就可以参考一下。

## 思考题

我想请你思考一个问题：商品表 demo.goodsmaster 中，字段 “itemnumber” 是主键，而且满足自增约束，如果我删除了一条记录，再次插入数据的时候，就会出现字段 “itemnumber” 的值不连续的情况。请你想一想，如何插入数据，才能防止这种情况的发生呢？

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，也欢迎你分享你的朋友或同事，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 03 | 表：怎么创建和修改表？

下一篇 05 | 主键：如何正确设置主键？

## 精选留言 (9)

写留言



小白

2021-03-24

ALTER TABLE demo.goodsmaster AUTO\_INCREMENT=断点数值

1

2



FengX

2021-04-02

```
SELECT @itemnumber=MIN(a.itemnumber)+1
FROM demo.goodsmaster as a
WHERE NOT EXISTS(
SELECT b.itemnumber
FROM demo.goodsmaster as b...
```

展开 ∨

**洛奇**

2021-03-27

不用非得自增主键连续吧？？？

展开 ∨

作者回复: 是的，也可使用其它字段作为主键，包括使用多字段作为联合主键。需要注意的是，要避免因为主键字段数据的修改影响到关联的历史数据查询。若是不考虑水平拆分的问题，带来额外设置上的麻烦，则自增序列是最佳的主键字段选择。

**洛奇**

2021-03-27

请问朱老师，INSERT...ON DUPLICATE 语句存在死锁的可能吗？我记得在一次携程面试的时候，面试官考我这个问题，我当时却以为它是原子操作。

展开 ∨

**李鸣**

2021-03-22

```
set @zizen = 0;
```

```
UPDATE demo.goodsmaster
```

```
set itemnumber = (select @zizen := @zizen + 1 as nid) where demo.goodsmaster.itemnumber IS NULL;
```

展开 ∨

**Harry**

2021-03-17

小结一下：

1. 如果要为所有字段插入值，则可以省略字段。
2. 为字段新增或修改非空约束时，会对表中之前存储的数据进行验证，如果不符合约束...

展开 ∨



**Harry**

2021-03-17

看完此篇，  
查漏补缺，  
增删改查，  
却也不差

展开 ∨

**星空下**

2021-03-16

使用max获取当前最大的主键id，id+1作为新的主键插入数据库。但是高并发场景下可能会出现主键冲突问题。其他的暂时没有想到，希望老师给出权威答案。

展开 ∨

作者回复: 通过max函数获取最大值，然后加1，这个逻辑太简单了，不足以应对高并发的场景。应该根据实际情况，在应用层面设计一个模块，专门来处理主键ID的计算，解决唯一性，不连续等问题。



2

**右耳朵猫咪**

2021-03-16

老师好 我想请教您一个问题 mysql分页有必要加order by 某个字段吗 听说如果不加的话 查询第二页数据的时候可能会查询到第一页的数据 如果加order by,可能会影响性能，这该如何取舍呢？

作者回复: 建议加ORDER BY，分页本身就是一种查询的优化



1

