



下载APP



28 | 手把手带你设计一个完整的连锁超市信息系统数据库（下）

2021-05-18 朱晓峰

MySQL 必知必会

[进入课程 >](#)**讲述：朱晓峰**

时长 08:36 大小 7.88M



你好，我是朱晓峰。

上节课，我们完成了项目的需求分析和业务流程的梳理，为设计数据库做好了准备工作，接下来我们就可以开始具体的设计了。所以，今天，我就带你来建库建表、创建外键约束、视图、存储过程和触发器，最后制定容灾和备份的策略，从而完成一个完整的连锁超市项目数据库的设计，帮助你提高设计高效可靠的数据库的能力。


首先，我们一起来创建数据库和数据表。



如何创建数据库和数据表？

经过上节课的分库分表操作，我们把数据库按照业务模块，拆分成了多个数据库。其中，盘点模块中的数据表分别被拆分到了营运数据库（operation）和库存数据库（inventory）中。

下面我们就按照上节课的分库策略，分别创建营运数据库和库存数据库：


 复制代码

```
1 mysql> CREATE DATABASE operation;
2 Query OK, 1 row affected (0.03 sec)
3
4 mysql> CREATE DATABASE inventory;
5 Query OK, 1 row affected (0.02 sec)
```

接下来，我们来分别创建下这两个数据库中的表。

商户表、门店表、员工表、商品常用信息表和商品不常用信息表从属于营运数据库，我们先把这 5 个表创建出来。

商户表（operation.enterprise）：

 复制代码

```
1 mysql> CREATE TABLE operation.enterprise
2 -> (
3 -> groupnumber SMALLINT PRIMARY KEY, -- 组号
4 -> groupname VARCHAR(100) NOT NULL, -- 名称
5 -> address TEXT NOT NULL, -- 地址
6 -> phone VARCHAR(20) NOT NULL, -- 电话
7 -> contactor VARCHAR(50) NOT NULL -- 联系人
8 -> );
9 Query OK, 0 rows affected (0.05 sec)
```

门店表（operation.branch）：

 复制代码

```
1 mysql> CREATE TABLE operation.branch
2 -> (
3 -> branchid SMALLINT PRIMARY KEY, -- 门店编号
4 -> groupnumber SMALLINT NOT NULL, -- 组号
5 -> branchname VARCHAR(100) NOT NULL, -- 门店名称
```

```
6 -> address TEXT NOT NULL,          -- 地址
7 -> phone VARCHAR(20) NOT NULL,      -- 电话
8 -> branchtype VARCHAR(20) NOT NULL,  -- 门店类别
9 -> CONSTRAINT fk_branch_enterprice FOREIGN KEY (groupnumber) REFERENCES operat
10 -> );
11 Query OK, 0 rows affected (0.07 sec)
```

员工表 (operation.employee) :

[复制代码](#)

```
1 mysql> CREATE TABLE operation.employee
2 -> (
3 -> employeeid SMALLINT PRIMARY KEY,    -- 员工编号
4 -> groupnumber SMALLINT NOT NULL,      -- 组号
5 -> branchid SMALLINT NOT NULL,         -- 门店编号
6 -> workno VARCHAR(20) NOT NULL,        -- 工号
7 -> employeename VARCHAR(100) NOT NULL, -- 员工名称
8 -> pid VARCHAR(20) NOT NULL,           -- 身份证
9 -> address VARCHAR(100) NOT NULL,      -- 地址
10 -> phone VARCHAR(20) NOT NULL,        -- 电话
11 -> employeeeduty VARCHAR(20) NOT NULL, -- 职责
12 -> CONSTRAINT fk_employee_branch FOREIGN KEY (branchid) REFERENCES operation.b
13 -> );
14 Query OK, 0 rows affected (0.07 sec)
```

商品常用信息表 (operation.goods_o) :

[复制代码](#)

```
1 mysql> CREATE TABLE operation.goods_o
2 -> (
3 -> itemnumber MEDIUMINT PRIMARY KEY,   -- 商品编号
4 -> groupnumber SMALLINT NOT NULL,       -- 组号
5 -> barcode VARCHAR(50) NOT NULL,        -- 条码
6 -> goodsname TEXT NOT NULL,             -- 名称
7 -> salesprice DECIMAL(10,2) NOT NULL,   -- 售价
8 -> PRIMARY KEY (groupnumber,itemnumber) -- 主键
9 -> );
10 Query OK, 0 rows affected (0.06 sec)
```

商品不常用信息表 (operation.goods_f) :

[复制代码](#)

```
1 mysql> CREATE TABLE operation.goods_f
2 -> (
3 -> itemnumber MEDIUMINT PRIMARY KEY,      -- 商品编号
4 -> groupnumber SMALLINT NOT NULL,         -- 组号
5 -> specification TEXT NOT NULL,           -- 规格
6 -> unit VARCHAR(20) NOT NULL,             -- 单位
7 -> PRIMARY KEY (groupnumber,itemnumber)    -- 主键
8 -> );
9 Query OK, 0 rows affected (0.06 sec)
```

好了，现在我们来创建库存数据库中的表。仓库表、库存表、盘点单头表、盘点单明细表、盘点单头历史表和盘点单明细历史表，从属于库存数据库。

仓库表 (inventory.stockmaster) :

[复制代码](#)


```
1 mysql> CREATE TABLE inventory.stockmaster
2 -> (
3 -> stockid SMALLINT PRIMARY KEY,          -- 仓库编号
4 -> groupnumber SMALLINT NOT NULL,         -- 组号
5 -> branchid SMALLINT NOT NULL,           -- 门店编号
6 -> stockname VARCHAR(100) NOT NULL,       -- 仓库名称
7 -> stockkind VARCHAR(20) NOT NULL,        -- 仓库种类
8 -> CONSTRAINT fk_stock_branch FOREIGN KEY (branchid) REFERENCES operation.bran
9 -> );
10 Query OK, 0 rows affected (0.07 sec)
```

库存表 (inventory.inventory) :

[复制代码](#)


```
1 mysql> CREATE TABLE inventory.inventory
2 -> (
3 -> id INT PRIMARY KEY AUTO_INCREMENT,     -- 库存编号
4 -> groupnumber SMALLINT NOT NULL,         -- 组号
5 -> branchid SMALLINT NOT NULL,           -- 门店编号
6 -> stockid SMALLINT NOT NULL,            -- 仓库编号
7 -> itemnumber MEDIUMINT NOT NULL,        -- 商品编号
8 -> itemquantity DECIMAL(10,3) NOT NULL   -- 商品数量
9 -> );
10 Query OK, 0 rows affected (0.06 sec)
```

盘点单头表 (inventory.invcounthead) :

 复制代码

```
1 mysql> CREATE TABLE inventory.invcounthead
2 -> (
3 -> listnumber INT PRIMARY KEY,           -- 单号
4 -> groupnumber SMALLINT NOT NULL,        -- 组号
5 -> branchid SMALLINT NOT NULL,           -- 门店编号
6 -> stockid SMALLINT NOT NULL,            -- 仓库编号
7 -> recorder SMALLINT NOT NULL,           -- 录入人编号
8 -> recordingdate DATETIME NOT NULL       -- 录入时间
9 -> );
10 Query OK, 0 rows affected (0.07 sec)
```

盘点单明细表 (inventory.invcountdetails) :

 复制代码

```
1 mysql> CREATE TABLE inventory.invcountdetails
2 -> (
3 -> id INT PRIMARY KEY AUTO_INCREMENT,    -- 明细编号
4 -> listnumber INT NOT NULL,               -- 单号
5 -> groupnumber SMALLINT NOT NULL,         -- 组号
6 -> branchid SMALLINT NOT NULL,           -- 门店编号
7 -> stockid SMALLINT NOT NULL,            -- 仓库编号
8 -> itemnumber MEDIUMINT NOT NULL,        -- 商品编号
9 -> accquant DECIMAL(10,3) NOT NULL,      -- 结存数量
10 -> invquant DECIMAL(10,3) NOT NULL,      -- 盘存数量
11 -> plquant DECIMAL(10,3) NOT NULL       -- 盈亏数量
12 -> );
13
14 Query OK, 0 rows affected (0.07 sec)
```

盘点单头历史表 (inventory.invcountheadhist) :

 复制代码

```
1 mysql> CREATE TABLE inventory.invcountheadhist
2 -> (
3 -> listnumber INT PRIMARY KEY,           -- 单号
4 -> groupnumber SMALLINT NOT NULL,        -- 组号
5 -> branchid SMALLINT NOT NULL,           -- 门店编号
6 -> stockid SMALLINT NOT NULL,            -- 仓库编号
7 -> recorder SMALLINT NOT NULL,           -- 录入人编号
8 -> recordingdate DATETIME NOT NULL,      -- 录入时间
9 -> confirmer SMALLINT NOT NULL,          -- 验收人编号
10 -> confirmationdate DATETIME NOT NULL   -- 验收时间
11 -> );
12 Query OK, 0 rows affected (0.10 sec)
```

盘点单明细历史表 (inventory.invcountdetailshist) :

[复制代码](#)

```
1 mysql> CREATE TABLE inventory.invcountdetailshist
2 -> (
3 -> id INT PRIMARY KEY AUTO_INCREMENT,      -- 明细编号
4 -> listnumber INT NOT NULL,                  -- 单号
5 -> groupnumber SMALLINT NOT NULL,            -- 组号
6 -> branchid SMALLINT NOT NULL,               -- 门店编号
7 -> stockid SMALLINT NOT NULL,                -- 仓库编号
8 -> itemnumber MEDIUMINT NOT NULL,           -- 商品编号
9 -> accquant DECIMAL(10,3) NOT NULL,          -- 结存数量
10 -> invquant DECIMAL(10,3) NOT NULL,         -- 盘存数量
11 -> plquant DECIMAL(10,3) NOT NULL          -- 盈亏数量
12 -> );
13 Query OK, 0 rows affected (1.62 sec)
```

至此，我们完成了创建数据库和数据表的工作。为了提高查询的速度，我们还要为数据表创建索引。下面我们就来实际操作一下。

如何创建索引？

索引对提升数据查询的效率作用最大，没有之一。我们创建索引的策略是：

1. 所有的数据表都必须创建索引；
2. 只要是有可能成为查询筛选条件的字段，都必须创建索引。

这样做的原因是，当数据量特别大的时候，如果没有索引，一旦出现大并发，没有索引的表很可能会成为访问的瓶颈。而且这个问题十分隐蔽，不容易察觉，系统也不会报错，但是却会消耗大量的 CPU 资源，导致系统事实上的崩溃。

在之前的操作中，我们一共创建了 11 个数据表，下面我们就来——为这些表创建索引。


商户表中的组号字段，常被用于筛选条件。我们用商户表的组号字段为商户表创建索引，代码如下所示：

```
1 mysql> CREATE INDEX index_enterprice_groupname ON operation.enterprice (groupname);
2 Query OK, 0 rows affected (0.06 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

 复制代码


门店表的组号字段也常被用作筛选条件，所以，我们用组号字段为门店表创建索引，代码如下所示：

```
1 mysql> CREATE INDEX index_branch_groupnumber ON operation.branch (groupnumber);
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

 复制代码

除了组号字段，门店名称字段也常用在查询的筛选条件中，下面我们就用门店名称字段为门店表创建索引：

```
1 mysql> CREATE INDEX index_branch_branchname ON operation.branch (branchname);
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

 复制代码

门店类别字段也是用作筛选条件的字段之一，我们可以用门店类别字段为门店表创建索引，如下所示：

```
1 mysql> CREATE INDEX index_branch_branchtype ON operation.branch (branchtype);
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

 复制代码

在员工表中，组号、门店编号、身份证号、电话和职责字段常被用作查询的筛选条件，下面我们就分别用这几个字段为员工表创建索引。

第一步，用组号字段为员工表创建索引：

```
1 mysql> CREATE INDEX index_employee_groupnumber ON operation.employee (groupnumber);
```

 复制代码


```
2 Query OK, 0 rows affected (0.06 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

第二步，用门店编号字段为员工表创建索引：

[复制代码](#)

```
1 mysql> CREATE INDEX index_employee_branchid ON operation.employee (branchid);
2 Query OK, 0 rows affected (0.07 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

第三步，用身份证字段为员工表创建索引：

[复制代码](#)

```
1 mysql> CREATE INDEX index_employee_pid ON operation.employee (pid);
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

第四步，用电话字段为员工表创建索引：

[复制代码](#)

```
1 mysql> CREATE INDEX index_employee_phone ON operation.employee (phone);
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```


最后，我们用职责字段为员工表创建索引：

[复制代码](#)

```
1 mysql> CREATE INDEX index_employee_duty ON operation.employee (employee_duty);
2 Query OK, 0 rows affected (0.09 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```


对于商品常用信息表（operation.goods_o），我们发现，组号和售价字段常被用在查询筛选条件语句中，所以，我们分别用这两个字段为商品常用信息表创建索引。

首先，用组号字段为商品常用信息表创建索引，如下所示：

 复制代码


```
1 mysql> CREATE INDEX index_goodso_groupnumber ON operation.goods_o (groupnumber
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

然后，用价格字段为商品常用信息表创建索引，如下所示：

 复制代码

```
1 mysql> CREATE INDEX index_goodso_salesprice ON operation.goods_o (salesprice);
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

对于商品不常用信息表，只有组号字段经常用在查询的筛选条件中，所以，我们只需要用组号字段为商品不常用信息表创建索引。代码如下：


 复制代码

```
1 mysql> CREATE INDEX index_goodsf_groupnumber ON operation.goods_f (groupnumber
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

到这里，我们就完成了为营运数据库中的表创建索引的工作，下面我们来为库存数据库中的表创建索引。

首先是仓库表。这个表中经常被用做筛选条件的字段，是组号和门店编号字段。

我们先用组号字段为仓库表创建索引，代码如下：

 复制代码

```
1 mysql> CREATE INDEX index_stock_groupnumber ON inventory.stockmaster (groupnum
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

然后我们用门店编号字段为仓库表创建索引，代码如下：

 复制代码

```
1 mysql> CREATE INDEX index_stock_branchid ON inventory.stockmaster (branchid);
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

接下来，我们为库存表创建索引。库存表中常用于筛选的字段有组号、门店编号和商品编号字段。

我们先用组号字段来创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_inventory_groupnumber ON inventory.inventory (groupn
2 Query OK, 0 rows affected (0.11 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

然后，我们用门店编号字段创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_inventory_branchid ON inventory.inventory (branchid)
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

最后用商品编号字段创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_inventory_itemnumber ON inventory.inventory (itemnum
2 Query OK, 0 rows affected (0.07 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

盘点单头表也需要创建索引，常用的筛选字段是门店编号。那么，我们就用门店编号为盘点单头表创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcounthead_branchid ON inventory.invcounthead (bra
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

盘点单头明细表中常用于筛选的字段有门店编号和商品编号，我们分别用这 2 个字段创建索引。

首先是用门店编号字段创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcountdetails_branchid ON inventory.invcountdetail
2 Query OK, 0 rows affected (0.08 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

然后用商品编号字段创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcountdetails_itemnumber ON inventory.invcountdeta
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

盘点单头历史表数据量比较大，主要用于查询，常用的筛选字段有门店编号和验收时间。我们先用门店编号字段创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcounthaedhist_branchid ON inventory.invcountheadh
2 Query OK, 0 rows affected (0.06 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

然后用验收时间字段创建索引，代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcounthaedhist_confirmationdate ON inventory.invco
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

盘点单明细历史表是整个盘点模块中数据量最大的表，主要用于查询，索引对提升查询效率来说非常重要。要是忘了创建索引，很可能成为整个系统的瓶颈。

这个表中用作筛选条件的字段主要有门店编号和商品编号，我们分别用它们创建索引。首先是门店编号字段，创建索引的代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcountdetailshist_branchid ON inventory.invcountde
2 Query OK, 0 rows affected (0.05 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

然后是商品编号字段，创建索引的代码如下：

[复制代码](#)

```
1 mysql> CREATE INDEX index_invcountdetailshist_itemnumber ON inventory.invcount
2 Query OK, 0 rows affected (0.04 sec)
3 Records: 0 Duplicates: 0 Warnings: 0
```

到这里，索引就全部创建完成了。

由于我们把盘点单拆分成了单头和明细两个表，在应用中，经常要用到单头和明细的全部信息，所以，为了使代码更加简洁，查询更加方便，我们要为这两个表创建视图。

如何创建视图？

首先，我们为盘点单创建视图，代码如下：

[复制代码](#)

```
1 mysql> CREATE VIEW view_invcount
2   -> AS
3   -> SELECT a.*,b.itemnumber,b.accquant,b.invquant,b.plquant
4   -> FROM inventory.invcounthead AS a
5   -> JOIN
6   -> inventory.invcountdetails AS b
7   -> ON (a.listnumber=b.listnumber);
8 Query OK, 0 rows affected (0.04 sec)
```

然后，我们为盘点单历史表创建视图，代码如下：

[复制代码](#)

```
1 mysql> CREATE VIEW view_invcounthist
2   -> AS
3   -> SELECT a.*,b.itemnumber,b.accquant,b.invquant,b.plquant
4   -> FROM inventory.invcounttheadhist AS a
5   -> JOIN inventory.invcountdetailshist AS b
6   -> ON (a.listnumber=b.listnumber);
7 Query OK, 0 rows affected (0.02 sec)
```

如何创建存储过程？

在整个盘点模块中，有一个核心的计算模块，就是盘点单验收模块。这个计算模块，每次盘点结束都会被调用。为了提升执行效率，让代码更加模块化，使代码的可读性更好，我们可以把盘点表验收这个模块的数据处理逻辑，用存储过程的方式保存在服务器上，以方便应用程序进行调用。

下面我具体介绍存储过程的入口参数和数据处理逻辑。

存储过程的入口参数是单号和验收人的员工编号。存储过程的数据处理逻辑是：先用盈亏数量调整库存，计算方式是新库存 = 老库存 + 盈亏数量；然后把盘点单数据移到盘点单历史中去。

把盘点单明细表中的数据插入到盘点单明细历史表中；

把盘点单头表中的数据，插入到盘点单头历史表中；

删除盘点单明细表中的数据；

删除盘点单头表中的数据。

按照这个参数定义和计算逻辑，我们就可以用下面的代码来创建存储过程了：

 复制代码

```
1 CREATE DEFINER=`root`@`localhost` PROCEDURE `invcountconfirm`(mylistnumber INT
2 BEGIN
3 DECLARE done INT DEFAULT FALSE;
4 DECLARE mybranchid INT;
5 DECLARE myitemnumber INT;
6 DECLARE myplquant DECIMAL(10,3);
7 DECLARE cursor_invcount CURSOR FOR
8 SELECT branchid,itemnumber,plquant
9 FROM inventory.invcountdetails
10 WHERE listnumber = mylistnumber;
```

```
11 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
12 DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK;
13
14 START TRANSACTION;
15 OPEN cursor_invcount; -- 打开游标
16 FETCH cursor_invcount INTO mybranchid,myitemnumber,myplquant; -- 读入第一条记录
17 REPEAT
18     UPDATE inventory.inventory
19     SET itemquantity = itemquantity + myplquant    -- 更新库存
20     WHERE itemnumber = myitemnumber
21     AND branchid = mybranchid;
22     FETCH cursor_invcount INTO mybranchid,myitemnumber,myplquant; -- 读入下一条记录
23 UNTIL done END REPEAT;
24
25 CLOSE cursor_invcount;
26
27 INSERT INTO inventory.invcountdetailshist
28 (listnumber,groupnumber,branchid,stockid,itemnumber,accquant,invquant,plquant)
29 SELECT listnumber,groupnumber,branchid,stockid,itemnumber,accquant,invquant,
30 FROM inventory.invcountdetails
31 WHERE listnumber=mylistnumber; -- 把这一单的盘点单明细插入历史表
32
33 INSERT INTO inventory.invcounttheadhist
34 (listnumber,groupnumber,branchid,stockid,recorder,recordingdate,confirmer,conf
35 SELECT listnumber,groupnumber,branchid,stockid,recorder,recordingdate,myconf
36 FROM inventory.invcountthead
37 WHERE listnumber=mylistnumber; -- 把这一单的盘点单头插入历史
38 DELETE FROM inventory.invcountdetails WHERE listnumber = mylistnumber; -- 删除
39 DELETE FROM inventory.invcountthead WHERE listnumber = mylistnumber; -- 删除这
40 COMMIT;
41 END
```


具体的操作我都标注在代码里面了，你可以看一下。

如何创建触发器？

创建完了存储过程，我们已经完成了一大半，但是别急，还有一步工作没有做，就是创建触发器。

由于我们根据分库分表的策略，把商品信息表拆分成了商品常用信息表和商品不常用信息表，这样就很容易产生数据不一致的情况。为了确保商品常用信息表和商品不常用信息表中的数据保持一致，我们可以创建触发器，保证这 2 个表中删除其中一个表的一条记录的操作，自动触发删除另外一个表中对应的记录的操作。这样一来，就防止了一个表中的记录在另外一个表中不存在的情况，也就确保了数据的一致性。

创建触发器的代码如下所示：

 复制代码

```
1 DELIMITER //
```

```
2 CREATE TRIGGER operation.del_goods_o BEFORE DELETE -- 在删除前触发
```

```
3 ON operation.goods_o
```

```
4 FOR EACH ROW -- 表示每删除一条记录，触发一次
```

```
5 BEGIN -- 开始程序体
```

```
6 DELETE FROM operation.goods_f
```

```
7 WHERE groupnumber=OLD.groupnumber
```

```
8 AND itemnumber=OLD.itemnumber;
```

```
9 END
```

```
10 //
```

```
11 DELIMITER ;
```

```
12
```

```
13 DELIMITER //
```

```
14 CREATE TRIGGER operation.del_goodsf BEFORE DELETE -- 在删除前触发
```

```
15 ON operation.goods_f
```

```
16 FOR EACH ROW -- 表示每删除一条记录，触发一次
```

```
17 BEGIN -- 开始程序体
```

```
18 DELETE FROM operation.goods_o
```

```
19 WHERE groupnumber=OLD.groupnumber
```

```
20 AND itemnumber=OLD.itemnumber;
```

```
21 END
```

```
22 //
```

```
23 DELIMITER ;
```

到这儿呢，数据库数据表以及相关的索引、存储过程和触发器就都创建完了。可以说，我们已经完成了数据库的设计。

但是，在实际工作中，如果你只进行到这一步就打住了，那就还不能算是一个优秀的开发者。因为你考虑问题还不够全面。一个合格的系统设计者，不但要准确把握客户的需求，预见项目实施的前景，还要准备好对任何可能意外的应对方案。实际做项目时，不是纸上谈兵，什么情况都可能发生，我们需要未雨绸缪。所以，下面我们就来设计下项目的容灾和备份策略。

如何制定容灾和备份策略？

为了防止灾害出现，我设置了主从架构。为了方便你理解，我采用的是一主一从的架构，你也可以搭建一主多从的架构，原理都是一样的。

主从架构的核心是，从服务器实时自动同步主服务器的数据，一旦主服务器宕机，可以切换到从服务器继续使用。这样就可以把灾害损失降到最低。

下面我就和你一起，搭建一下主从服务器。

如何搭建主从服务器？

第一步，确保从服务器可以访问主服务器（在同一网段），例如，可以把主服务器的 IP 地址设置为：

```
1 主服务器IP: 192.168.1.100
```

[复制代码](#)

需要注意的是，主服务器入口方向的 3306 号端口需要打开，否则从服务器无法访问主服务器的 MySQL 服务器。

同时，我们把从服务器的 IP 地址设置为：

```
1 从服务器IP: 192.168.1.110
```

[复制代码](#)

第二步，修改主从服务器的系统配置文件 my.ini，使主从服务器有不同的 ID 编号，并且指定需要同步的数据库。

在主服务器的配置文件中，我们把主服务器的编号修改为：server-id = 1。

```
1 # ***** Group Replication Related *****
2 # Specifies the base name to use for binary log files. With binary logging
3 # enabled, the server logs all statements that change data to the binary
4 # log, which is used for backup and replication.
5 log-bin=mysql-bin          -- 二进制日志名称
6 binlog-do-db = operation   -- 需要同步的数据库：营运数据库
7 binlog-do-db = inventory   -- 需要同步的数据库：库存数据库
8
9 # ***** Group Replication Related *****
10 # Specifies the server ID. For servers that are used in a replication topology
```

[复制代码](#)

```
11 # you must specify a unique server ID for each replication server, in the
12 # range from 1 to 2^32 - 1. "Unique" means that each ID must be different
13 # from every other ID in use by any other source or replica.
14 server-id=1          -- 主服务器的ID设为1
15
```

然后，我们来修改从服务器的配置文件 my.ini，把从服务器的编号设置为 server-id = 2。

[复制代码](#)

```
1 # ***** Group Replication Related *****
2 # Specifies the base name to use for binary log files. With binary logging
3 # enabled, the server logs all statements that change data to the binary
4 # log, which is used for backup and replication.
5 log-bin=mysql-bin          -- 二进制日志名称
6 replicate_do_db = operation -- 需要同步过来的数据库：营运数据库
7 replicate_do_db = inventory -- 需要同步过来的数据库：库存数据库
8
9 # ***** Group Replication Related *****
10 # Specifies the server ID. For servers that are used in a replication topology
11 # you must specify a unique server ID for each replication server, in the
12 # range from 1 to 2^32 - 1. "Unique" means that each ID must be different
13 # from every other ID in use by any other source or replica.
14 server-id=2                -- 从服务器的编号为2
```

第三步，在主从服务器上都保存配置文件，然后分别重启主从服务器上的 MySQL 服务器。

第四步，为了使从服务器可以访问主服务器，在主服务器上创建数据同步用户，并赋予所有权限。这样，从服务器就可以实时读取主服务器的数据了。

[复制代码](#)

```
1 mysql> CREATE USER 'myreplica'@'%' IDENTIFIED BY 'mysql';
2 Query OK, 0 rows affected (0.02 sec)
3
4 mysql> GRANT ALL ON *.* TO 'myreplica'@'%';
5 Query OK, 0 rows affected (0.99 sec)
```

第五步，在从服务器上启动数据同步，开始从主服务器中同步数据。


[复制代码](#)

```
1 mysql>change master to master_host='192.168.1.100',master_port=3306,master_use
```

```
2 Query OK, 0 rows affected (0.02 sec)
```

启动同步的时候，你需要注意的是，必须指明主服务器上二进制日志中的位置 `master_log_pos`。也就是说，你准备从主服务器的二进制日志的哪个位置开始同步数据。你可以通过在主服务器上，用 SQL 语句 “`SHOW BINLOG EVENTS IN 二进制日志名`” 获取这个值。下面的代码可以启动同步：

```
1 mysql>start slave;
2 Query OK, 0 rows affected (0.02 sec)
```

 复制代码


如何制定数据备份策略？

设置了主从服务器，也不是万无一失。

我曾经就遇到过这样一件事：我们把主从服务器搭在了某大厂几台不同的云服务器上，自以为没问题了，没想到大厂也有失手的时候，居然整个地区全部宕机，导致我们的主从服务器同时无法使用，近千家商户无法开展业务，损失惨重。

所以，无论系统的架构多么可靠，我们也不能大意。备份仍然是必不可少的步骤。我们可以在应用层面调用类似下面的命令进行备份：

```
1 H:\>mysqldump -u root -p --databases
2 inventory operation > H:\backup\Monday\mybackup.sql
```

 复制代码

我在项目中设定的策略是，每天晚上 12:00 做一个自动备份，循环备份 7 天，创建 7 个文件夹，从 Monday 到 Sunday，每个文件夹中保存对应的备份文件，新的覆盖旧的。

这个逻辑也很简单，你很容易理解，我就不多解释了，你不要忘了做这一步工作就可以了。

总结

今天这节课，我给你详细讲解了建库建表、创建索引、存储过程、触发器，以及容灾和备份策略。有几点你需要格外重视一下。

索引是提升查询执行速度的关键，创建的原则是：所有的数据表都要创建索引；有可能作为筛选条件的字段，都要用来创建索引。

另外，容灾和备份是数据库系统设计中必不可少的部分。因为在现实生活中，什么情况都可能发生，我们无法预见，但是可以尽量避免。在设计阶段的未雨绸缪，可以帮助我们减少很多损失。

最后我要提醒你的是，MySQL 的相关知识实践性非常强，决不能停留在纸面上。我在课中演示的代码，都是在实际环境中运行过的，你课下一定要跟着实际操作一下。毕竟，学习知识最好的办法，就是在解决实际问题中学习。

思考题

在今天的课程中，我演示了搭建主从服务器的过程。其中，在第四步，我专门创建了一个用来同步数据的账号“myreplica”。我想请你思考一下，我为什么要这样做？直接用“root”账号不行吗？

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，也欢迎你把它分享给你的朋友或同事，我们下节课见。

提建议

更多学习推荐



175 道 Go 工程师 大厂常考面试题

限量免费领取



© 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 27 | 手把手带你设计一个完整的连锁超市信息系统数据库（上）

下一篇 特别发送（一） | 经典面试题讲解第一弹

精选留言 (2)

[写留言](#)**朱晓峰** 置顶

2021-05-20

你好，我是朱晓峰，下面我就来公布一下这节课思考题的答案：

这节课，我们学习了设计信息系统数据库（下）。下面是思考题的答案：

尽量不要在设置主从服务器时使用root 账号，原因主要是出于安全考虑，设置主服务器...
展开

**朱晓峰** 置顶

2021-05-20

你好，我是朱晓峰，下面我就来公布一下上节课思考题的答案：

上节课，我们学习了设计信息系统数据库（上）。下面是思考题的答案：

可以考虑垂直分表： ...
展开 ▼

