

NSD DATABASE DAY02

1. [MySQL索引创建与删除](#)
2. [MySQL存储引擎的配置](#)

1 MySQL索引创建与删除

1.1 问题

本案例要求熟悉MySQL索引的类型及操作方法，主要练习以下任务：

- 普通索引、唯一索引、主键索引的创建/删除
- 自增主键索引的创建/删除
- 建立员工表yg、工资表gz，数据内容如表-1、表-2所示，设置外键实现同步更新与同步删除

表-1 员工表yg的数据

yg_id	name
1	Jerry
2	Tom

表-2 工资表gz的数据

gz_id	Name	gz
1	Jerry	12000
2	Tom	8000

1.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：索引的创建与删除

创建表的时候指定INDEX索引字段

创建库home：

01. my sql> create database home;
02. Query OK, 1 row affected (0.00 sec)

允许有多个INDEX索引字段。比如，以下操作在home库中创建了tea4表，将其中的id、name作为索引字段：

01. my sql> USE home;
02. Database changed

[Top](#)

```

03.  my sql> CREATE TABLE tea4(
04.      -> id char( 6) NOT NULL,
05.      -> name varchar( 6) NOT NULL,
06.      -> age int( 3) NOT NULL,
07.      -> gender ENUM( 'boy','girl') DEFAULT 'boy',
08.      -> INDEX( id),INDEX( name)
09.      -> );
10.  Query OK, 0 rows affected ( 0.59 sec)

```

查看新建tea4表的字段结构，可以发现两个非空索引字段的KEY标志为MUL：

```

01.  my sql> DESC tea4;
02.  +-----+-----+-----+-----+-----+
03.  | Field | Type          | Null | Key | Default | Extra |
04.  +-----+-----+-----+-----+-----+
05.  | id    | char( 6)      | NO   | MUL | NULL    |      |
06.  | name  | varchar( 6)   | NO   | MUL | NULL    |      |
07.  | age   | int( 3)       | NO   |     | NULL    |      |
08.  | gender| enum( 'boy','girl') | YES  |     | boy     |      |
09.  +-----+-----+-----+-----+-----+
10.  4 rows in set ( 0.00 sec)

```

2) 删除现有表的某个INDEX索引字段

比如，删除tea4表中名称为named的INDEX索引字段：

```

01.  my sql> drop INDEX name ON tea4;           //删除name字段的索引
02.  Query OK, 0 rows affected ( 0.18 sec)
03.  Records: 0 Duplicates: 0 Warnings: 0
04.
05.  my sql> DESC tea4;                         //确认删除结果
06.  +-----+-----+-----+-----+-----+
07.  | Field | Type          | Null | Key | Default | Extra |
08.  +-----+-----+-----+-----+-----+
09.  | id    | char( 6)      | NO   | MUL | NULL    |      |
10.  | name  | varchar( 6)   | NO   |     | NULL    |      |
11.  | age   | int( 3)       | NO   |     | NULL    |      |
12.  | gender| enum( 'boy','girl') | YES  |     | boy     |      |
13.  +-----+-----+-----+-----+-----+
14.  4 rows in set ( 0.00 sec)

```

[Top](#)

3) 在已有的某个表中设置INDEX索引字段

比如，针对tea4表的age字段建立索引，名称为 nianling：

```
01.  my sql> CREATE INDEX nianling ON tea4( age);    //针对指定字段创建索引
02.  Query OK, 0 rows affected (0.62 sec)
03.  Records: 0 Duplicates: 0 Warnings: 0
04.
05.  my sql> DESC tea4;                            //确认创建结果
06.  +-----+-----+-----+-----+-----+-----+
07.  | Field | Type          | Null | Key | Default | Extra |
08.  +-----+-----+-----+-----+-----+-----+
09.  | id    | char(6)       | NO   | MUL | NULL    |      |
10.  | name  | varchar(6)    | NO   |     | NULL    |      |
11.  | age   | int(3)        | NO   | MUL | NULL    |      |
12.  | gender| enum('boy','girl')| YES  |     | boy     |      |
13.  +-----+-----+-----+-----+-----+-----+
14.  4 rows in set (0.00 sec)
```

4) 查看指定表的索引信息

使用SHOW INDEX 指令：

```
01.  my sql> SHOW INDEX FROM tea4\G
02.  ***** 1 row *****
03.      Table: tea4
04.      Non_unique: 1
05.      Key_name: id
06.      Seq_in_index: 1
07.      Column_name: id
08.      Collation: A
09.      Cardinality: 0
10.      Sub_part: NULL
11.      Packed: NULL
12.      Null:
13.      Index_type: BTREE                //使用B树算法
14.      Comment:
15.      Index_comment:
16.  ***** 2. row *****
17.      Table: tea4
```

[Top](#)

```

18.      Non_unique: 1
19.      Key_name: nianling           //索引名称
20.      Seq_in_index: 1
21.      Column_name: age             //字段名称
22.      Collation: A
23.      Cardinality: 0
24.      Sub_part: NULL
25.      Packed: NULL
26.      Null:
27.      Index_type: BTREE
28.      Comment:
29.      Index_comment:
30.      2 rows in set ( 0.00 sec)

```

5) 创建表的时候指定UNIQUE索引字段

UNIQUE表示唯一性的意思，同一个表中可以有多个字段具有唯一性。

比如，创建tea5表，将id、name字段建立设置UNIQUE索引，age字段设置INDEX索引：

```

01.  my sql> CREATE TABLE tea5(
02.      -> id char( 6) ,
03.      -> name varchar( 4) NOT NULL,
04.      -> age int( 3) NOT NULL,
05.      -> UNIQUE( id) ,UNIQUE( name) ,INDEX( age)
06.      -> );
07.  Query OK, 0 rows affected ( 0.30 sec)

```

查看新建tea5表的字段结构，可发现UNIQUE字段的KEY标志为UNI；另外，由于字段name必须满足“NOT NULL”的非空约束，所以将其设置为UNIQUE后会自动变成了PRIMARY KEY主键字段：

```

01.  my sql> DESC tea5;           //确认设置结果
02.  +-----+-----+-----+-----+-----+
03.  | Field | Type      | Null | Key | Default | Extra |
04.  +-----+-----+-----+-----+-----+
05.  | id    | char( 6)  | YES  | UNI | NULL    |      |
06.  | name  | varchar( 4) | NO   | PRI | NULL    |      |
07.  | age   | int( 3)   | NO   | MUL | NULL    |      |
08.  +-----+-----+-----+-----+-----+
09.  3 rows in set ( 0.03 sec)

```

[Top](#)

6) 删除UNIQUE索引、在已有的表中设置UNIQUE索引字段

先删除tea5表name字段的唯一索引（与删除INDEX索引的方法相同）：

```
01.  my sql> DROP INDEX name ON tea5;           //清除UNIQUE索引
02.  Query OK, 0 rows affected (0.97 sec)
03.  Records: 0 Duplicates: 0 Warnings: 0
04.
05.  my sql> DESC tea5;                         //确认删除结果
06.  +-----+-----+-----+-----+-----+
07.  | Field | Type   | Null | Key | Default | Extra |
08.  +-----+-----+-----+-----+-----+
09.  | id    | char(6) | YES  | UNI | NULL    |      |
10.  | name  | varchar(4) | NO   |     | NULL    |      |
11.  | age   | int(3)  | NO   | MUL | NULL    |      |
12.  +-----+-----+-----+-----+-----+
13.  3 rows in set (0.00 sec)
```

重新为tea5表的name字段建立UNIQUE索引，并确认结果：

```
01.  my sql> CREATE UNIQUE INDEX name ON tea5( name); //建立UNIQUE索引
02.  Query OK, 0 rows affected (0.47 sec)
03.  Records: 0 Duplicates: 0 Warnings: 0
04.
05.  my sql> DESC tea5;                         //确认设置结果
06.  +-----+-----+-----+-----+-----+
07.  | Field | Type   | Null | Key | Default | Extra |
08.  +-----+-----+-----+-----+-----+
09.  | id    | char(6) | YES  | UNI | NULL    |      |
10.  | name  | varchar(4) | NO   | PRI | NULL    |      |
11.  | age   | int(3)  | NO   | MUL | NULL    |      |
12.  +-----+-----+-----+-----+-----+
13.  3 rows in set (0.00 sec)
```

7) 建表时设置PRIMARY KEY主键索引

主键索引实际上在前面已经接触过了，建表的时候可以直接指定。如果表内一开始没有主键字段，则新设置的非空UNIQUE字段相当于具有PRIMARY KEY主键约束。[Top](#)

每个表中的主键字段只能有一个。

建表的时候，可以直接在某个字段的“约束条件”部分指定PRIMARY KEY；也可以在最后指定PRIMARY KEY(某个字段名)。比如：

```
01. my sql> CREATE TABLE biao01(  
02.     - > id int( 4) PRIMARY KEY,           //直接在字段定义时约束  
03.     - > name varchar( 8)  
04.     - > );  
05. Query OK, 0 rows affected ( 0.19 sec)
```

或者：

```
01. my sql> CREATE TABLE biao02(  
02.     - > id int( 4),  
03.     - > name varchar( 8),  
04.     - > PRIMARY KEY( id)                   //所有字段定义完，最后指定  
05.     - > );  
06. Query OK, 0 rows affected ( 0.17 sec)
```

在建表的时候，如果主键字段为int类型，还可以为其设置AUTO_INCREMENT自增属性，这样当添加新的表记录时，此字段的值会自动从1开始逐个增加，无需手动指定。比如，新建一个tea6表，将id列作为自增的主键字段：

```
01. my sql> CREATE TABLE tea6(  
02.     - > id int( 4) AUTO_INCREMENT,  
03.     - > name varchar( 4) NOT NULL,  
04.     - > age int( 2) NOT NULL,  
05.     - > PRIMARY KEY( id)  
06.     - > );  
07. Query OK, 0 rows affected ( 0.29 sec)
```

8) 删除现有表的PRIMARY KEY主键索引

如果要移除某个表的PRIMARY KEY约束，需要通过ALTER TABLE指令修改。比如，以下操作将清除biao01表的主键索引。

清除前（主键为id）：

```
01. my sql> DESC biao01;  
02. +-----+-----+-----+-----+-----+-----+
```

[Top](#)

```

03. | Field | Type      | Null | Key | Default | Extra |
04. +-----+-----+-----+-----+-----+-----+
05. | id   | int(4)    | NO   | PRI | NULL    |       |
06. | name | varchar(8)| YES  |     | NULL    |       |
07. +-----+-----+-----+-----+-----+-----+
08. 2 rows in set (0.00 sec)

```

清除操作：

```

01. my sql> ALTER TABLE biao01 DROP PRIMARY KEY;
02. Query OK, 0 rows affected (0.49 sec)
03. Records: 0 Duplicates: 0 Warnings: 0

```

清除后（无主键）：

```

01. my sql> DESC biao01;
02. +-----+-----+-----+-----+-----+-----+
03. | Field | Type      | Null | Key | Default | Extra |
04. +-----+-----+-----+-----+-----+-----+
05. | id   | int(4)    | NO   |     | NULL    |       |
06. | name | varchar(8)| YES  |     | NULL    |       |
07. +-----+-----+-----+-----+-----+-----+
08. 2 rows in set (0.00 sec)

```

当尝试删除tea6表的主键时，会出现异常：

```

01. my sql> ALTER TABLE tea6 DROP PRIMARY KEY;
02. ERROR 1075 (42000): Incorrect table definition; there can be only one auto column and it i

```



这是因为tea6表的主键字段id具有AUTO_INCREMENT自增属性，提示这种字段必须作为主键存在，因此若要清除此主键必须先清除自增属性——修改id列的字段定义：

```

01. my sql> ALTER TABLE tea6 MODIFY id int(4) NOT NULL;
02. Query OK, 0 rows affected (0.75 sec)
03. Records: 0 Duplicates: 0 Warnings: 0

```

[Top](#)

然后再清除主键属性就OK了：

```
01. my sql> ALTER TABLE tea6 DROP PRIMARY KEY;           //清除主键
02. Query OK, 0 rows affected ( 0.39 sec)
03. Records: 0 Duplicates: 0 Warnings: 0
04.
05. my sql> desc tea6;                                     //确认清除结果
06. +-----+-----+-----+-----+-----+-----+
07. | Field | Type      | Null | Key | Default | Extra |
08. +-----+-----+-----+-----+-----+-----+
09. | id    | int(4)    | NO   |     | NULL    |       |
10. | name  | varchar(4)| NO   |     | NULL    |       |
11. | age   | int(2)    | NO   |     | NULL    |       |
12. +-----+-----+-----+-----+-----+-----+
13. 3 rows in set ( 0.01 sec)
```

9) 为现有表添加PRIMARY KEY主键索引

重新为tea6表指定主键字段，仍然使用id列：

```
01. my sql> ALTER TABLE tea6 ADD PRIMARY KEY(id);        //设置主键字段
02. Query OK, 0 rows affected ( 0.35 sec)
03. Records: 0 Duplicates: 0 Warnings: 0
04.
05. my sql> DESC tea6;                                     //确认设置结果
06. +-----+-----+-----+-----+-----+-----+
07. | Field | Type      | Null | Key | Default | Extra |
08. +-----+-----+-----+-----+-----+-----+
09. | id    | int(4)    | NO   | PRI | NULL    |       |
10. | name  | varchar(4)| NO   |     | NULL    |       |
11. | age   | int(2)    | NO   |     | NULL    |       |
12. +-----+-----+-----+-----+-----+-----+
13. 3 rows in set ( 0.00 sec)
```

步骤二：创建数据库并设置外键实现同步更新与同步删除

根据实验任务要求，两个表格的字段结构如表-1、表-2所示。

1) 创建yg表，用来记录员工工号、姓名

其中yg_id列作为主键，并设置自增属性

[Top](#)


```

01. my sql> CREATE TABLE yg(
02.     -> yg_id int( 4) AUTO_INCREMENT,
03.     -> name char( 16) NOT NULL,
04.     -> PRIMARY KEY(yg_id)
05.     -> );
06. Query OK, 0 rows affected ( 0.15 sec)

```

2) 创建gz表，用来记录员工的工资信息

其中gz_id需要参考员工工号，即gz表的gz_id字段设为外键，将yg表的yg_id字段作为参考键：

```

01. my sql> CREATE TABLE gz(
02.     -> gz_id int( 4) NOT NULL,
03.     -> name char( 16) NOT NULL,
04.     -> gz float( 7,2) NOT NULL DEFAULT 0,
05.     -> INDEX( name),
06.     -> FOREIGN KEY( gz_id) REFERENCES yg( yg_id)
07.     -> ON UPDATE CASCADE ON DELETE CASCADE
08.     -> );
09. Query OK, 0 rows affected ( 0.23 sec)

```

3) 为yg表添加2条员工信息记录

因yg_id有AUTO_INCREMENT属性，会自动填充，所以只要为name列赋值就可以了。

插入表记录可使用INSERT指令，这里先执行下列操作，具体在下一章学习：

```

01. my sql> INSERT INTO yg( name) VALUES( 'Jerry' ),( 'Tom' );
02. Query OK, 2 rows affected ( 0.16 sec)
03. Records: 2 Duplicates: 0 Warnings: 0

```

确认yg表的数据记录：

```

01. my sql> SELECT * FROM yg;
02. +-----+-----+
03. | yg_id | name |
04. +-----+-----+
05. | 1 | Jerry |

```

[Top](#)

```

06.      |      2 | Tom      |
07.      +-----+-----+
08.      2 rows in set (0.00 sec)

```

4) 为gz表添加2条工资信息记录

同上，数据参考图-2，插入相应的工资记录（gz_id字段未指定默认值，也未设置自增属性，所以需要手动赋值）：

```

01.      my sql> INSERT INTO gz( gz_id,name,gz)
02.          -> VALUES( 1,'Jerry',12000),( 2,'Tom',8000)
03.          ->;
04.      Query OK, 2 rows affected (0.06 sec)
05.      Records: 2 Duplicates: 0 Warnings: 0

```

确认gz表的数据记录：

```

01.      my sql> SELECT * FROM gz;
02.      +-----+-----+-----+
03.      | gz_id | name | gz      |
04.      +-----+-----+-----+
05.      |      1 | Jerry | 12000.00 |
06.      |      2 | Tom  | 8000.00  |
07.      +-----+-----+-----+
08.      2 rows in set (0.05 sec)

```

5) 验证表记录的UPDATE更新联动

将yg表中Jerry用户的yg_id修改为1234：

```

01.      my sql> update yg SET yg_id=1234 WHERE name='Jerry';
02.      Query OK, 1 row affected (0.05 sec)
03.      Rows matched: 1 Changed: 1 Warnings: 0

```

确认修改结果：

```

01.      my sql> SELECT * FROM yg;
02.      +-----+-----+

```

[Top](#)

```

03.  | yg_id | name |
04.  +-----+-----+
05.  |    2 | Tom  |
06.  | 1234 | Jerry |
07.  +-----+-----+
08.  2 rows in set (0.00 sec)

```

同时也会发现，gz表中Jerry用户的gz_id也跟着变了：

```

01.  my sql> SELECT * FROM gz;
02.  +-----+-----+-----+
03.  | gz_id | name | gz      |
04.  +-----+-----+-----+
05.  | 1234 | Jerry | 12000.00 |
06.  |    2 | Tom  | 8000.00  |
07.  +-----+-----+-----+
08.  2 rows in set (0.00 sec)

```

6) 验证表记录的DELETE删除联动
删除yg表中用户Jerry的记录：

```

01.  my sql> DELETE FROM yg WHERE name='Jerry';
02.  Query OK, 1 row affected (0.05 sec)

```

确认删除结果：

```

01.  my sql> SELECT * FROM yg;
02.  +-----+-----+
03.  | yg_id | name |
04.  +-----+-----+
05.  |    2 | Tom  |
06.  +-----+-----+
07.  1 row in set (0.00 sec)

```

查看gz表中的变化（Jerry的记录也没了）：

[Top](#)

```

01.  my sql> SELECT * FROM gz;
02.  +-----+-----+-----+
03.  | gz_id | name | gz      |
04.  +-----+-----+-----+
05.  |    2 | Tom  | 8000.00 |
06.  +-----+-----+-----+
07.  1 row in set ( 0.00 sec)

```

7) 删除指定表的外键约束

先通过SHOW指令获取表格的外键约束名称：

```

01.  my sql> SHOW CREATE TABLE gz\G
02.  ***** 1 row *****
03.      Table: gz
04.  Create Table: CREATE TABLE `gz` (
05.    `gz_id` int( 4) NOT NULL,
06.    `name` char( 16) NOT NULL,
07.    `gz` float( 7,2) NOT NULL DEFAULT '0.00',
08.    KEY `name` (`name`),
09.    KEY `gz_id` (`gz_id`),
10.    CONSTRAINT `gz_ibfk_1` FOREIGN KEY (`gz_id`) REFERENCES `yg` (`yg_id`) ON DELETE
11.    ) ENGINE=InnoDB DEFAULT CHARSET=utf8
12.  1 row in set ( 0.00 sec)

```

其中gz_ibfk_1即删除外键约束时要用到的名称。

删除操作：

```

01.  my sql> ALTER TABLE gz DROP FOREIGN KEY gz_ibfk_1;
02.  Query OK, 0 rows affected ( 0.01 sec)
03.  Records: 0 Duplicates: 0 Warnings: 0

```

确认删除结果：

```

01.  my sql> SHOW CREATE TABLE gz\G
02.  ***** 1 row *****
03.      Table: gz

```

[Top](#)

```

04. Create Table: CREATE TABLE `gz` (
05.   `gz_id` int(4) NOT NULL,
06.   `name` char(16) NOT NULL,
07.   `gz` float(7,2) NOT NULL DEFAULT '0.00',
08.   KEY `name` (`name`),
09.   KEY `gz_id` (`gz_id`)
10. ) ENGINE=InnoDB DEFAULT CHARSET=utf8
11. 1 row in set (0.00 sec)

```

2 MySQL存储引擎的配置

2.1 问题

本案例要求MySQL数据存储引擎的使用，完成以下任务操作：

- 可用的存储引擎类型
- 查看默认存储类型
- 更改表的存储引擎

2.2 步骤

实现此案例需要按照如下步骤进行。

步骤一：查看存储引擎信息

登入MySQL服务器，查看当前支持哪些存储引擎。

使用mysql命令连接，以root用户登入：

```

01. [root@dbvr1 ~]# mysql -u root -p
02. Enter password:
03. Welcome to the MySQL monitor.  Commands end with ; or \g.
04. Your MySQL connection id is 9
05. Server version: 5.7.17 MySQL Community Server (GPL)
06.
07. Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
08.
09. Oracle is a registered trademark of Oracle Corporation and/or its
10. affiliates. Other names may be trademarks of their respective
11. owners.
12.
13. Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14.
15. mysql>

```

[Top](#)

执行SHOW ENGINES\G指令可列表查看，MySQL 5.6可用的存储引擎有9种（除最后的FEDERATED以外，其他8种都支持），其中默认采用的存储引擎为InnoDB：

```
01.  my sql> SHOW ENGINES\G
02.  ***** 1. row *****
03.      Engine: InnoDB
04.      Support: DEFAULT //此存储引擎为默认
05.      Comment: Supports transactions, row- level locking, and foreign keys
06.      Transactions: YES
07.      XA: YES
08.      Savepoints: YES
09.  ***** 2. row *****
10.      Engine: MRG_MYISAM
11.      Support: YES
12.      Comment: Collection of identical MyISAM tables
13.      Transactions: NO
14.      XA: NO
15.      Savepoints: NO
16.  ***** 3. row *****
17.      Engine: MEMORY
18.      Support: YES
19.      Comment: Hash based, stored in memory, useful for temporary tables
20.      Transactions: NO
21.      XA: NO
22.      Savepoints: NO
23.  ***** 4. row *****
24.      Engine: BLACKHOLE
25.      Support: YES
26.      Comment: /dev/null storage engine ( anything you write to it disappears)
27.      Transactions: NO
28.      XA: NO
29.      Savepoints: NO
30.  ***** 5. row *****
31.      Engine: MyISAM
32.      Support: YES
33.      Comment: MyISAM storage engine
34.      Transactions: NO
35.      XA: NO
36.      Savepoints: NO
37.  ***** 6. row *****
38.      Engine: CSV
```

[Top](#)

```

39.      Support: YES
40.      Comment: CSV storage engine
41. Transactions: NO
42.      XA: NO
43.      Savepoints: NO
44. ***** 7. row *****
45.      Engine: ARCHIVE
46.      Support: YES
47.      Comment: Archive storage engine
48. Transactions: NO
49.      XA: NO
50.      Savepoints: NO
51. ***** 8. row *****
52.      Engine: PERFORMANCE_SCHEMA
53.      Support: YES
54.      Comment: Performance Schema
55. Transactions: NO
56.      XA: NO
57.      Savepoints: NO
58. ***** 9. row *****
59.      Engine: FEDERATED
60.      Support: NO           //此引擎不被支持
61.      Comment: Federated MySQL storage engine
62. Transactions: NULL
63.      XA: NULL
64.      Savepoints: NULL
65. 9 rows in set (0.01 sec)

```

或者直接查看系统变量default_storage_engine 的值，也可确认默认采用的存储引擎是InnoDB：

```

01.  my sql> SHOW VARIABLES LIKE 'default_storage_engine';
02.  +-----+-----+
03.  | Variable_name | Value |
04.  +-----+-----+
05.  | default_storage_engine | InnoDB |
06.  +-----+-----+
07.  1 row in set (0.00 sec)

```

[Top](#)

步骤二：修改默认存储引擎

在 mysql> 环境中，可以直接通过SET指令更改默认的存储引擎（只在本次连接会话过程中有效，退出重进即失效）。比如临时修改为MyISAM，可执行下列操作：

```
01.  my sql> SET default_storage_engine=MyISAM;           //改用MyISAM引擎
02.  Query OK, 0 rows affected (0.00 sec)
03.
04.  my sql> SHOW VARIABLES LIKE 'default_storage_engine'; //确认结果
05.  +-----+-----+
06.  | Variable_name | Value |
07.  +-----+-----+
08.  | default_storage_engine | MyISAM |
09.  +-----+-----+
10.  1 row in set (0.00 sec)
```

若希望直接修改MySQL服务程序所采用的默认存储引擎，应将相关设置写入配置文件/etc/my.cnf，并重启服务后生效。比如：

```
01.  [ root@dbsvr1 ~] # vim /etc/my.cnf
02.  [ my sql d]
03.  ... ..
04.  default_storage_engine=MEMORY           //改用MEMORY引擎
05.
06.  [ root@dbsvr1 ~] # systemctl restart mysqld.service //重启服务
```

重新登入 mysql> 确认修改结果：

```
01.  [ root@dbsvr1 ~] # mysql -u root -p
02.  Enter password:
03.  Welcome to the MySQL monitor. Commands end with ; or \g.
04.  Your MySQL connection id is 3
05.  Server version: 5.7.17 MySQL Community Server (GPL)
06.
07.  Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.
08.
09.  Oracle is a registered trademark of Oracle Corporation and/or its
10.  affiliates. Other names may be trademarks of their respective
11.  owners.
12.
```

[Top](#)


```

13.   Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
14.
15.
16.   my sql> SHOW VARIABLES LIKE 'default_storage_engine';
17.   +-----+-----+
18.   | Variable_name      | Value |
19.   +-----+-----+
20.   | default_storage_engine | MEMORY |           //默认引擎已修改
21.   +-----+-----+
22.   1 row in set ( 0.00 sec)
23.
24.   my sql> exit
25.   Bye

```

为了避免后续实验障碍，测试完后记得恢复原状——移除默认引擎设置，或者将其修改为 InnoDB 即可：

```

01.   [ root@dbsvr1 ~] # vim /etc/my.cnf
02.   [ my sql]
03.   ...
04.   default_storage_engine=InnoDB
05.   [ root@dbsvr1 ~] # systemctl restart my sqld.service

```

确认恢复结果（选项 -e 可调用指定的 SQL 操作后返回 Shell 命令行）：

```

01.   [ root@dbsvr1 ~] # my sql - u root - p - e "SHOW VARIABLES LIKE 'default_storage_engine';
02.   Enter password:
03.   +-----+-----+
04.   | Variable_name      | Value |
05.   +-----+-----+
06.   | default_storage_engine | InnoDB |
07.   +-----+-----+

```

