# 2021

# 数据库

SHUJUKU

微信公众号：灰灰考研

# 第七章 索引、视图和触发器

- 理解索引的概念和分类
- 掌握索引的创建
- 了解查看索引信息的方式
- 掌握创建视图
- 掌握管理视图
- 了解通过视图修改数据
- 了解触发器的概念
- 掌握创建、删除触发器
- 了解触发器的常见应用

## 一、索引

在数据库操作中，用户经常需要查找特定的数据，而索引则用来快速寻找那些具有特定值的记录。例如，当执行“select*from student where sno='11101001'”语句时，如果没有索引，MySQL数据库必须从第一条记录开始扫描表，直至找到sno字段值是11101001的记录。表里面的记录数量越多，这个操作花费的时间代价就越高。如果在作为搜索条件的字段上创建了索引，MySQL在查找时，无需扫描任何记录即可迅速得到目标记录所在的位置，能大大提高查找的效率。

### （1）索引概述

如果把数据表看作一本书，则表的索引就如同书的目录一样，可以大大提高查询速度，改善数据库的性能。其具体表现如下：

① 唯一性索引可以保证数据记录的唯一性。

② 可以加快数据的检索速度。

③ 可以加快表与表之间的连接，这一点在实现数据的参照完整性方面有特别的意义。

④ 在使用ORDER BY和GROUP BY子句进行数据检索时，可以显著减少查询中分组和排序的时间。

⑤ 在检索数据的过程中使用优化隐藏器，可以提高系统性能。

（2）索引的分类

① 普通索引是由KEY或INDEX定义的索引，它是MySQL中的基本索引类型，允许在定义索引的字段中插入重复值和空值。该类型索引可以创建在任何数据类型中。

② 唯一索引是由UNIQUE定义的索引，指索引字段的值必须唯一，但允许有空值。如果是在多个字段上建立的组合索引，则字段值的组合必须唯一。在创建主键和唯一约束的字段上会自动创建唯一索引。主键索引是一种特殊的唯一索引，不允许有空值。

③ 全文索引是由FULLTEXT定义的索引，是指在定义索引的字段上支持值的全文查找。该索引类型允许在索引字段上插入重复值和空值，它只能创建在CHAR、VARCHAR或TEXT类型的字段上。旧版的MySQL只有MyISAM存储引擎支持全文索引, MySQL 5.6版本后InnoDB存储引擎也支持全文索引。

④ 空间索引是由SPATIAL定义的索引，是只能在空间数据类型的字段上建立的索引。MySQL中空间数据类型有4种，分别是GEOMETRY、POINT、LINESTRING和POLYGON。需要注意的是，创建空间索引的字段，必须将其声明为NOT NULL。旧版的MySQL只有MyISAM存储引擎支持空间索引，从MySQL 5.7.4实验室版本开始，InnoDB存储引擎新增了对空间索引的支持功能。

⑤ 单列索引指在表中单个字段上创建的索引。它可以是普通索引、唯一索引或者全文索引，只要保证该索引只对应表中的一个字段即可。

⑥ 多列索引指在表中多个字段上创建的索引。只有在查询条件中使用了这些字段中的第一个字段时，该索引才会被使用。例如在学生表的“学号”“姓名”和“班级”字段上创建一个多列索引，那么，只有在查询条件中使用了“学号”字段时，该索引才会被使用。

（3）索引的设计原则

索引设计不合理或缺少索引都会给数据库的应用造成障碍。高效的索引对于用户获得良好的性能体验非常重要。设计索引时，应该考虑以下原则：

① 索引并非越多越好一个表中如有大量的索引，不仅占用磁盘空间，而且会影响INSERT、UPDATE、DELETE等语句的性能。因为在更改表中的数据的同时，索引也会进行调整和更新。

② 避免对经常更新的表建立过多的索引避免对经常更新的表建立过多的索引，并且索引中的字段要尽可能得少。对经常查询的字段应该建立索引，但要避免对不必要的字段建立索引。

③ 数据量小的表最好不要使用索引由于数据较少，查询花费的时间可能比遍历索引的时间还要短，索引可能不会产生优化的效果。

④ 在不同值较少的字段上不要建立索引字段中的不同值比较少，例如学生表的“性别”字段，只有“男”和“女”两个值，这样的字段就无须建立索引，建立索引后不但不会提高查询效率，反而会严重降低更新速度。

⑤ 为经常需要进行排序、分组和连接查询的字段建立索引为频繁进行排序或分组的字段和经常进行连接查询的字段创建索引。

（4）创建索引

1. 创建表的时候直接创建索引用CREATE TABLE命令创建表的时候就创建索引，此方式简单、方便。其语法格式如下：

```
CREATE TABLE表名
 (字段名  数据类型[完整性约束条件],
  字段名  数据类型[完整性约束条件],
  …
  字段名 数据类型,
    [UNIQUE|FULLTEXT|SPATIAL]   INDEX|KEY   [别名](字段名[(长度)]) [ASC|DESC]
      );
```

【例】创建teacher_1表，同时在表的tname字段上建立普通索引。SQL语句如下：

```
CREATE TABLE teacher_1
(tno CHAR (4)  NOT NULL,
tname VARCHAR(10)  NOT NULL,
tgender CHAR(1),
tedu VARCHAR(10) ,
tpro VARCHAR(8),
INDEX (tname)
);
```

【例】创建course_1表，在cname字段上创建名为UK_cname的唯一索引，并且按照升序排列。SQL语句如下：

```
 CREATE TABLE course_1
(  cno  CHAR(4) NOT NULL PRIMARY KEY,
   cname VARCHAR(40),
   cperiod INT,
   credit DECIMAL(3,1),
   ctno CHAR(4),
   brief VARCHAR(255),
   UNIQUE INDEX UK_cname(cname ASC)
   );
```

【例】创建course_2表，同时在course_2表中的brief字段上创建名为FT_brief的全文索引。SQL语句如下：

```
CREATE TABLE course_2
(cno CHAR(4)  NOT NULL  PRIMARY  KEY ,
cname VARCHAR(40),
cperiod INT,
credit DECIMAL(3,1) ,
ctno CHAR(4),
brief VARCHAR(255),
FULLTEXT INDEX FT_brief(brief)
)ENGINE=MyISAM;
```

【例】创建一个名为e_0的表，在数据类型为GEOMETRY的space字段上创建空间索引。SQL语句如下：

```
CREATE TABLE e_0
( space GEOMETRY NOT NULL,
 SPATIAL INDEX sp(space)
 ) ENGINE=MyISAM;
```

【例】创建course_3表，同时在course_3表中的cname字段上创建名为IDX_cname的单列索引，索引长度为10。

```
CREATE TABLE course_3(
  cno CHAR(4) NOT NULL PRIMARY KEY,
  cname VARCHAR(40),
  cperiod INT,
  credit DECIMAL(3,1),
  ctno CHAR(4),
  brief VARCHAR(255),
   INDEX IDX_cname(cname(10))
);
```

【例】创建elective_1表，在表中的sno字段和cno字段上建立多列索引。对应的SQL语句如下：

```
CREATE TABLE elective_1(
   sno CHAR(8) NOT NULL,
   cno CHAR(4) NOT NULL,
   score INT,
   INDEX IDX_multi(sno,cno)
   );
```

2. 在已经存在的表上使用CREATE INDEX语句创建索引

CREATE INDEX语句基本的语法形式如下：

```
CREATE [UNIQUE] [FULLTEXT] [SPATIAL] INDEX索引名
ON表名(字段名[(长度)] [ASC | DESC] [,…]);
```

先创建一个没有任何索引的学生表stu。表中包含sno（学号）字段，sname（姓名）字段， sgender（性别）字段，sb
（出生日期）字段，sclass（班级）字段，sresume（简历）字段。

创建stu表的SQL语句如下：

```
CREATE TABLE stu(
sno char(8) NOT NULL,
sname VARCHAR(10) NOT NULL,
sgender CHAR(1),
sbirth DATE,
sclass VARCHAR(20),
sresume VARCHAR(255)
)ENGINE=MyISAM;
```

```
mysql> SHOW CREATE TABLE stu\G
*************************** 1. row ***************************
       Table: stu
Create Table: CREATE TABLE `stu` (
  `sno` char(8) NOT NULL,
  `sname` varchar(10) NOT NULL,
  `sgender` char(1) DEFAULT NULL,
  `sclass` float DEFAULT NULL,
  `sbirth` char(4) DEFAULT NULL,
  `sresume` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`sno`),
  UNIQUE KEY `stu_sno` (`sno`),
  KEY `stu_name` (`sname`),
  KEY `stu_sname_sclass` (`sname`,`sclass`),
  FULLTEXT KEY `stu_sresume` (`sresume`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

【例】创建一个名为e_1的表，在数据类型为GEOMETRY的space字段上创建空间索引。SQL语句如下：

① 创建e_1表。

```
CREATE TABLE e_1
 ( space GEOMETRY NOT NULL,
) ENGINE=MyISAM;
```

② 在e_1表的space字段上创建空间索引。

```
CREATE SPATIAL INDEX sp_space ON e_1(space);
```

3. 在已经存在的表上使用ALTER TABLE语句创建索引 其语法格式如下：

```
ALTER TABLE表名
ADD [UNIQUE|FULLTEXT|SPATIAL] INDEX索引名(字段名[(长度)] [ASC|DE
SC]);
```

【例】在stu表的sname字段上创建名为stu_sname的普通索引。SQL语句如下：

```
ALTER TABLE stu ADD INDEX  stu_sname(sname);
```

```
mysql> SHOW CREATE TABLE stu\G
*************************** 1. row ***************************
       Table: stu
Create Table: CREATE TABLE `stu` (
  `sno` char(8) NOT NULL,
  `sname` varchar(10) NOT NULL,
  `sgender` char(1) DEFAULT NULL,
  `sclass` float DEFAULT NULL,
  `sbirth` char(4) DEFAULT NULL,
  `sresume` varchar(255) DEFAULT NULL,
  UNIQUE KEY `stu_sno` (`sno`),
  KEY `stu_sname` (`sname`),
  KEY `stu_sname_sclass` (`sname`,`sclass`),
  FULLTEXT KEY `stu_sresume` (`sresume`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

【例】创建一个名为e_2的表，在数据类型为GEOMETRY的space字段上创建空间索引。

SQL语句如下：

① 创建e_2表。

```
CREATE TABLE e_2(
space GEOMETRY NOT NULL,
) ENGINE=MyISAM;
```

② 在e_2表space字段上创建空间索引。

```
ALTER TABLE e_2 ADD SPATIAL INDEX sp_space(space);
```

（5）删除索引

如果某些索引降低了数据库的性能，或者根本就没有必要创建该索引，可以考虑将索引删除。删除索引有两种方式，具体如下：

① 使用DROP INDEX删除索引：

```
DROP INDEX索引名ON表名;
```

② 使用ALETR TABLE删除索引

```
ALTER TABLE表名DROP INDEX索引名;
```

【例 】使用ALTER TABLE将stu表中的stu_sname索引删除。SQL语句如下：

```
ALTER TABLE stu DROP INDEX stu_sname;
```

## 二、视图

（1）视图的基本概念

视图是一种数据库对象，是从一个或多个基表（或视图）中导出的虚表。

视图的创建语句中，SELECT子句引用的数据表称为视图的基表，视图可以看作虚拟表或虚表。

通过视图访问的数据不作为独立的对象存储在数据库内。视图被定义后便存储在数据库中，通过视图看到的数据只是存放在基表中的数据。

当对通过视图看到的数据进行修改时，相应的基表的数据也会发生变化，同时，若基表的数据发生变化，这种变化也会自动反映到视图中。视图可以是一个基表数据的一部分，也可以是多个基表数据的联合；视图也可以由一个或多个其他视图产生。

视图通常用来进行以下三种操作：

• 筛选表中的记录。

• 防止未经许可的用户访问敏感数据。

• 将多个物理数据表抽象为一个逻辑数据表。

视图上的操作和基表类似，但是数据库管理系统对视图的更新操作（INSERT、DELETE、UPDATE）往往存在一定的限制。数据库管理系统对视图进行的权限管理和基表也有所不同。

视图可以增强数据的逻辑独立性和安全性。

（2）视图的优点

视图只是保存在数据库中的SELECT查询。因此，对查询执行的大多数操作也可以在视图上进行。也就是说视图只是给查询起了一个名字，把它作为对象保存在数据库中。只要使用简单的SELECT语句即可查看视图中查询的执行结果。视图是定义在基表（视图的数据源）之上的，对视图的一切操作最终会转换为对基表的操作。

为什么要引入视图呢？这是由于视图具有如下优点。

① 视图能够简化用户的操作。视图使用户可以将注意力集中在自己关心的数据上，如果这些数据不是直接来自于基表，则可以通过定义视图，使用户眼中的数据结构简单、清晰，并且可以简化用户的数据查询操作。

例如，那些来源于若干张表连接查询的视图，就将表与表之间的连接操作对用户隐藏了起来。换句话说，用户所做的只是对一个虚表的简单查询，而这个虚表是怎样得到的，用户无需了解。

② 视图使用户能从多种角度看待同一数据。视图机制使不同的用户能以不同的方式看待同一数据，当不同用户使用同一个数据库时，这种灵活性是非常重要的。

③ 视图使重构数据库具备逻辑独立性。数据的逻辑独立性是指当数据库重构时，如增加新的表或对原表增加新的字段时，用户和用户程序不受影响。

④ 视图能够对机密数据提供安全保护。有了视图机制，就可以在设计数据库应用系统时，对不同的用户定义不同的视图，使机密数据不出现在不应看到这些数据的用户视图上。

（3）定义视图

在MySQL中，使用CREATE VIEW语句创建视图。语法格式如下：

```
CREATE [OR REPLACE] [ALGORITHM={UNDEFINED|MERGE|TEMPTABLE}]
VIEW视图名[(字段名列表)]
AS
  SELECT语句
[ WITH [CASCADED|LOCAL] CHECK OPTION ]
```

在可更新视图中加入WITH CHECK OPTION子句，当该视图是根据另一个视图定义的时候， LOCAL和CASCADED关键字将决定检查测试的范围。LOCAL关键字对CHECK OPTION进行了限制，使其仅作用在定义的视图上，CASCADED关键字则会对与该视图相关的所有视图和基表进行检查。如果未给定任一关键字，默认值为CASCADED。

创建视图时要求创建者具有针对视图的CREATE VIEW权限，以及针对SELECT语句选择的每一列上的某些权限。对于在SELECT语句中其他地方使用的列，创建者必须具有SELECT权限。如果还有OR REPLACE子句，创建者必须在视图上具有DROP权限。

视图属于数据库。默认情况下，将在当前数据库上创建新视图。如果想在指定数据库中创建视图，则需要将视图名称指定为“数据库名.视图名”。

【例】在“学生选课”数据库中创建一个基于teacher表的teacher_view视图，要求查询并输出所有教师的tname（姓名）字段、tgender（性别）字段、tpro（职称）字段。先把当前数据库设为“学生选课”数据库，执行以下SQL语句：

```
CREATE VIEW teacher_view
AS
SELECT tname,tgender,tpro FROM teacher;
```

执行上述语句，在“学生选课”数据库中创建teacher_view视图。使用SELECT语句查询teacher_view视图，结果如图：

```
mysql> SELECT * FROM teacher_view;
+-----------+---------+--------------+
| tname     | tgender | tpro         |
+-----------+---------+--------------+
| 吴亚飞    | 男      | 讲师         |
| 李琦      | 男      | 副教授       |
| 马艳红    | 女      | 讲师         |
| 万丽      | 女      | 助理讲师     |
+-----------+---------+--------------+
4 rows in set (0.00 sec)
```

【例】在“学生选课”数据库中创建一个基于teacher表的teacher1_view视图，要求查询并输出所有教师的tname（姓名）字段、tgender（性别）字段、tpro（职称）字段，并将视图中的字段名设为教师姓名、教师性别和教师职称。先把当前数据库设为“学生选课”数据库，执行以下SQL语句：

```
CREATE VIEW teacher1_view(教师姓名,教师性别,教师职称)
AS
SELECT tname,tgender,tpro FROM teacher;
```

执行上述语句，在“学生选课”数据库中创建teacher1_view视图。使用SELECT语句查询teacher1_view视图，结果如图：

```
mysql> SELECT * FROM teacher1_view;
+--------------+--------------+--------------+
| 教师姓名     | 教师性别     | 教师职称     |
+--------------+--------------+--------------+
| 吴亚飞       | 男           | 讲师         |
| 李琦         | 男           | 副教授       |
| 马艳红       | 女           | 讲师         |
| 万丽         | 女           | 助理讲师     |
+--------------+--------------+--------------+
4 rows in set (0.00 sec)
```

【例】在“学生选课”数据库中创建一个基于学生表studentInfo、课程表course和选课表elective的nopass_view视图，要求查询并输出所有不及格学生的sno（学号）字段，sname（姓名）字段，cname（课程名）字段，score（成绩）字段。先把当前数据库设为“学生选课”数据库，执行以下SQL语句：

```
    CREATE VIEW nopass_view
    AS
       SELECT studentInfo.sno AS学号,sname AS姓名,cname AS课程名,
       score AS成绩FROM studentInfo a INNER JOIN elective b ON a.sno
=b.sno
       INNER JOIN course c ON b.cno=c.cno
       WHERE score<60;
```

使用SELECT语句查询nopass_view视图，结果如图：

```
mysql> SELECT * FROM nopass_view;
+----------+----------+-------------+--------+
| 学号     | 姓名     | 课程名      | 成绩   |
+----------+----------+-------------+--------+
| 10102001 | 王斌     | 文学欣赏    |     50 |
| 11101001 | 刘淑芳   | 文学欣赏    |     49 |
| 10101001 | 张永峰   | 音乐欣赏    |     51 |
+----------+----------+-------------+--------+
3 rows in set (0.02 sec)
```

（4）查看视图

查看视图，是指查看数据库中已经存在的视图的定义。查看视图必须有SHOWVIEW权限。查看视图的方式有三种。

1. 使用DESCRIBE语句查看视图在MySQL中，使用DESCRIBE语句可以查看视图的字段信息，包括字段名、字段类型等。DESCRIBE语句的语法格式如下所示：

```
DESCRIBE视图名;
```

2. 使用SHOW TABLE STATUS语句查看视图：

```
SHOW TABLE STATUS LIKE '视图名';
```

对比发现：在表的显示信息中，Engine（存储引擎）、Date_length（数据长度）、Index_length（索引长度）等项都有具体的值，但是Comment项没有信息，说明这是表而不是视图，这也是视图和表最直接的区别。

3. 使用SHOW CREATE VIEW语句查看视图在MySQL中，使用SHOW CREATE VIEW语句不仅可以查看创建视图的定义语句，还可以查看视图的字符编码以及视图中记录的行数。

SHOW CREATE VIEW语句的语法格式如下：

```
SHOW CREATE VIEW  视图名;
```

（5）修改和删除视图

1. 修改视图修改视图就是修改数据库中已经存在的视图的定义。

在MySQL中，修改视图的方式有两种。

① 使用CREATE或REPLACE VIEW语句

语法格式为：

```
CREATE或REPLACE [ALGORITHM={UNDEFINED|MERGE|TEMPTABLE}]
VIEW视图名[(字段名列表)]
AS
 select语句
[ WITH [CASCADED|LOCAL] CHECK OPTION ]
```

使用CREATE或REPLACE VIEW语句创建视图时，如果视图已经存在，则用语句中的视图定义修改已存在的视图。如果视图不存在，则创建一个视图。

② 使用ALTER VIEW语句

语法格式为：

```
ALTER [ALGORITHM={UNDEFINED|MERGE|TEMPTABLE}]
VIEW视图名[(字段名列表)]
AS
   SELECT语句
[ WITH [CASCADED|LOCAL] CHECK OPTION ]
```

【例】使用ALTER VIEW语句，修改创建的teacher_view视图，查询输出所有职称为“讲师”的教师的tname字段、tpro字段。

SQL语句如下：

```
ALTER VIEW teacher_view
AS
  SELECT tname AS姓名,tpro AS职称FROM teacher WHERE tpro='讲师';
```

使用SELECT语句查询teacher_view视图，结果如图：

```
mysql> SELECT * FROM teacher_view;
+-----------+--------+
| 姓名      | 职称   |
+-----------+--------+
| 吴亚飞    | 讲师   |
| 马艳红    | 讲师   |
+-----------+--------+
2 rows in set (0.00 sec)
```

2. 删除视图当不再需要视图时，可以将视图删除。删除视图只是将视图的定义删除，并不会影响基表中的数据。删除视图的语法格式为：

```
DROP VIEW [IF EXISTS]视图名1[,视图名2]…;
```

在上述语法格式中，视图名可以有一个或多个，即同时删除一个或多个视图。视图名之间用逗号分隔。删除视图必须有DROP VIEW权限。IF EXISTS可选项表示删除视图时如果存在指定视图，则将指定视图删除，如果不存在指定视图，删除操作也不会出现错误。

（6）更新视图

更新视图是指通过视图来插入、删除和更新基表中的数据。因为视图是一个虚拟表，其中并没有数据，修改视图中的数据实际上是在修改基表中的数据。只要满足一些限制条件，就可以通过视图自由地插入、删除和更新数据。

1. 使用INSERT语句向视图中插入数据使用视图插入数据与向基表中插入数据一样，都可以通过INSERT语句来实现。插入数据的操作是针对视图中字段的插入操作，而不是针对基表中所有字段的插入操作。使用视图插入数据要满足一定的限制条件。

• 使用INSERT语句进行插入操作的用户必须有在基表中插入数据的权限，否则插入操作会失败。

• 如果视图上没有包含基表中所有属性为NOT NULL的字段，那么插入操作会由于那些字段存在NULL值而失败。

• 如果视图中的数据是由聚合函数或者表达式计算得到的，则插入操作不成功。

• 不能在使用了DISTINCT、UNION、TOP、GROUP BY或HAVING子句的视图中插入数据。

• 如果在创建视图的CREATE VIEW语句中使用了WITH CHECK OPTION子句，那么所有对视图进行修改的语句必须符合WITH CHECK OPTION中的限定条件。

• 对于由多个基表连接查询而生成的视图来说，一次插入操作只能作用于一个基表上。

【例】在“学生选课”数据库中，基于studentInfo表创建一个名为student_view的视图。该视图包含所有学生的sno字段、sname字段、sgender字段的数据。SQL语句如下：

```
USE学生选课;
CREATE VIEW student_view
AS
  SELECT  sno,sname,sgender
  FROM studentInfo;
```

接下来向student_view视图中插入一条数据，sno（学号）字段的值为11101004，sname（姓名）字段的值为“张三”、sgender（性别）字段的值为“女”。实现上述操作，可以使用INSERT语句：

```
INSERT INTO student_view
VALUES('11101004', '张三','女');
```

成功执行上述语句后，使用SELECT语句查看该视图和学生表studentInfo中的数据！

2. 使用UPDATE语句更新视图中数据在视图中更新数据与在基表中更新数据一样，都需要使用UPDATE语句。当视图中的数据来源于多个基表时，与插入操作一样，每次更新操作只能更新一个基表中的数据。通过视图修改存在于多个基表中的数据时，要分别对不同的基表使用UPDATE语句。在视图中使用UPDATE语句进行更新操作时也受到与进行插入操作时一样的限制。

【例】将前面的视图student_view中sname（姓名）字段值为“张三”的学生记录的sgender（性别）字段值更新为“男”。

SQL语句如下：

```
mysql> SELECT * FROM student_view;
+----------+----------+---------+
| sno      | sname    | sgender |
+----------+----------+---------+
| 10101001 | 张永峰   | 男      |
| 10101002 | 何小丽   | 女      |
| 10101003 | 张宇     | 男      |
| 10102001 | 王斌     | 女      |
| 10102002 | 包玉明   | 男      |
| 10102003 | 孙平平   | 女      |
| 10102004 | 翁静静   | 女      |
| 11101001 | 刘淑芳   | 女      |
| 11101002 | 王亚旭   | 男      |
| 11101003 | 高磊     | 男      |
| 11101004 | 张三     | 男      |
+----------+----------+---------+
11 rows in set (0.00 sec)
```

成功执行上述语句后，查看视图中的数据，结果如图：

```
mysql> SELECT * FROM studentInfo;
+----------+----------+---------+---------------------+-----------------+
| sno      | sname    | sgender | sbirth              | sclass          |
+----------+----------+---------+---------------------+-----------------+
| 10101001 | 张永峰   | 男      | 1993-08-01 00:00:00 | 电子商务101     |
| 10101002 | 何小丽   | 女      | 1992-11-03 00:00:00 | 电子商务101     |
| 10101003 | 张宇     | 男      | 1992-08-21 00:00:00 | 电子商务101     |
| 10102001 | 王斌     | 女      | 1991-07-14 00:00:00 | 网络技术101     |
| 10102002 | 包玉明   | 男      | 1993-11-15 00:00:00 | 网络技术101     |
| 10102003 | 孙平平   | 女      | 1992-02-27 00:00:00 | 网络技术101     |
| 10102004 | 翁静静   | 女      | 1992-05-09 00:00:00 | 网络技术101     |
| 11101001 | 刘淑芳   | 女      | 1994-06-10 00:00:00 | 电子商务111     |
| 11101002 | 王亚旭   | 男      | 1993-03-18 00:00:00 | 电子商务111     |
| 11101003 | 高磊     | 男      | 1993-05-11 00:00:00 | 电子商务111     |
| 11101004 | 张三     | 男      | NULL                | NULL            |
+----------+----------+---------+---------------------+-----------------+
11 rows in set (0.00 sec)
```

3. 使用DELETE语句删除数据通过视图删除数据与在基表中删除数据的方式一样，都需要使用DELETE语句。在视图中删除的数据，同时也会从基表中删除。当一个视图连接了两个以上的基表时，对该视图中数据的删除操作是不允许的。

【例】删除视图student_view中sname字段值为“张三”的数据。

SQL语句如下：

```
USE学生选课;
DELETE FROM student_view
WHERE sname='张三';
```

然后用select语句查看该视图和学生表studentinfo中的数据！

三、触发器

（1）触发器概述

触发器是MySQL 5.0新增的功能，是一种与表操作（INSERT、UPDATE、DELETE）有关的数据库对象。触发器定义了一系列操作，这一系列操作称为触发程序。当触发器所在表上出现INSERT、UPDATE、DETETE操作时，将激活触发器。触发器基于一个表创建，但是可以针对多个表进行操作，因此触发器可以用来对表实施复杂的完整性约束。

触发器具有以下优点：

① 触发器可以自动执行。当对表进行INSERT、UPDATE、DELETE操作，试图修改表中的数据时，相应操作的触发器立即自动执行。

② 触发器可以对数据库中相关表进行层叠更改。这比直接把代码写在前端的做法更安全合理。

③ 触发器可以实现表的约束实现不了的复杂约束。在触发器中可以引用其他表中的字段，从而实现多表之间的复杂约束。

④ 触发器可以维护冗余数据，实现外键级联等。

（2）创建触发器

创建触发器用CREATE TRIGGER语句。CREATE TRIGGER语句的语法格式如下：

```
CREATE TRIGGER触发器名
AFTER/BEFORE INSERT/UPDATE/DELETE ON表名
FOR EACH ROW
    BEGIN
      SQL语句:#触发程序
    END;
```

① 触发器是数据库对象，因此创建触发器时，需要指定该触发器属于哪一个数据库。

② 触发器是在表上创建的。这个表必须是基表，不能是临时表，也不能是视图。

③ MySQL触发器的触发事件有三种：INSERT、UPDATE、DELETE。

• INSERT：将新记录插入表时激活触发程序。

• UPDATE：更改表中的记录时激活触发程序。

• DELETE：从表中删除记录时激活触发程序。

④ 触发器的触发时间有两种：BEFORE和AFTER。BEFORE表示在触发事件发生之前执行触发程序，AFTER表示在触发事件发生之后执行触发程序。

⑤ FOR EACH ROW表示行级触发器。目前，MySQL仅支持行级触发器，不支持语句级别的触发器。FOR EACH ROW表示INSERT、UPDATE、DELETE操作影响的每一条记录都会执行一次触发程序。

⑥ 触发程序中的SELECT语句不能产生结果集。

⑦ 触发程序中可以使用OLD关键字与NEW关键字。

• 向表中插入新记录时，在触发程序中可以使用NEW关键字表示新记录。当需要访问新记录中的某个字段时，可以使用“NEW.字段名”进行访问。

• 从表中删除某条旧记录时，在触发程序中可以使用OLD关键字表示删除的旧记录。当需要访问删除的旧记录中的某个字段时，可以使用“OLD.字段名”进行访问。

• 修改表中的某条记录时，在触发程序中可以使用NEW关键字表示修改后的记录，使用OLD关键字表示修改前的记录。当需要访问修改后的记录中的某个字段时，可以使用“NEW.字段名”进行访问。当需要访问修改前的记录中的某个字段时，可以使用“OLD.字段名”进行访问。

• OLD记录是只读的，在触发程序中只能引用它，不能更改它。在BEFORE触发程序中，可使用“SET NEW.字段名=值”更改NEW记录的值。但在AFTER触发程序中，不能使用“SET NEW.字段名=值”更改NEW记录的值。

• 对于INSERT操作，只有NEW关键字是合法的。对于DELETE操作，只有OLD关键字是合法的。对于UPDATE操作，NEW关键字和OLD关键字都是合法的。

（3）触发器的使用

1. 使用触发器实现检查约束在MySQL中，可以使用复合数据类型SET或ENUM对字段的取值范围进行检查约束，也可以实现对离散的字符串类型数据的检查约束。对于数值类型的字段，不建议使用SET或者ENUM数据类型实现检查约束，可以使用触发器实现。

【例】使用触发器实现检查约束，在向elective表插入记录时，score字段的值或者为空，或者取值0～100。如果score字段的值不满足要求，小于0则填入0，大于100则填入100。对应的SQL语句如下：

```
    USE学生选课;
    DELIMITER  //
    CREATE TRIGGER tr_elective_insert BEFORE INSERT ON elective FOR
EACH ROW
      BEGIN
        IF(NEW.score IS NOT NULL&&NEW.score<0)
        THEN
          SET NEW.score=0;
        ELSEIF(NEW.score IS NOT NULL&&NEW.score>100)
             THEN
               SET NEW.score=100;
        END IF;
      END //
    DELIMITER ;
```

这个触发器的触发时间是BEFORE，即将记录插入到表之前，先执行触发程序。在触发程序中判断新插入的记录的score字段的值是否小于0或者大于100，若是，将score字段的值改为0或100，再插入到表中。下面用一条INSERT语句测试tr_elective_insert触发器，执行结果如图：

```
mysql> INSERT INTO elective VALUES('10101001','C002',200);
Query OK, 1 row affected (0.04 sec)

mysql> SELECT * FROM elective;
+----------+------+-------+
| sno      | cno  | score |
+----------+------+-------+
| 10101001 | c001 |    73 |
| 10101001 | C002 |   100 |
| 10101001 | c003 |    81 |
| 10101001 | c004 |    51 |
| 10101002 | c001 |    78 |
| 10101003 | c003 |    69 |
| 10102001 | c001 |    50 |
```

下面用一条UPDATE语句测试tr_elective_UPDATE触发器，执行结果如图：

```
mysql> UPDATE elective SET score=200 WHERE sno='10101001' AND cno='C002';
Query OK, 0 rows affected (0.05 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> SELECT * FROM elective;
+----------+------+-------+
| sno      | cno  | score |
+----------+------+-------+
| 10101001 | c001 |    73 |
| 10101001 | C002 |   100 |
| 10101001 | c003 |    81 |
| 10101001 | c004 |    51 |
| 10101002 | c001 |    78 |
```

这个触发器的触发时间是BEFORE，在记录的更新操作执行前，先执行触发程序。再在触发程序中判断更新的score字段的值是不是满足要求，如果不满足，则将要更新的score字段的值修改为原来score字段的值，然后再对记录进行更新操作。

2. 使用触发器维护冗余数据冗余的数据需要额外的维护。维护冗余数据时，为了避免数据不一致问题的发生，最好交由系统（例如触发器）自动维护。

【例】使用触发器实现：当一位教师退休或调离时，将该教师的信息放入old_teacher表中。old_teacher表的结构如图：

```
mysql> DESC old_teacher;
+---------+-------------+------+-----+---------+----------------+
| Field   | Type        | Null | Key | Default | Extra          |
+---------+-------------+------+-----+---------+----------------+
| no      | int(11)     | NO   | PRI | NULL    | auto_increment |
| tname   | varchar(8)  | NO   |     | NULL    |                |
| tgender | char(1)     | YES  |     | NULL    |                |
| tedu    | varchar(10) | YES  |     | NULL    |                |
| tpro    | varchar(10) | YES  |     | NULL    |                |
+---------+-------------+------+-----+---------+----------------+
5 rows in set (0.01 sec)
```

对应的SQL语句：

```
USE学生选课;
DELIMITER  //
CREATE TRIGGER tr_teacher_delete AFTER DELETE on teacher FOR EAC
H ROW
  BEGIN
   INSERT INTO old_teacher(tname,tgender,tedu,tpro)
   VALUES(OLD.tname,  OLD.tgender,OLD.tedu,OLD.tpro);
  END //
DELIMITER ;
```

下面用一条DELETE语句测试tr_teacher_delete触发器。先用SELECT语句查询teacher表的数据，查询结果如图7-28所示。将teacher表中的工号为“t006”的记录删除，删除记录后teacher表和old_teacher表中的数据如图：

```
mysql> SELECT * FROM teacher;
+------+----------+---------+----------------+-------------+
| tno  | tname    | tgender | tedu           | tpro        |
+------+----------+---------+----------------+-------------+
| t001 | 吴亚飞   | 男      | 本科           | 讲师        |
| t002 | 李琦     | 男      | 硕士研究生     | 副教授      |
| t003 | 王艳红   | 女      | 硕士研究生     | 讲师        |
| t004 | 马志超   | 男      | 博士研究生     | 教授        |
| t005 | 万丽     | 女      | 硕士研究生     | 助理讲师    |
| t006 | 和平     | 男      | 本科           | 教授        |
+------+----------+---------+----------------+-------------+
6 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM teacher;
+------+----------+---------+----------------+-------------+
| tno  | tname    | tgender | tedu           | tpro        |
+------+----------+---------+----------------+-------------+
| t001 | 吴亚飞   | 男      | 本科           | 讲师        |
| t002 | 李琦     | 男      | 硕士研究生     | 副教授      |
| t003 | 王艳红   | 女      | 硕士研究生     | 讲师        |
| t004 | 马志超   | 男      | 博士研究生     | 教授        |
| t005 | 万丽     | 女      | 硕士研究生     | 助理讲师    |
+------+----------+---------+----------------+-------------+
5 rows in set (0.00 sec)

mysql> SELECT * FROM old_teacher;
+----+--------+---------+-------+--------+
| no | tname  | tgender | tedu  | tpro   |
+----+--------+---------+-------+--------+
|  1 | 和平   | 男      | 本科  | 教授   |
+----+--------+---------+-------+--------+
```

3. 使用触发器实现外键级联选项对于使用InnoDB存储引擎的表而言，可以通过设置外键级联选项CASCADE、SETNULL或者NO ACTION（RESTRICT），将外键约束关系交由InnoDB存储引擎自动维护。

外键级联选项CASCADE、SET NULL或者NO ACTION（RESTRICT）的含义如下：① CASCADE：从父表中删除或更新对应的行时，同时自动删除或更新子表中匹配的行。ON DELETE CASCADE和ON UPDATE CASCADE都被InnoDB存储引擎所支持。② SET NULL：从父表中删除或更新对应的行时，同时将子表中的外键列设为空。注意，在外键列没有被约束为NOT NULL时才有效。ON DELETE SET NULL和ON UPDATE SET NULL都被InnoDB存储引擎所支持。

③ NO ACTION：InnoDB存储引擎拒绝删除或者更新父表。

④ RESTRICT：拒绝删除或者更新父表。指定RESTRICT（或者NO ACTION）和忽略ON DELETE或者ON UPDATE选项的效果是一样的。

对于使用InnoDB存储引擎的表之间存在外键约束关系但是没有设置级联选项或者使用的数据库表为MyISAM（MyISAM表不支持外键约束关系）时，可以使用触发器来实现外键约束之间的级联选项。

【例】创建“后勤管理”数据库，数据库中有学生表stu和宿舍表dorm，学生表stu中有学号sno、姓名sname、性别sgender、班级sclass等字段，主键为学号sno字段，宿舍表dorm中有宿舍号dno、床位号bno、学号sno等字段，宿舍号dno和床位号bno字段联合起来做主键，分别用触发器实现宿舍表dorm的学号sno字段和学生表stu的学号sno字段之间的外键级联选项CASCADE、SET NULL。

① 创建“后勤管理”数据库。

② 创建学生表stu和宿舍表dorm。

③ 在学生表stu和宿舍表dorm中插入记录。

④ 创建触发器tr_1，实现宿舍表dorm中学号sno字段和学生表stu中学号sno字段的外键CASCADE级联选项，在学生表stu中删除某个学生记录时，宿舍表dorm中对应学生的住宿记录也被删除。

⑤ 创建触发器tr_2，实现宿舍表dorm中学号sno字段和学生表stu中学号sno字段的外键SET NULL级联选项，在学生表中修改某个学生的学号时，宿舍表中对应学生的学号sno字段值设为NULL。

⑥ 删除学生表stu中学号sno字段值为10101001的记录，再将学生表stu中学号sno字段值从10101002修改为10101003，查看触发器tr_1和tr_2的执行结果。

（4）查看触发器的定义

① 使用SHOW TRIGGERS命令查看触发器的定义。使用“SHOW TRIGGERS\G”命令可以查看当前数据库中所有触发器的信息。使用“SHOW TRIGGER LIKE模式\G”命令可以查看与模式模糊匹配的触发器的信息。

【例】查看在前面的stu表上创建的触发器的信息。

对应的SQL语句如下：

```
SHOW TRIGGERS LIKE'stu%'\G
```

当使用一个含有“SHOW TRIGGERS”的LIKE子句时，待匹配的表达式会与触发器定义时所在的表名称相比较，而不与触发器的名称相比较。

② 使用SHOW CREATE TRIGGER命令查看触发器的定义。使用“SHOW CREATE TRIGGER触发器名”命令可以查看指定名称的触发器的定义。

③ 通过查询information_schema数据库中的triggers表，可以查看触发器的定义。MySQL中所有触发器的定义都存放在information_schema数据库的triggers表中，查询triggers表时，可以查看所有数据库中所有触发器的详细信息，查询语句如下：

```
SELECT * FROM information_schema.triggers\G
```

⑤ 删除触发器如果不再使用某个触发器，可以使用DROP TRIGGER语句将其删除。

DROP TRIGGER语句语法如下：

```
DROP TRIGGER触发器名;
```

# 本章小结

• 索引是一种特殊类型的数据对象，它可以用来提高对表中数据的访问速度，而且还能够对表实施完整性约束。

• MySQL中的索引类型包括普通索引、唯一索引、全文索引和空间索引。其中，唯一索引要求任意两行的被索引字段不能存在重复值。索引可以在表中单个字段上创建，也可以在表中多个字段上创建。在多个字段上创建索引时，只有在查询条件中使用了这些字段中的第一个字段时，该索引才会被引用。

• 视图是从一个或多个表中导出来的表，是一种虚拟表。视图的结构和数据依赖于基本表。

• 视图可以简化查询语句，提高数据库的安全性。视图还可以修改基本表中的数据。

• 触发器是一种特殊类型的存储过程，在某个指定的事件发生时被激活。

• 触发器的两种类型：AFTER触发器和BEFORE触发器。AFTER触发器是先执行触发事件，再执行触发程序。BEFORE触发器是先执行触发程序，再执行触发事件。

• INSERT触发器在对触发器表执行插入记录操作时被触发。UPDATE触发器在对触发器表执行更新记录操作时被触发。DELETE触发器在对触发器表执行删除记录操作时被触发。

• 在触发程序中可以使用NEW关键字和OLD关键字。NEW关键字表示新插入的记录或更新后的记录。OLD关键字表示删除的记录或更新前的记录。

• 创建触发器使用CREATE TRIGGER命令，删除触发器使用DELETE TRIGGER命令。如果要修改触发器，可以先删除，再创建。

实训项目：
项目1：在“网上书店”数据库中创建索引并查看维护
目的：掌握索引的创建、维护、使用。
内容：
（1）在会员表的联系方式列上定义唯一索引。
（2）在图书表的图书名称列上定义普通索引。
（3）在订购表的图书编号和订购日期列上创建多列索引。
（4）删除以上所建索引。
项目2：在“网上书店”数据库中创建视图并维护使用
目的：掌握视图的定义、维护、使用。
内容：
（1）定义基于图书表的视图（包含图书编号、图书名称、作者、价格、出版社、图书类别）。
（2）查询图书表视图，输出图书的名称和价格，并把查询结果按价格降序排列。
（3）查询图书表视图，输出价格最高的三种图书的名称和价格。

课后习题

一. 选择题

1. （　）的功能是视图可以实现的。

A. 将用户限定在表中的特定行上

B. 将用户限定在特定列上

C. 将多个表中的列连接起来

D. 将多个数据库的视图连接起来

2. 下列哪个选项是在使用视图修改数据时需要注意的？（　）

A. 在一个UPDATE语句中修改的字段必须属于同一个基本表

B. 一次可以修改多个视图的基本表

C. 视图中所有字段的修改必须遵守基本表所定义的各种数据完整性约束

D. 可以对视图中的计算字段进行修改

3. 下列关于视图的说法哪个是错误的？（ ）

A. 视图可以集中数据，简化和定制不同用户对数据集的不同要求

B. 视图可以使用户只关心他感兴趣的某些特定数据和他所负责的特定任务

C. 视图可以让不同的用户以不同的方式看到不同或者相同的数据集

D. 视图不能用于连接多表

4. 下列几项中，关于视图叙述正确的是：（ ）

A. 视图是一张虚表，所有的视图中不含有数据

B. 不允许用户使用视图修改表中的数据

C. 视图只能访问所属数据库的表，不能访问其他数据库的表

D. 视图既可以通过表得到，也可以通过其他视图得到

5. （ ）是索引的类型。

A. 唯一索引　B. 普通索引　C. 多列索引　D. 全文索引

6. 一张表中至多可以有（　）个普通索引。

A. I　　B. 249　　C. 3　　D. 无限多

7. （　）语句用来创建一个触发器。

A. CREATE PROCEDURE　　B. CREATE TRIGGER　　C. DROP PROCEDURE　　D. DROP TRIGGER

8. 触发器创建在（　）上。

A. 表　　B. 视图　　C. 数据库　　D. 查询

9. 当删除（　）时，与它关联的触发器也同时删除。

A. 视图　　B. 临时表　　C. 过程　　D. 表

二. 问答题

1. 简单说明视图的基本概念及优点。

2. 举例说明简单SELECT查询和视图的区别与联系。

3. 举例说明索引的概念与作用。

4. 举例说明什么是全文索引并写出创建全文索引的SQL语句。

5. 什么是触发器？它与存储过程有什么区别与联系？

6. 使用触发器有什么优点？

# THANKS

## 感谢观看

微信公众号：灰灰考研