**两张关联的表,将两张表的关联的字段加上索引，能大大提高查询的速度，另外索引给数据量少的那张表的关联字段加上为好，另外查询经常要用的字段也可以加上索引，尽量用EXPLAIN来查看sql查询的情况，看那些查询中可以改成可以运用到索引的sql**

**Mysql整型的取值范围:**



**Float，double:浮点型**

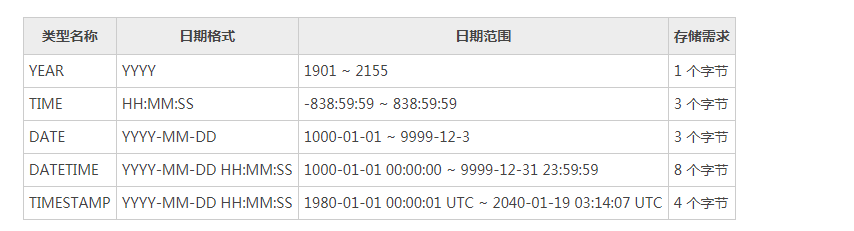
**Decimal：定点数**

浮点类型和定点类型都可以用(M, D)来表示，其中M称为精度，表示总共的位数；D称为标度，表示小数的位数。  
浮点数类型的取值范围为 M（1～255）和 D（1～30，且不能大于 M-2），分别表示显示宽度和小数位数。M 和 D 在 FLOAT 和DOUBLE 中是可选的，FLOAT 和 DOUBLE 类型将被保存为硬件所支持的最大精度。DECIMAL 的默认 D 值为 0、M 值为 10。

浮点数相对于定点数的优点是在长度一定的情况下，浮点数能够表示更大的范围；缺点是会引起精度问题。

不论是定点还是浮点类型，如果用户指定的精度超出精度范围，则会四舍五入进行处理。

**时间类型：**



**Year:存储年份**

**TIME：存储时间信息的值 格式：HH：MM：SS**

**DATE类型：存储日期：YY—MM-DD**

**DATETIME:包含日期和时间：YYYY—MM-DD HH：MM：SS**

## TIMESTAMP :同上，但是取值范围小，优先选这个。

**Mysql 修改数据表相关语句：**



**增加字段：alter table 表名 add column 列名 int first/after ：用在制定新增字段在表中的位置**

**删除字段:alter table xxx drop xx**

**修改名称用change 修改名称modify**

**删除表：drop table xx**

**主键：**

在修改表时添加主键约束

Alter table 数据表 add primary key(列明)

**约束：**

**查看表的所有信息，例如约束什么的。**

show create table test3

**主键与外键的区分：**

外键是表的一个字段，不是本表的主键，但对应另一个表的主笔爱哦，定义外键后，不允许删除另一个表中具有关联关系的行

主键中的类型与外键表的字段的类型一定要一致

外键的主要作用是保持数据的一致性、完整性。例如，部门表 tb\_dept 的主键是 id，在员工表 tb\_emp5 中有一个键 deptId 与这个 id 关联。

主表（父表）：对于两个具有关联关系的表而言，相关联字段中主键所在的表就是主表。

从表（子表）：对于两个具有关联关系的表而言，相关联字段中外键所在的表就是从表。

**在创建表示设置外键约束：**

**Constrain 外键名 foreign key 字段名 references 主表名 主键列**

**在修改时添加：**

Alter table 数据表名 add constraint 索引名 foreign key （列名） references 主表名（列名）

**删除外键约束**

Alter table 表名 drop foreign key 外键约束名

**唯一约束:**

**在定义列之后直接使用unique**

Name varchar(22) unique

**在修改表时添加唯一约束**

Alter table 数据表 add constraint 约束名 unique（列名）

**删除唯一约束**

Alter table test表 drop index 约束名

**检查约束：**

mysql不支持检查约束 简单的可以使用枚举

**limit：**接受一个或者两个数字参数，参数必须是一个整数常量，如果给定两个参数，第一个参数返回记录行的偏移量，第二个参数返回记录行的最大数目

偏移量指的是从哪一行开始显示，第二个参数则表示显示多少行

**比较运算符：**



**Least：least(value1,value2)**

当参数有两个或者更多时，返回值最小的那个。

**Greatest: Greatest (value1,value2)**

与上面Least相反，该函数返回最大一个值

**内连接**是通过在查询中设置链接条件的暗示，来移除查询结果集中的某些数据行后的交叉连接，简单来说，就是利用条件表达式来消除交叉连接的某些数据航

**外连接：**先将连接的表分为基表和参考表，再以基表为依据返回满足和不满足的记录

**外连接分为：**左外链接，和右外连接

在左外连接的结果集中，除了匹配的行之外，还包括坐标中有但在右边中不匹配的行，对于这样的行，从由表中选中的列的值被设置成NULL，即左外连接的结果集中的NULL值表示由表中没有找到与左表相符的记录

**子查询：**

**常用操作：ANY,all,in,Exists**

**Exits**:用于检查子查询是否至少会返回一行数据，该子查询实际上并不返回任何数据，而是返回值true或者false，

**Exists**指定一个子查询，检测行的存在

**分组：**

对于 GROUP BY 子句的使用，需要注意以下几点。

GROUP BY 子句可以包含任意数目的列，使其可以对分组进行嵌套，为数据分组提供更加细致的控制

GROUP BY 子句列出的每个列都必须是检索列或有效的表达式，但不能是聚合函数。若在 SELECT 语句中使用表达式，则必须在 GROUP BY 子句中指定相同的表达式。

除聚合函数之外，SELECT 语句中的每个列都必须在 GROUP BY 子句中给出。

若用于分组的列中包含有 NULL 值，则 NULL 将作为一个单独的分组返回；若该列中存在多个 NULL 值，则将这些 NULL 值所在的行分为一组。

**Group\_contact:**计算那些行同属于一组，将属于同一组的列显示出来，要返回那些列，由函数参数（字段名）决定，分组必须有个标准，就是根据group by指定的列来进行分组

**Having：指定过滤条件（主要针对于查询出来的结果进行过滤，例如对聚合函数的别名）**

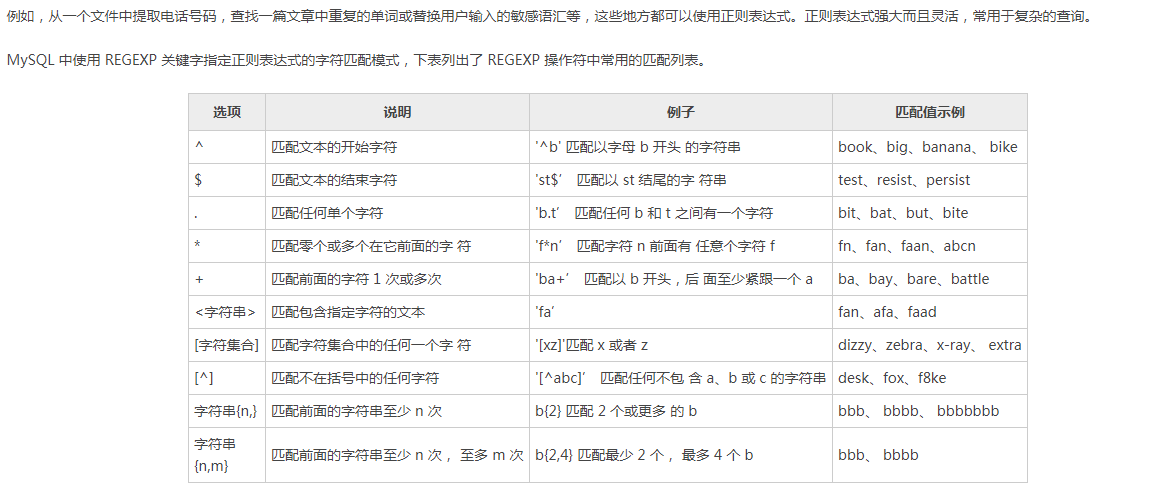
HAVING 子句和 WHERE 子句非常相似，HAVING 子句支持 WHERE 子句中所有的操作符和语法，但是两者存在几点差异：

WHERE 子句主要用于过滤数据行，而 HAVING 子句主要用于过滤分组，即 HAVING 子句基于分组的聚合值而不是特定行的值来过滤数据，主要用来过滤分组。

WHERE 子句不可以包含聚合函数，HAVING 子句中的条件可以包含聚合函数。

HAVING 子句是在数据分组后进行过滤，WHERE 子句会在数据分组前进行过滤。WHERE 子句排除的行不包含在分组中，可能会影响 HAVING 子句基于这些值过滤掉的分组。

**正则表达式：**通常被用来检索或替换符合某个模式的文本内容，根据指定的匹配模式匹配文中符合要求的特殊字符串。



**Insert into 表 （1,2,3,4，）select key1,key2, from 表2**

**视图：**

视图并不同于数据表，它们的区别在于以下几点：

视图不是数据库中真实的表，而是一张虚拟表，其结构和数据是建立在对数据中真实表的查询基础上的。

存储在数据库中的查询操作 SQL 语句定义了视图的内容，列数据和行数据来自于视图查询所引用的实际表，引用视图时动态生成这些数据。

视图没有实际的物理记录，不是以数据集的形式存储在数据库中的，它所对应的数据实际上是存储在视图所引用的真实表中的。

视图是数据的窗口，而表是内容。表是实际数据的存放单位，而视图只是以不同的显示方式展示数据，其数据来源还是实际表。

视图是查看数据表的一种方法，可以查询数据表中某些字段构成的数据，只是一些 SQL 语句的集合。从安全的角度来看，视图的数据安全性更高，使用视图的用户不接触数据表，不知道表结构。

视图的建立和删除只影响视图本身，不影响对应的基本表。

**优点：**

#### 1) 定制用户数据，聚焦特定的数据

#### 2) 简化数据操作

#### 3) 提高基表数据的安全性

#### 4) 共享所需数据

#### 5) 更改数据格式

#### 6) 重用 SQL 语句

**注意：**

* 创建视图需要足够的访问权限。
* 创建视图的数目没有限制。
* 视图可以嵌套，即从其他视图中检索数据的查询来创建视图。
* 视图不能索引，也不能有关联的触发器、默认值或规则。
* 视图可以和表一起使用。
* 视图不包含数据，所以每次使用视图时，都必须执行查询中所需的任何一个检索操作。如果用多个连接和过滤条件创建了复杂的视图或嵌套了视图，可能会发现系统运行性能下降得十分严重。因此，在部署大量视图应用时，应该进行系统测试。

**创建视图：**

Create view 视图名as select语句

**查询视图**

Describe(desc) 视图名

**修改视图：**

Alter view 视图名 as select语句

如果视图中包含以下接口中则不客更新

* 聚合函数 SUM()、MIN()、MAX()、COUNT() 等。
* DISTINCT 关键字。
* GROUP BY 子句。
* HAVING 子句。
* UNION 或 UNION ALL 运算符。
* 位于选择列表中的子查询。
* FROM 子句中的不可更新视图或包含多个表。
* WHERE 子句中的子查询，引用 FROM 子句中的表。
* ALGORITHM 选项为 TEMPTABLE（使用临时表总会使视图成为不可更新的）的时候。

**删除视图**

Drop view 视图名

**存储过程：**

优点：

1. 封装性：存储过程被创建后，可以在程序中被多次调用，而不必重新编写该存储过程的 SQL 语句，并且数据库专业人员可以随时对存储过程进行修改，而不会影响到调用它的应用程序源代码。
2. 可增强sql语句的功能和灵活性：存储过程可以用流程控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的运算
3. 可减少网络流量：由于存储过程是在服务器端运行的，且执行速度快，因此当客户计算机上调用该存储过程时，网络中传送的只是该调用语句，从而可降低网络负载。
4. 高性能：存储过程执行一次后，产生的二进制代码就驻留在缓冲区，在以后的调用中，只需要从缓冲区中执行二进制代码即可，从而提高了系统的效率和性能。
5. 提高数据库的安全性和数据的完整性：使用存储过程可以完成所有数据库操作，并且可以通过编程的方式控制数据库信息访问的权限。

**Delimiter:** 在 MySQL 中，服务器处理 SQL 语句默认是以分号作为语句结束标志的。然而，在创建存储过程时，存储过程体可能包含有多条 SQL 语句，这些 SQL 语句如果仍以分号作为语句结束符，那么 MySQL 服务器在处理时会以遇到的第一条 SQL 语句结尾处的分号作为整个程序的结束符，而不再去处理存储过程体中后面的 SQL 语句，这样显然不行。为解决这个问题，通常可使用 DELIMITER 命令将结束命令修改为其他字符。

**删除存储过程：**

Drop procedure 存储过程名

**触发器：**

*new：当触发插入和更新事件时可用，指向的是被操作的记录*

*old： 当触发删除和更新事件时可用，指向的是被操作的记录*

**创建触发器：**

Create trigger 触发器名

Delimiter ;;

Before/after on 表名

For each row

begin

sql语句

end ;;

Delimiter ;

删除触发器：

Drop trigger if exists 触发器名

**索引的优点**

1. 通过创建唯一索引，保证数据库表每行数据的唯一性
2. 大大加快数据查询速度
3. 在使用分组和排序进行数据查询时，可以显著减少查询中分组和排序的时间
4. **索引的缺点**
5. 维护索引需要耗费数据库资源
6. 索引需要占用磁盘空间，索引文件可能比数据文件更快达到最大文件尺寸
7. 当对表的数据进行增删改的时候，因为要维护索引，速度会受到影响
8. **索引的分类**
9. **普通索引和唯一索引**

主键索引是一种特殊的唯一索引，不允许有控制

1. **单列索引和复合索引**

单列索引只包含单个列

复合索引指多个字段上创建的索引，只有在查询条件上使用创建索引时的第一个字段，所以才会被使用，使用复合索引时遵循最左前缀集合

1. **全文索引**

全文索引类型为FULLTEXT，在定义索引的列上支持值得全文查找，允许在这些索引列中插入重复值和空值，全文索引可以在CHAR，VARCHAR，TEXT类型上列上创建，MYSQL只有MYISAM存储引擎支持全文索引

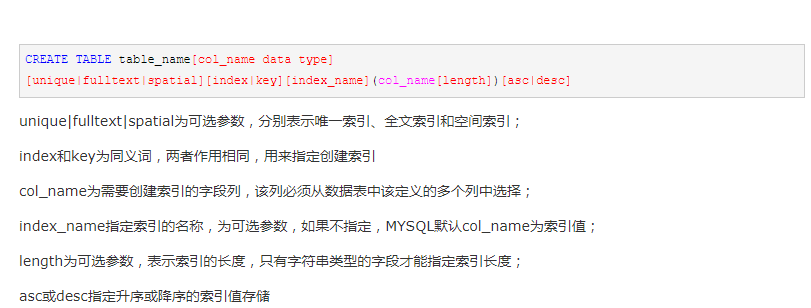
**4、空间索引**

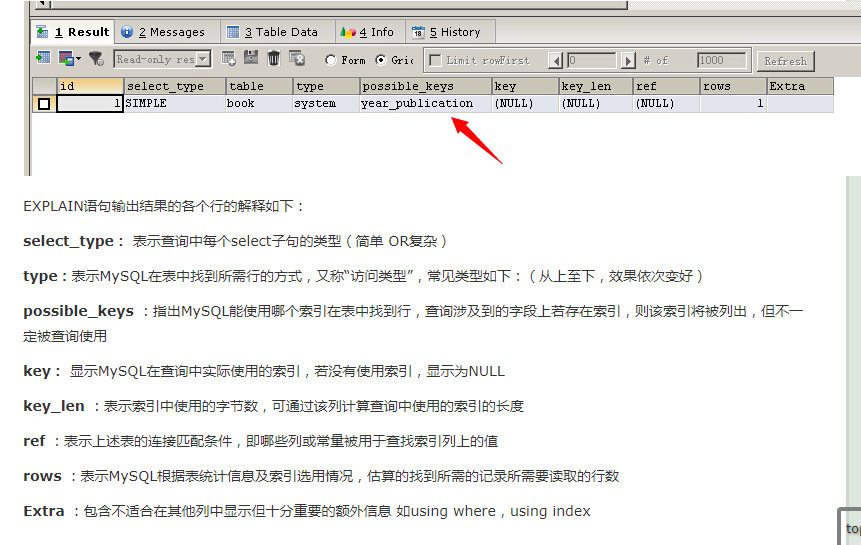
空间索引是对空间数据类型的字段建立的索引，MYSQL中的空间数据类型有4种，

分别是GEOMETRY、POINT、LINESTRING、POLYGON。

MYSQL使用SPATIAL关键字进行扩展，使得能够用于创建正规索引类型的语法创建空间索引。创建空间索引的列，必须

将其声明为NOT NULL，空间索引只能在存储引擎为MYISAM的表中创建





在已经存在的表中创建索引可以用ALTER TABLE 或者CREATE INDEX

Alter table add UNIUQE titleIds(title)



**创建表时创建索引：**

Create table xxx

(

Xxxx,

Xxxx,

Xxx,

Index(xx) –一般索引

Unique index(xxx) –唯一索引

)

**查看索引**

Show index from 表名

**删除索引**

Drop index 索引名 on 表名

**建立普通索引**

CREATE INDEX BkNameIdx ON book(bookname)

**建立唯一索引**

CREATE UNIQUE INDEX UniqidIdx ON book(bookId)

**建立复合索引**

CREATE INDEX BkAuAndInfoIdx ON book(AUTHORS(**20**),info(**50**))

**建立全文索引**

CREATE FULLTEXT INDEX infoFTIdx ON t6(info);

全文搜索：

SELECT \* FROM tablename WHERE MATCH(column1, column2) AGAINST(‘xxx′, ‘sss′, ‘ddd′)

查询索引字段1字段2，中包含xxx，sss，ddd的列

查看索引：

show index from 007\_news

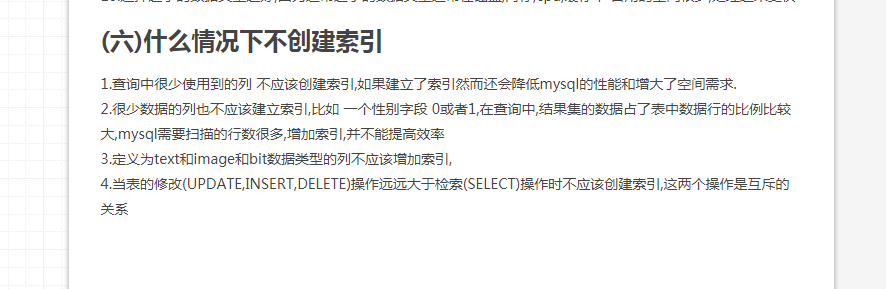
**建立空间索引**

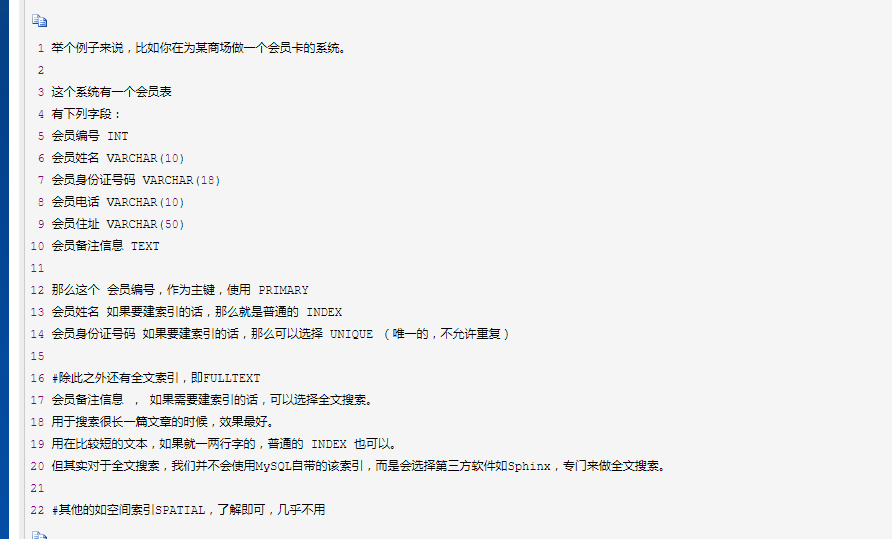
CREATE SPATIAL INDEX spatIdx ON t7(g)

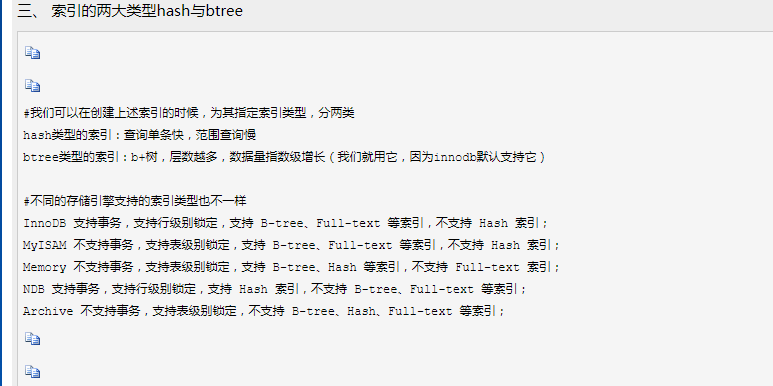
删除索引：

DORP INDEX IndexName ON `TableName`









**Mysql一句sql实现查询数据是否存在，不存在就新增(DUAL是指临时表，这个固定是这个)**

Insert into table(字段1，字段2 )select ‘字段1的数据’,’字段2的数据’ from DUAL where not exists(select 字段1 from table where 字段1=’字段1的数据’)

# 存储过程：

1. 数据库中查看存储过程

Show create 数据库.存储过程名 show create firstsite.p\_add\_role

1. 在存储过程中，用ROW\_COUNT()来判断增删改操作的成功与否，用FOUND\_ROWS()来判断查询操作的成功与否

Eg: if ROW\_COUNT()=1 指成功; else 失败；

**事务：**

事务特性：原子性，一致性，隔离性，持续性 简称ACID特性

**原子性**：事务中的操作要么全部执行，要么全部不执行，不能只完成部分操作，

**一致性**：事务开始之前，数据库处于一致性的状态，事物结束后，数据库必须扔处于一致性状态， 是巨亏的一致性的定义是由用户负责，例如，在银行转账中，用户可以定义转账前后连个账户金额之和保持不变

**隔离性**：视同必须保证事务不收其他并发执行事务的影响，即当多个事务同时运行时，各事务之间相互隔离，不可相互干扰，事务查看数据时所处的状态，要么是另一个并发事务修改它之前的状态，要么是另一个并发事务修改它之后的状态，事务不会查看中间状态的数据，隔离性通过系统的并发控制机制实现

**持久性**：一个已完成的事务对数据所做的任何变动在系统中都是永久有效的，二者必居其一，因此，他对事物的修改具有可恢复性，即当事务失败是，他对数据的修改都会恢复到该事务执行前的状态

事务的 ACID 原则保证了一个事务或者成功提交，或者失败回滚，二者必居其一。因此，它对事务的修改具有可恢复性。即当事务失败时，它对数据的修改都会恢复到该事务执行前的状态。

* 开启事务：Start Transaction
* 事务结束：End Transaction
* 提交事务：Commit Transaction
* 回滚事务：Rollback Transaction

Mysql中开启事物：

DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK; (要在声明变量之后，不能在之前)

然后进行sql语句

Sql语句结束后 commit

Eg:

DECLARE EXIT HANDLER FOR SQLEXCEPTION ROLLBACK;

Set result=0;

Sqly语句填写xxxxxxx

SQL语句结束

COMMIT;

**外键:**

1. 创建表的时候添加外键:

CONSTRAINT `user\_info\_ibfk\_1` FOREIGN KEY (`pk\_userid`)

REFERENCES `user` (`pk\_userid`) ON DELETE CASCADE ON UPDATE CASCADE

1. 给已存在的表添加外键

alter table 子表的数据表名 add constraint FK\_ID foreign key (外键名) REFERENCES 父表的表名 (主键名字)

alter table student add constraint FK\_ID foreign key (gid) REFERENCES grade (id)

alter table goods\_spec add CONSTRAINT `FK\_goods\_spec` FOREIGN KEY(goods\_id) REFERENCES goods(id) ON UPDATE CASCADE ON DELETE CASCADE

1. 给外键设置附加项 例如主键删除数据 关联的外键的数据删除

在添加的外键后面加上

ON UPDATE {CASCADE|SET NULL|NO ACTION|RESTRICT}

CASCADE update/delete 的时候同时对从表进行update或者delete的操作

SET NULL 表示主表update/delete 的时候同时对从表匹配的记录设置为null，但是注意子表的外键不能为not null

no action 不进行任何操作

restrict 表示拒绝主表删除或者修改外键关联列，（在不定义on delete 和on update的时候这是默认设置，也是最安全的设置）

4.删除外键

alter table 表名 drop foreign key 定义的外键

多张表查询不要用join来关联表查询，可以直接单张表的查询



Mysql 一条语句删除多张表

DELETE a,s,g from albums a , goods\_spec s,goods\_spec\_item g where a.goods\_id= s.goods\_id and s.goods\_id = g.goods\_id and s.goods\_id=92

**MySQL用户：**

1. **创建mysql用户**

Create user ‘用户名’@’主机名’ identified by ‘密码’

Create user ‘jiazi123’@’localhost’ identified by ‘tige’

1. **修改用户账号**

**Rename user 旧用户名 to 新用户名**

Rename user ‘xxx’@’localhost’ to ‘xxx’@’xxxx’

1. **修改用户口令(密码)**

Set password for 用户名=

Password（‘xxx’）

**4.删除用户**

Drop user ‘用户名’

**授权用户权限**

**Grant 权限种类 on 数据库范围**

**To ‘用户名’**

**Identified by ‘密码**

**With grant option**

**事例：**

**Grant select insert update delete on \*.\***

**To ‘test123’@’localhost’**

**With grant option;**

**删除用户权限：Revoke**

**Revoke insert on \*.\***

**From ‘test123’@’localhost’**

**数据库备份：**数据库备份是指通过导出数据或者复制表文件的方式来制作数据库的副本。当数据库出现故障或遭到破坏时，将备份的数据库加载到系统，从而使数据库从错误状态恢复到备份时的正确状态。

**select \* from news into OUTFILE 'E:/InforMation/MySql/file2.txt'**

**FIELDS TERMINATED by '\t'**

**LINES TERMINATED by '\n';**