**本节所讲内容：**

1. **字段修饰符**
2. **清空表记录**
3. **索引**
4. **外键**
5. **视图**

**一：字段修饰符 (约束)**

## **1：null和not null修饰符**

**我们通过这个例子来看看**

**mysql> create table worker(id int not null,name varchar(8) not null,pass varchar(20) not null);**

**mysql> insert into worker values(1,'HA','123456');**

**mysql> insert into worker values(1,'LB',null);**

**ERROR 1048 (23000): Column 'pass' cannot be null 不能为null**

**mysql> insert into worker values(2,'HPC','');**

**注：NOT NULL 的字段是不能插入“NULL”的，只能插入“空值”。**

**我们可能有这些疑问<null和not null区别>**

**1、字段类型是not null，为什么可以插入空值**

**2、为什么not null的效率比null高**

**3、判断字段不为空的时候，到底要  select \* from table where column <> '' 还是要用 select \* from table where column is not null 呢。**

**“空值” 和 “NULL”有什么不一样？**

**1、空值是不占用空间的**

**2、mysql中的NULL其实是占用空间的，下面是来自于MYSQL官方的解释**

**“NULL columns require additional space in the row to record whether their values are NULL. For MyISAM tables, each NULL column takes one bit extra, rounded up to the nearest byte.”**

**#“空列需要行中的额外空间来记录其值是否为空。对于MyISAM表，每个NULL列需要一个额外的位，四舍五入到最接近的字节。**

**比如：一个杯子，空值''代表杯子是真空的，NULL代表杯子中装满了空气，虽然杯子看起来都是空的，但是里面是有空气的。**

**对于问题2，为什么not null的效率比null高？**

**NULL 其实并不是空值，而是要占用空间，所以mysql在进行比较的时候，NULL 会参与字段比较，所以对效率有一部分影响。**

**而且索引时不会存储NULL值的，所以如果索引的字段可以为NULL，索引的效率会下降很多。**

**-Mysql难以优化引用可空列查询，它会使索引、索引统计和值更加复杂。可空列需要更多的存储空间，还需要mysql内部进行特殊处理。可空列被索引后，每条记录都需要一个额外的字节，还能导致MyISAM中固定大小的索引变成可变大小的索引--------这也是《高性能mysql第二版》介绍的解读：“可空列需要更多的存储空间”：需要一个额外字节作为判断是否为NULL的标志位“需要mysql内部进行特殊处理”**

**所以使用not null 比null效率高**

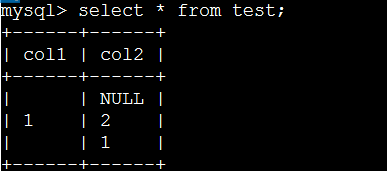
**对于问题3. 判断字段不为空的时候，到底要  select \* from table where column <> '' 还是要用 select \* from table where column is not null我们举例看看**

**mysql> create table test(col1 varchar(10) not null, col2 varchar(10) null)ENGINE=MyISAM;**

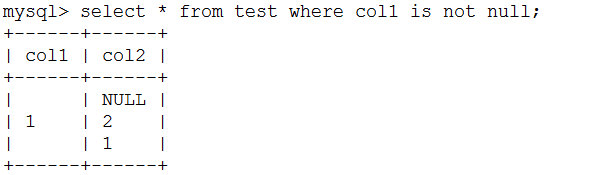
**mysql> insert into test values('',null);**

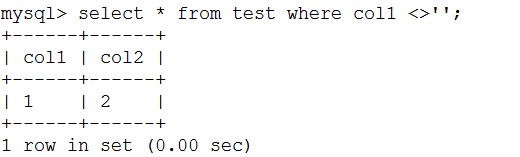
**mysql> insert into test values('1','2');**

**mysql> insert into test values('','1');**

****

**下面我分别用这两条语句查询看看**

****

****

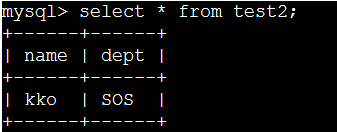
**为空表示不占空间，null占用空间**

## **2：default 设定字段的默认值**

**为字段指定默认的值**

**mysql> create table test2(name varchar(8) not null,dept varchar(25) default 'SOS');**

**mysql> insert into test2 (name) values ('kko');**

****

**总结 ：**

**如果字段没有设定default ,mysql依据这个字段是null还是not null，如果为可以为null，则为null。如果不可以为null，报错。。**

**如果时间字段，默认为当前时间 ，插入0时，默认为当前时间。**

**如果是enum 类型，默认为第一个元素。**

## **3：auto\_increment字段约束**

**自动增长**

**只能修饰 int字段。 表明mysql应该自动为该字段生成一个唯一没有用过的数（每次在最大ID值的基础上加1。特例：如果目前最大ID是34，然后删除34，新添加的会是35.）。对于主键，这是非常 有用的。 可以为每条记录创建一个惟一的标识符**

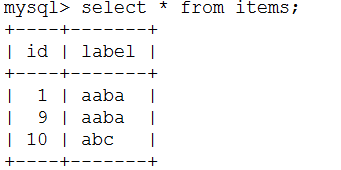
**mysql> create table items ( id int not null auto\_increment primary key , label varchar(20) not null);**

**mysql> insert into items (label) values ('aaba');**

**mysql> insert into items values (9,'aaba');**

**再插入一条id将为多少**

**mysql> insert into items (label) values ('abc');**

****

**Id为10**

**mysql> insert into items values (9,'adl');**

**ERROR 1062 (23000): Duplicate entry '9' for key 'PRIMARY'**

**insert into items (label) values ('abcs'); IDmax =11 max=11**

**delete from items where label='abcs'; IDmax=10 max=11**

**insert into items (label) values ('abcsw'); Idmax=11 max=12**

**主键约束唯一**

# **二：清除表中的记录**

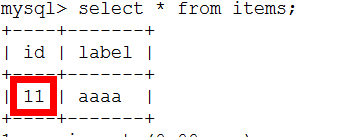
**清空表中所有记录**

## **方法一：delete 不加where条件，清空所有表记录。但是delete不会清零auto\_increment 值**

**mysql> delete from items;**

****

**mysql> insert into items (label) values ("aaaa");**

****

## **方法二：删除表中所有记录，清auto\_increment 值。**

**truncate**

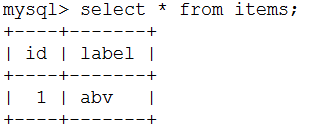
**作用： 删除表的所有记录，并清零auto\_increment 值。新插入的记录从1开始。**

**语法： truncate table name；**

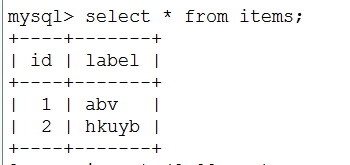
**mysql> truncate table items;**

****

**mysql> insert into items values(null,'abv');**

****

**mysql> insert into items(label)values('hkuyb');**



# **三：索引**

**索引是一种特殊的文件(InnoDB数据表上的索引是表空间的一个组成部分)，它们包含着对数据表里所有记录的引用指针。更通俗的说，数据库索引好比是一本书前面的目录，能加快数据库的查询速度。**

**优点：为了加快搜索速度，减少查询时间 。**

**缺点：**

**1 索引是以文件存储的。如果索引过多，占磁盘空间较大。而且他影响： insert ,update ,delete 执行时间。**

**2索引中数据必须与数据表数据同步：如果索引过多，当表中数据更新的时候后，索引也要同步更新，这就降低了效率。**

## **索引的类型**

**1、普通索引**

**2、唯一性索引**

**3、主键索引（主索引）**

**4、复合索引**

### **普通索引**

**最基本的索引，不具备唯一性，就是加快查询速度**

**创建普通索引:**

**方法一：创建表时添加索引**

**create table 表名（**

**列定义**

**index 索引名称 （字段）**

**index 索引名称 （字段）**

**）**

**注：可以使用key，也可以使用index 。index 索引名称 （字段） ，索引名称，可以加也可以不加，不加使用字段名作为索引名。。**

**mysql> create table demo( id int(4), name varchar(20), pwd varchar(20), index(pwd) );**

**注意：index和 key 是相同的**

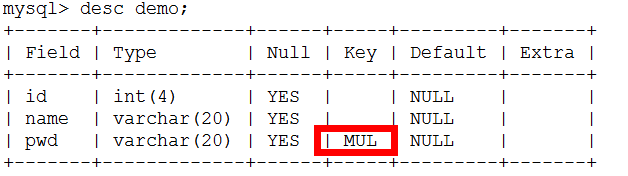
**mysql> create table demo1( id int(4), name varchar(20), pwd varchar(20), key(pwd) );**

**mysql> create table demo2( id int(4), name varchar(20), pwd varchar(20), key index\_pwd(pwd) ); #加上名称**

**方法二： 当表创建完成后，使用alter为表添加索引：**

**alter table 表名 add index 索引名称 （字段1，字段2.....）;**

**查看索引**

****

**注：如果Key是MUL, 就是一般性索引，该列的值可以重复, 该列是一个非唯一索引的前导列(第一列)或者是一个唯一性索引的组成部分但是可以含有空值NULL。就是表示是一个普通索引。**

**我们先删除索引**

**mysql> alter table demo drop key pwd; 注意此处的pwd指的是索引的名称,而不是表中pwd的那个字段**

**再用alter添加**

**mysql> alter table demo add key(pwd);**

### **唯一索引**

**与普通索引基本相同，但有一个区别：索引列的所有值都只能出现一次，即必须唯一，用来约束内容，字段值只能出现一次。应该加唯一索引。唯一性允许有NULL值<允许为空>。**

**创建唯一索引：**

**方法一：创建表时加唯一索引**

**create table 表名(**

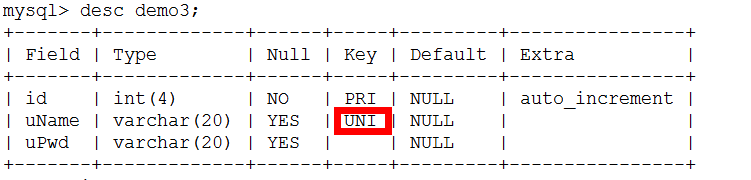
**列定义：**

**unique key 索引名 （字段）;**

**)**

**注意：常用在值不能重复的字段上，比如说用户名，电话号码，身份证号。**

**mysql> create table demo3(id int(4)** **auto\_increment primary key, uName varchar(20), uPwd varchar(20), unique index (uName));**

****

**方法二：修改表时加唯一索引**

**alter table 表名 add unique 索引名 (字段);**

**mysql> alter table demo3 drop key uName;**

**mysql> alter table demo3 add** **unique(uName);**

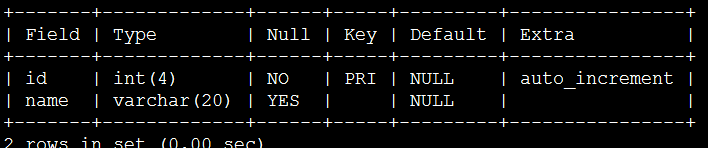
### **主键索引**

**查询数据库，按主键查询是最快的，每个表只能有一个主键列，可以有多个普通索引列。主键列要求列的所有内容必须唯一，而索引列不要求内容必须唯一，不允许为空**

**创建主键索引**

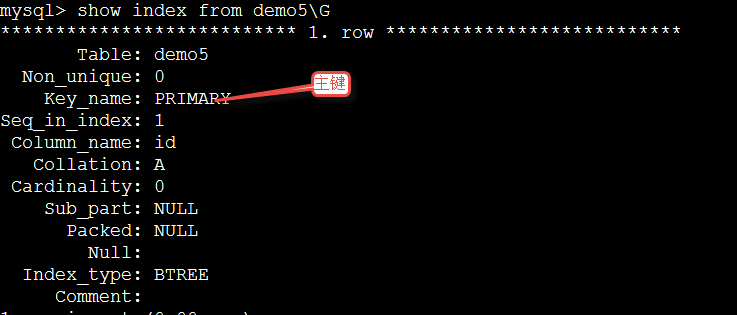
**方法一：创建表创建主键索引**

**mysql> create table demo5( id int(4) not null auto\_increment, name varchar(20) default null,primary key(id));**

****

**mysql> show create table demo5;**

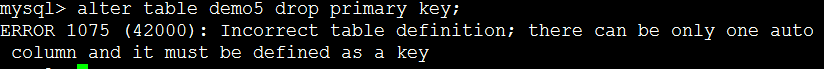
**show index from demo5 \G**

****

**方法二：创建表后添加<不推荐>如果生产的数据无法保证唯一，创建主键报错**

**再添加**

**先删除测试**

****

**删除遇到这种情况是auto\_increment的原因**

**mysql> alter table demo5 change id id int(4) not null; 先取消自增长**

**mysql> alter table demo5 drop primary key; 再删除主键**

**mysql> alter table demo5 change id id int(4) not null primary key auto\_increment;**

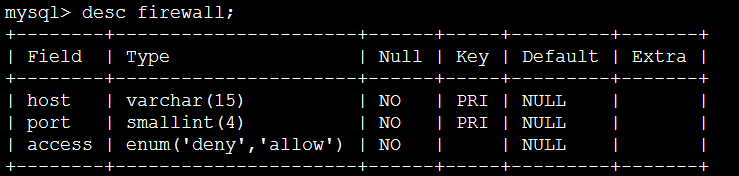
**总结：主键索引，唯一性索引区别：主键索引不能有NULL，唯一性索引可以有空值**

### **复合索引**

**索引可以包含一个、两个或更多个列。两个或更多个列上的索引被称作复合索引**

**例： 创建一个表存放服务器允许或拒绝的IP和port，要记录中IP和port要唯一。**

**mysql> create table firewall ( host varchar(15) not null ,port smallint(4) not null ,access enum('deny','allow') not null, primary key (host,port)); 联合主键**

****

**mysql> insert into firewall values('10.96.52.46',22,'deny');**

**mysql> insert into firewall values('10.96.52.46',21,'allow');**

**mysql> insert into firewall values('10.96.52.46',21,'allow');**

**ERROR 1062 (23000): Duplicate entry '10.96.52.46-21' for key 'PRIMARY'**

**插入一样就报错，唯一**

**总结：**

**建表的时候如果加各种索引，顺序如下：**

**create table 表名（字段定义，PRIMARYKEY (`bId`),UNIQUE KEY `bi` (`bImg`),KEY `bn` (`bName`),KEY `ba` (`author`)）**

### **全文索引 (FULLTEXT INDEX)**

**全文索引(也称全文检索)是目前搜索引擎使用的一种关键技术。它能够利用「分词技术「等多种算法智能分析出文本文字中关键字词的频率及重要性，然后按照一定的算法规则智能地筛选出我们想要的搜索结果**

**mysql 在数据量较大的情况下，高并发连接的情况下。**

**select 语句 where bName like '%网%'**

**使用% \_ 通配符，不通过索引，直接全表扫描。**

**ABSUWU LIKE ‘%U\_U’**

**数据库压力大。**

**mysql的解决方案：全文索引：3.2开始支持全文索引。无法正确支持中文。**

**从MySQL 5.7.6开始 MySQL内置了ngram全文检索插件，用来支持中文分词**

**你的表当前默认的存储引擎:**

**mysql> show create table 表名;**

**全文索引只能用在 varchar text**

**创建全文索引：**

**方法一：创建表时创建**

**create table 表名（**

**列定义，**

**fulltext key 索引名 （字段）;**

**）**

**方法二：修改表时添加**

**alter table 表名 add fulltext 索引名 （字段）;**

**ALTER TABLE `books` ADD FULLTEXT [索引名] (`author` )**

**强烈注意：MySQL自带的全文索引只能用于数据库引擎为MyISAM的数据表，如果是其他数据引擎，则全文索引不会生效**

**一般交给第三方软件进行全文索引**

**http://sphinxsearch.com/**

### **索引设计原则：**

1. **索引并非越多越好**
2. **数据量不大不需要建立索引**
3. **列中的值变化不多不需要建立索引 row id**
4. **经常排序（order by 字段）和分组（group by 字段）的列需要建立索引**

**select a.bTypeId,(select b.bTypeName from category b where a.bTypeId = b.bTypeId) bn,count(\*) from books a group by bTypeId;**

1. **唯一性约束对应使用唯一性索引**

**Table (id pri,use,name index,pass)**

# **四：外键约束**

### **什么是外键约束：**

**foreign key就是表与表之间的某种约定的关系，由于这种关系的存在，我们能够让表与表之间的数据，更加的完整，关连性更强。**

**关于完整性，关连性我们举个例子**

**有二张表，一张是用户表，一张是订单表：**

**1》如果我删除了用户表里的用户，那么订单表里面与这个用户有关的数据，就成了无头数据了，不完整了。**

**2》如果我在订单表里面，随便插入了一条数据，这个订单在用户表里面，没有与之对应的用户。这样数据也不完整了。**

**如果有外键的话，就方便多了，可以不让用户删除数据，或者删除用户的话，通过外键同样删除订单表里面的数据，这样也能让数据完整。**

### **创建外键约束：**

**外键： 每次插入或更新时，都会检查数据的完整性。**

#### **方法一：通过create table创建外键**

**语法：**

**create table 数据表名称(**

**...,**

**[CONSTRAINT [约束名称]] FOREIGN KEY [外键字段]   
    REFERENCES [外键表名](外键字段，外键字段2…..)  
    [ON DELETE C**[**AS**](http://www.jzxue.com/tag/FlashAS/)**CADE ]  
    [ON UPDATE CASCADE ]**

**)**

**关于参数的解释：**

**RESTRICT: 拒绝对父表的删除或更新操作。**

**CASCADE: 从父表删除或更新且自动删除或更新子表中匹配的行。ON DELETE CASCADE和ON UPDATE CASCADE都可用**

**注意：on update cascade是级联更新的意思，on delete cascade是级联删除的意思，意思就是说当你更新或删除主键表，那外键表也会跟随一起更新或删除。**

**精简化后的语法：**

**语法：foreign key 当前表的字段 references 外部表名 （关联的字段） ENGINE =innodb**

**注：创建成功，必须满足以下4个条件：**

**1、确保参照的表和字段存在。**

**2、组成外键的字段被索引。**

**3、必须使用ENGINE指定存储引擎为：innodb.**

**4、外键字段和关联字段，数据类型必须一致。**

**例子：我们创建一个数据库，包含用户信息表和订单表**

**mysql> create database market;**

**mysql> use market;**

**mysql> create table `user`(id int(11) not null auto\_increment, name varchar(50) not null default '', sex int(1) not null default '0', primary key(id))ENGINE=innodb;**

**#创建时，如果表名是sql关键字，使用时，需要使用反引号``**

**mysql> create table `order`(o\_id int(11) auto\_increment, u\_id int(11) default '0', username varchar(50), money int(11), primary key(o\_id), index(u\_id), foreign key order\_f\_key(u\_id) references user(id) on delete cascade on update cascade) ENGINE=innodb;**

**注：**

**1:on delete cascade on update cascade 添加级联删除和更新：**

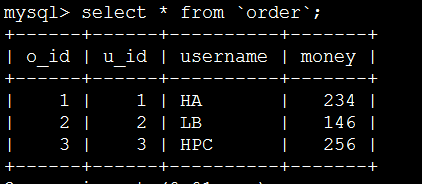
**2: ：确保参照的表user中id字段存在。 组成外键的字段u\_id被索引。 必须使用type指定存储引擎为：innodb。**

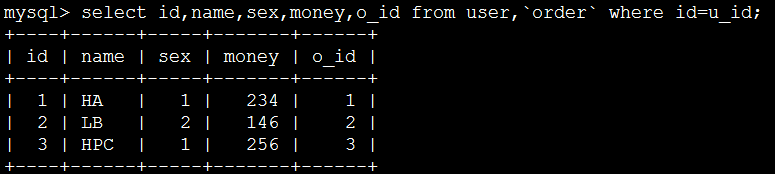
**外键字段和关联字段，数据类型必须一致。**

**插入测试数据**

**mysql> insert into user(name,sex)values('HA',1),('LB',2),('HPC',1);**

**mysql> insert into `order` (u\_id,username,money)values(1,'HA',234),(2,'LB',146),(3,'HPC',256);**

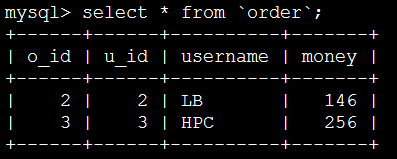
****

****

**测试级联删除：**

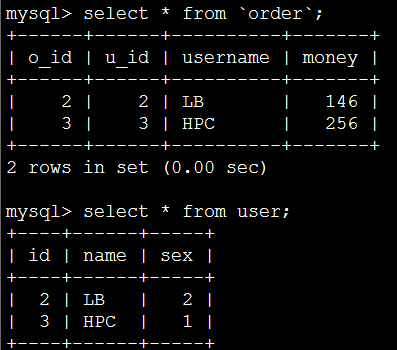
**mysql> delete from user where id=1; 删除user表中id为1的数据**

**再查看order表**

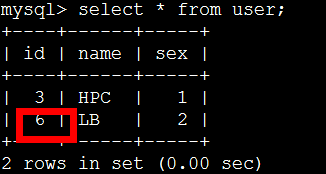
****

**测试级联更新：**

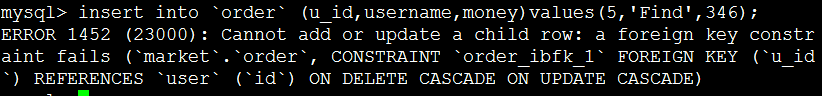
**更新前数据状态**

****

**mysql> update user set id=6 where id=2;**

****

**测试数据完整性**

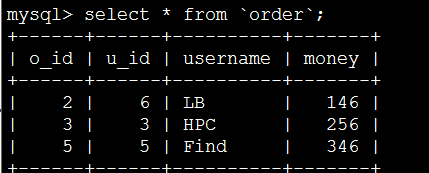
****

**外键约束，order表受user表的约束**

**在order里面插入一条数据u\_id为5用户，在user表里面根本没有，所以插入不进去**

**mysql> insert into user values(5,'Find',1);**

**mysql> insert into `order` (u\_id,username,money)values(5,'Find',346); 这里u\_id 只能是5**

****

#### **方法二：通过alter table 创建外键和级联更新，级联删除**

**语法：**

**alter table 数据表名称 add**

**[constraint [约束名称] ] foreign key (外键字段,..) references 数据表(参照字段,...)**

**[on update cascade|set null|no action]**

**[on delete cascade|set null|no action]**

**)**

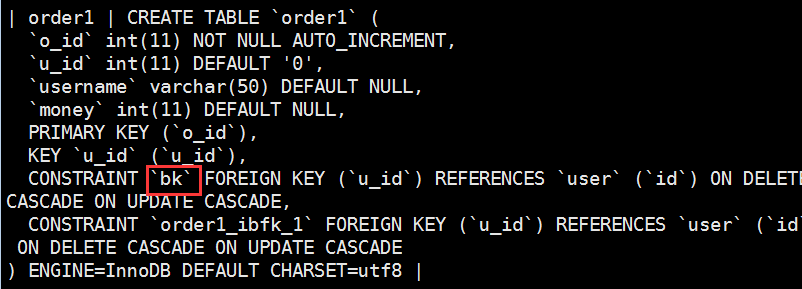
**mysql> create table order1(o\_id int(11) auto\_increment, u\_id int(11) default '0', username varchar(50), money int(11), primary key(o\_id), index(u\_id)) ENGINE=innodb;**

**mysql> alter table order1 add foreign key(u\_id) references user(id) on delete cascade on update cascade, ENGINE =innodb;**

**mysql> alter table order1 add constraint `bk`foreign key(u\_id) references user(id) on delete cascade on update cascade,type=innodb; 指定外键名称**

**一定要记得带上innodb**

**mysql> show create table order1;**



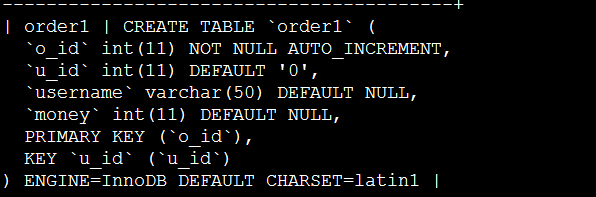
### **删除外键：**

**语法**

**alter table 数据表名称 drop foreign key 约束（外键）名称**

**mysql> alter table order1 drop foreign key order1\_ibfk\_1;**

**mysql> show create table order1;**



# **五：视图**

### **什么是视图**

**视图就是一个存在于数据库中的虚拟表。**

**视图本身没有数据，只是通过执行相应的select语句完成获得相应的数据。**

**我们在怎样的场景使用它，为什么使用视图**

**如果某个查询结果出现的非常频繁，也就是，要经常拿这个查询结果来做子查询这种**

1. **视图能够简化用户的操作**

**视图机制用户可以将注意力集中在所关心的数据上。如果这些数据不是直接来自基本表，则可以通过定义视图，使数据库看起来结构简单、清晰，并且可以简化用户的数据查询操作**

1. **视图是用户能以不同的角度看待同样的数据。**

**对于固定的一些基本表，我们可以给不同的用户建立不同的视图，这样不同的用户就可以看到自己需要的信息了。**

1. **视图对重构数据库提供了一定程度的逻辑性。**

**比如原来的A表被分割成了B表和C表，我们仍然可以在B表和C表的基础上构建一个视图A，而使用该数据表的程序可以不变。**

1. **视图能够对机密数据提供安全保护**

**比如说，每门课的成绩都构成了一个基本表，但是对于每个同学只可以查看自己这门课的成绩，因此可以为每个同学建立一个视图，隐藏其他同学的数据，只显示该同学自己的**

1. **适当的利用视图可以更加清晰的表达查询数据。**

**有时用现有的视图进行查询可以极大的减小查询语句的复杂程度。**

### **创建视图**

**语法：create view视图名称（即虚拟的表名） as select 语句。**

**我们在book数据库中操作**

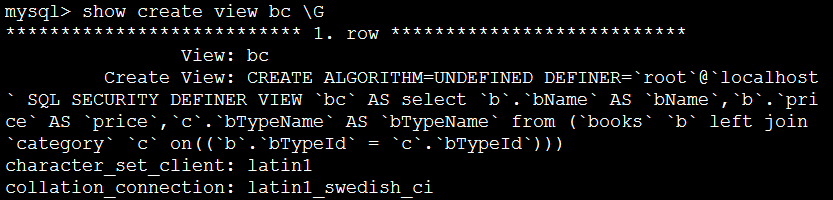
**mysql> create view bc as select b.bName ,b.price ,c.bTypeName from books as b left join category as c on b.bTypeId=c.bTypeId ;**

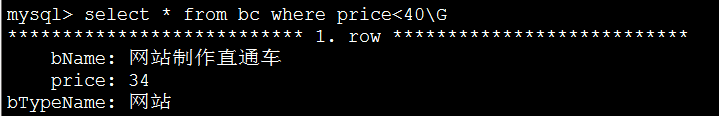
**可以按照普通表去访问。**

**另外视图表中的数据和原数据表中数据是同步的。**

**查看视图创建信息：**

**mysql> show create view bc \G**

** 查询视图中的数据**

****

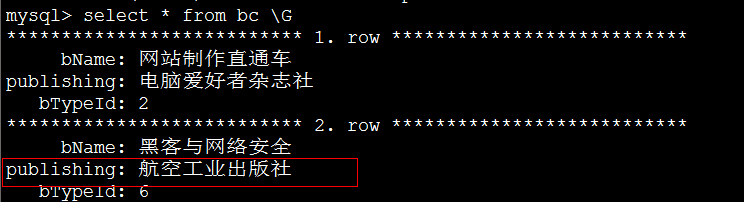
### **更新或修改视图**

**语法：**

**alter view视图名称（即虚拟的表名） as select 语句。**

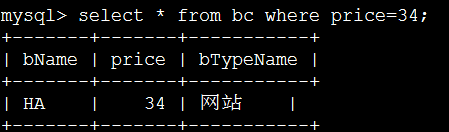
**update view视图名称（即虚拟的表名）set**

**mysql> alter view bc as select b.bName ,b.publishing ,c.bTypeId from books as b left join category as c on b.bTypeId=c.bTypeId ;**

****

**更新**

**mysql> update bc set bName='HA' where price=34;**

****

### **删除视图**

**drop view 视图名。**

**mysql> drop view bc;**

**学习mysql 先搞懂它的概念原理，慢慢做实验理解它真正的用法，然后结合实际为什么要用这个sql写法。**