

**数据库系统原理实践报告**

|  |  |
| --- | --- |
| 姓 名： | 江易星 |
| 学 院： | 计算机科学与技术学院 |
| 专 业： | 计算机科学与技术 |
| 班 级： | ACM1601 |
| 学 号： | U201614756 |
| 指导教师： | 袁平鹏 |

|  |  |
| --- | --- |
| 分数 |  |
| 教师签名 |  |

2019年 05 月 19 日

目 录

[1 课程任务概述 1](#_Toc9155697)

[2 软件功能学习 2](#_Toc9155698)

[2.1 任务要求 2](#_Toc9155699)

[2.2 完成过程 3](#_Toc9155700)

[2.2.1 练习两种完全备份方式 3](#_Toc9155701)

[2.2.2练习增加用户并配置权限 5](#_Toc9155702)

[2.3任务总结 7](#_Toc9155703)

[3 SQL练习 7](#_Toc9155704)

[3.1 任务要求 7](#_Toc9155705)

[3.1.1 建表 7](#_Toc9155706)

[3.1.2 数据更新 9](#_Toc9155707)

[3.1.3 查询 9](#_Toc9155708)

[3.1.4 DBMS函数及存储过程 10](#_Toc9155709)

[3.2 完成过程 10](#_Toc9155710)

[3.2.1 建表 10](#_Toc9155711)

[3.2.2 数据更新 10](#_Toc9155712)

[3.2.3 查询 10](#_Toc9155713)

[3.2.4 DBMS函数及存储过程和事务 10](#_Toc9155714)

[.3任务总结 14](#_Toc9155715)

[4 应用系统设计 15](#_Toc9155716)

[4.1系统设计目标 15](#_Toc9155717)

[4.2 需求分析 15](#_Toc9155718)

[4.3 总体设计 15](#_Toc9155719)

[4.4 数据库设计 15](#_Toc9155720)

[4.5 详细设计与实现 15](#_Toc9155721)

[4.6 系统测试 15](#_Toc9155722)

[4.7 系统设计与实现总结 15](#_Toc9155723)

[5 课程总结 15](#_Toc9155724)

[附录 16](#_Toc9155725)

# 1 课程任务概述

1 熟悉Microsoft　SQL　Server　Management　Studio的使用，练习备份和创建用户和配置用户权限等基本功能的使用。

２利用SQL语言实现对数据库的增删改查等操作。

３进行一个数据库应用系统的设计，完成一个小型数据库应用系统的设计与实现工作。

# 2 软件功能学习

## 2.1 任务要求

完成下列1-2题，并在实验报告中叙述过程，可适当辅以插图

1）练习sqlserver的两种完全备份方式：数据和日志文件的脱机备份。

2）练习在新增的数据库上增加用户并配置权限的操作，通过用创建的用户登录数据库并且执行未经授权的SQL语句验证自己的权限配置是否成功。

## 2.2 完成过程

（包括主要操作步骤描述及其执行效果，或者所用的SQL语句及语句执行、调试的主要过程、效果。）

### 2.2.1 练习两种完全备份方式

首先建立一个数据库，建立的数据库如图1所示：

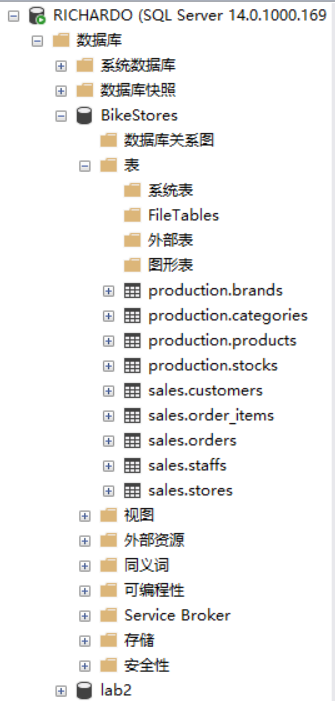


图1：建立的测试数据库

数据的备份需要两个步骤，首先需要创建备份设备，然后才能进行备份。代码如图2所示：

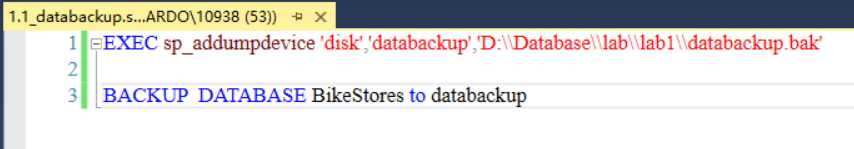


图2：数据备份代码

代码的第一行创建了一个逻辑名叫 databackup的备份设备，对应D:\Database\lab\lab1\databackup.bak文件。代码的第三行实现将BikeStore数据库备份到databackup设备中。

打开D:\Database\lab\lab1\databackup.bak文件，查看文件内容如图3所示：

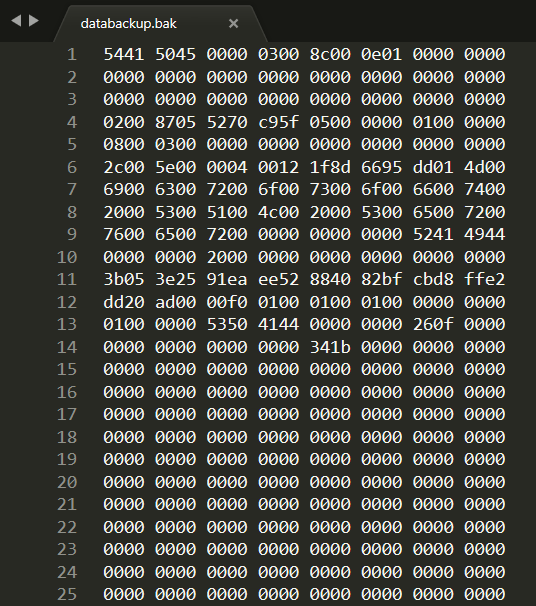


图3：文件备份内容

用同样方法把日志文件备份到该备份设备，代码如图4所示：

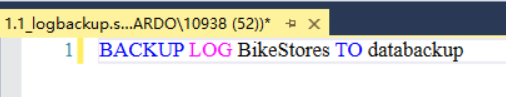


图4：备份日志

执行，发现出现以下报错信息，如图5所示：

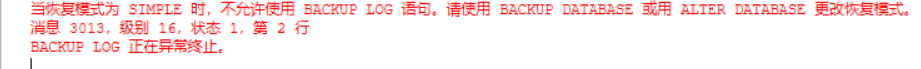


图5：备份日志报错

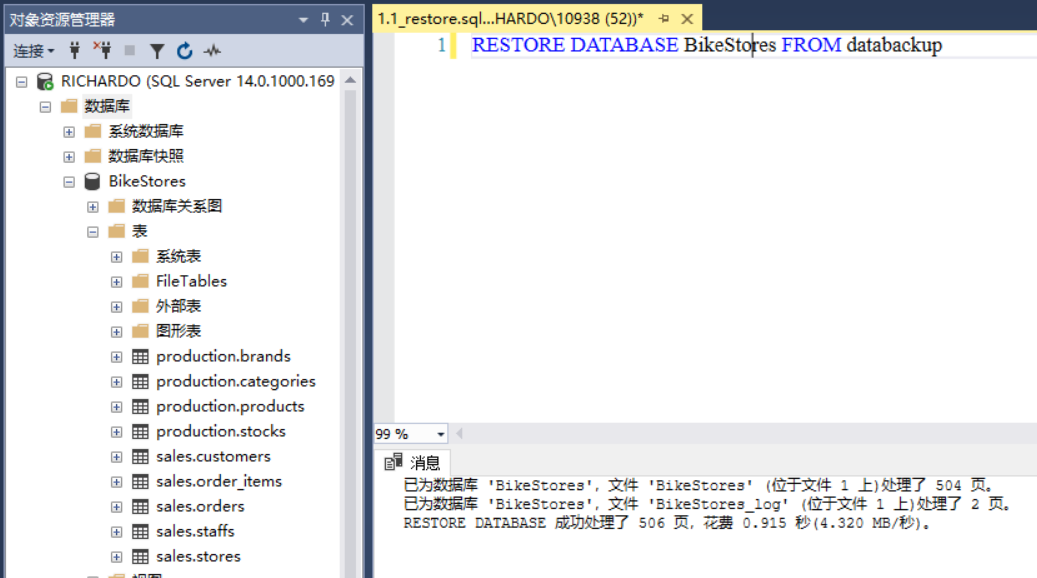
查阅微软的官方发现这是由于数据库的恢复模式设置为“简单”时，是不允许使用日志备份的，只需要将恢复模式修改为不是“简单”即可。具体方法是在数据库右击，选择属性，然后在选项页中选中选项，即可在右边设置恢复模式。

由于备份到同一个文件时，sqlserver会将后面的数据追加到原来数据的后面，所以这里不好区分数据库数据和日志数据，所以未给出图示。

删除数据库BikeStores，然后使用以下命令：

RESTORE　DATABASE BikeStores FROM databackup

执行词条命令之后刷新，发现数据库BikeStore又恢复了，如图6所示：



题6：恢复数据库

### 2.2.2练习增加用户并配置权限

在根目录下选择安全性->登录名，然后右击新建登录名，新建一个登录名为user1，密码为user1的新用户。如图7所示：

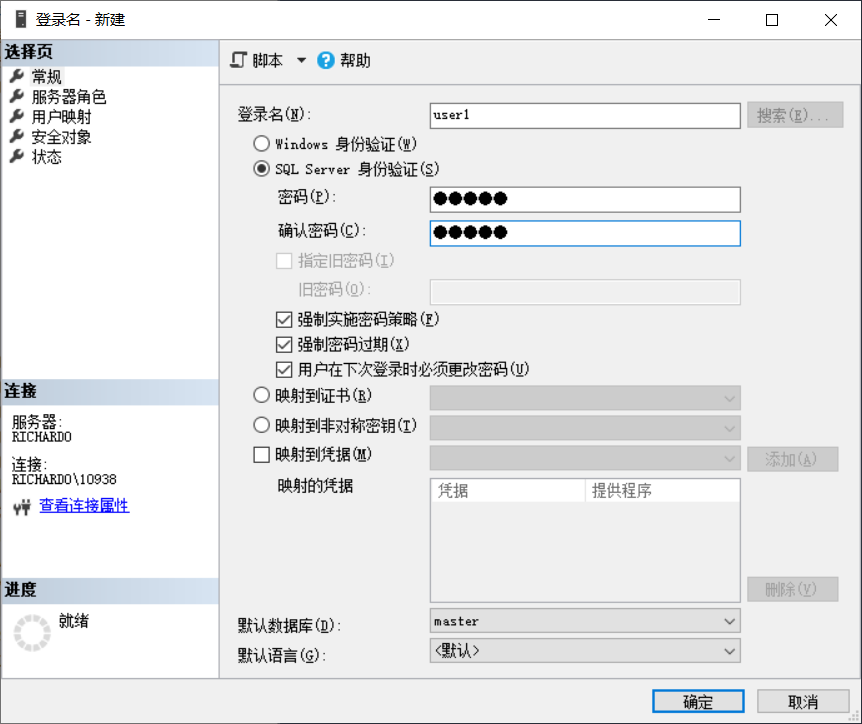


图7：创建一个新用户

接着在数据库BikeStore中添加一个用户:选择数据库BikeStore -> 安全性 -> 用户 右击选择新建用户,将user1添加进来,如图8所示:

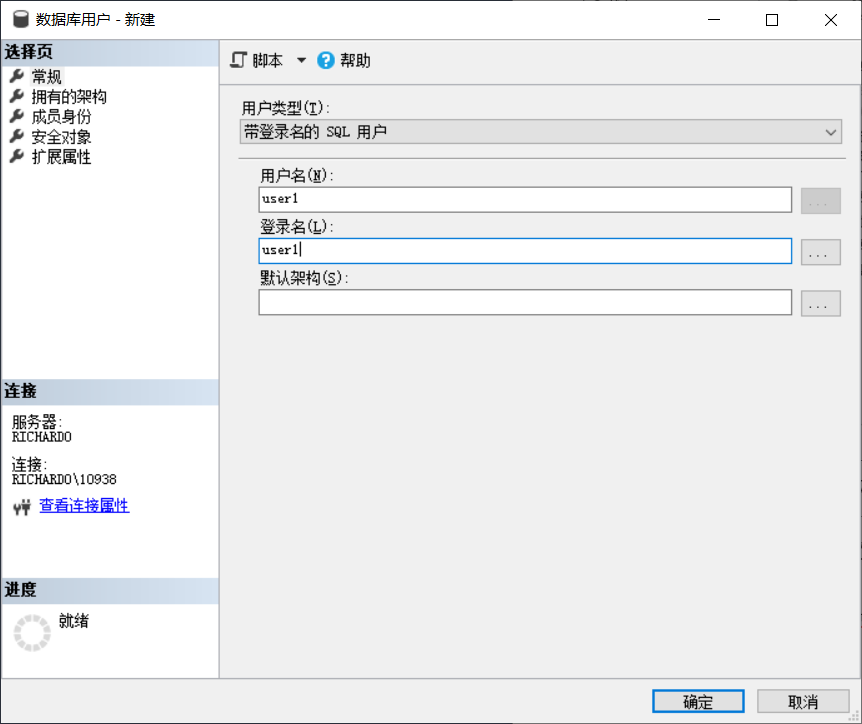


图8:在BikeStores数据库中添加用户user1

然后选择文件->断开与对象资源管理器的连接，身份验证选择 SQL Server身份验证，输入登录名user1，密码user1.连接。

连接之后我们查看BikeStores数据库，展开表，如图9所示：



图9：user1查看BikeStores数据库中的表

我们可以发现，这时候user1无法访问BikeStores数据库中的任何表。原因是我们没有给user1赋予任何的权限。所以切换到windows登录，给user1赋予对production.brands表赋予select的权限，对应的SQL语句：

|  |
| --- |
| GRANT select ON production.brands TO user1 |

再切换到user1登录，登录之后我们再次点击BikeStores数据库，点击表，这时可以看到production.brands表格，右键->选择前1000行可以看到表中内容，如图10所示：



图10：配置权限之后

## 2.3任务总结

当第一次采用创建的新用户user1登陆时，遇到如下错误，如图11所示：

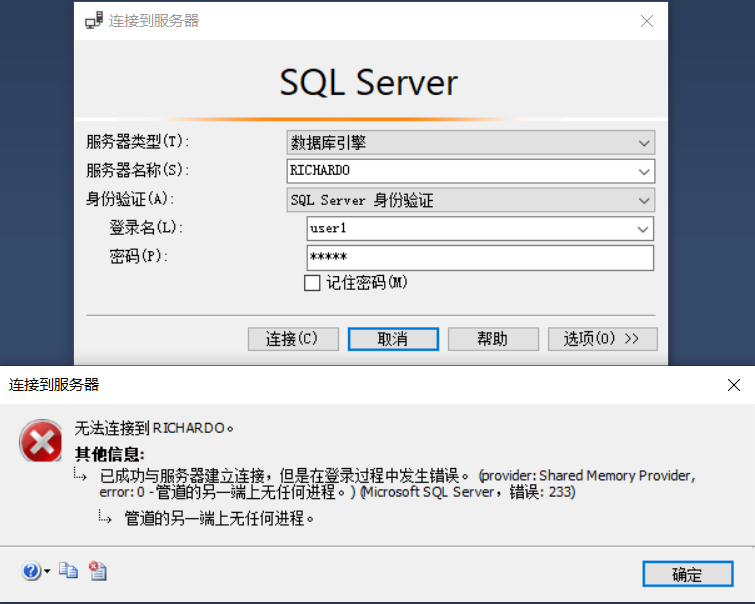


图11：报错信息

再网上搜索资料知道：这是由于登录模式设置为了windows登录导致的报错。修改方法为：在服务器图标（即对象资源管理器中第一行中右下角带绿色的图标）右击，选择属性，安全性。即可修改身份验证模式，如图12所示：

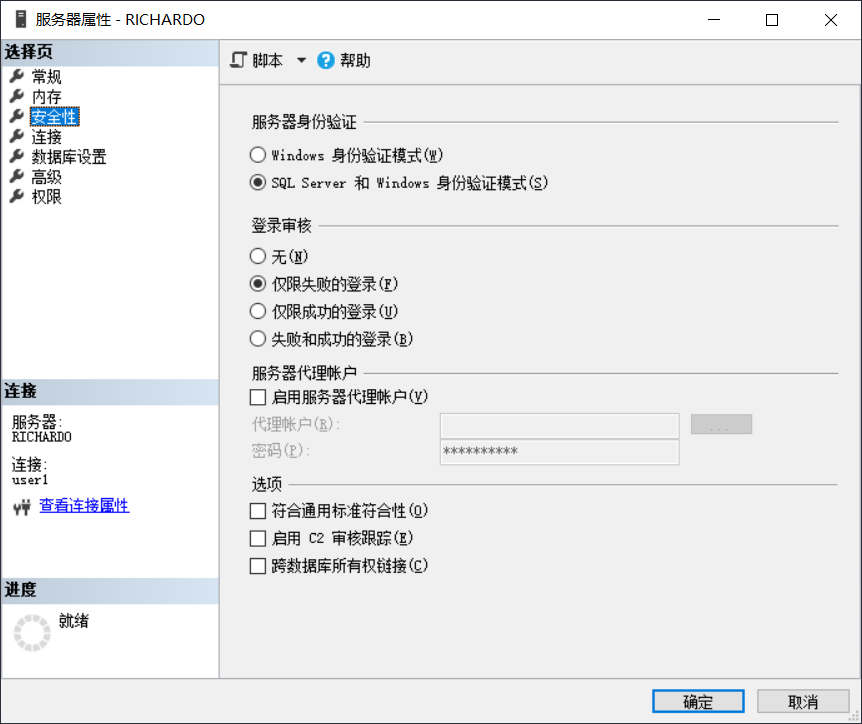


图12：修改身份验证方式

# 3 SQL练习

## 3.1 任务要求

### 3.1.1 建表

1）假设某微博平台的数据库中有下列关系，请在DBMS中创建这些关系，包括主码和外码的说明，并写出指定关系的建表SQL语句：

用户【用户ID，姓名，性别，出生年份，所在城市】记录所有注册用户的基本信息，英文表名和字段名如下：

USER(UID 整型，NAME 字符串，SEX 一位汉字，BYEAR 整型，CITY 字符串 )；

分类【分类ID，分类名称】记录微博平台中所有可能涉及的微博的类型，例如文学、艺术、军事等，英文表名和字段名如下：

LABEL(LID 整型，LNAME 字符串)；

博文【博文ID，标题，用户ID，年，月，日，正文】记录每一篇微博的基本信息，英文表名和字段名如下：

MBLOG(BID 整型，TITLE 字符串，UID 整型，PYEAR 整型，PMONTH 整型， PDAY 整型，CONT 字符串)，

写出该关系的建表SQL语句；

博文标注【博文ID，分类ID】记录每一篇微博的作者给该微博贴上的分类标签，一篇微博可以涉及不止一种分类，英文表名和字段名如下：

B\_L(BID 整型，LID 整型)；

关注【用户ID，被关注用户ID】记录每位用户关注的其他用户，每位用户可关注多人，英文表名和字段名如下：

FOLLOW(UID 整型，UIDFLED 整型)；

好友【用户ID， 好友ID】记录每位用户的好友（可多个），英文表名和字段名如下：

FRIENDS(UID 整型，FUID 整型);

订阅【用户ID, 订阅分类ID】记录用户订阅的每一种分类，英文表名和字段名如下：

SUB(UID 整型，LID 整型);

点赞【用户ID， 博文ID】记录用户点赞的每一篇微博，英文表名和字段名如下：

THUMB(UID 整型，BID 整型)，

写出该关系的建表SQL语句；

头条【年，月，日，博文ID，顺序号】记录每一天的热度排名前十的博文ID号以及该博文在热度前十名中的排名，英文表名和字段名如下：(这里的主码可以有多种情形)

TOPDAY(TYEAR整型，TMONTH 整型，TDAY 整型，BID 整型，TNO 整型)。

2）观察性实验

用户在订阅分类时是否一定要参考被参照关系的主码，并在实验报告中简述过程和结果。

3）数据准备

依据后续实验的要求，向上述表格中录入适当数量的实验数据，从而对相关的实验任务能够起到验证的作用。

### 3.1.2 数据更新

1）分别用一条sql语句完成对博文表基本的增、删、改的操作；

2）批处理操作

将关注3号用户的用户信息插入到一个自定义的新表FANS\_3中。

3）数据导入导出

通过查阅DBMS资料学习数据导入导出功能，并将任务2.1所建表格的数据导出到操作系统文件，然后再将这些文件的数据导入到相应空表。

在后续的上机实验环节，通过导入导出或者备份机制实现前次上机环节的数据恢复。

4）观察性实验

建立一个关系，但是不设置主码，然后向该关系中插入重复元组，然后观察在图形化交互界面中对已有数据进行删除和修改时所发生的现象。

5）触发器实验

编写一个触发器，用于实现对点赞表的完整性控制规则：当插入或者更新点赞关系时，如果博文作者就是点赞者本人，则拒绝执行。

### 3.1.3 查询

请分别用一条SQL语句完成下列各个小题的需求：

1）查询“张三”用户关注的所有用户的ID号、姓名、性别、出生年份，所在城市，并且按照出生年份的降序排列，同一个年份的则按照用户ID号升序排列。

2）查找没有被任何人点赞的博文ID、标题以及发表者姓名，并将结果按照标题字符顺序排列。

3）查找2000年以后出生的武汉市用户发表的进入过头条的博文ID；

4）查找订阅了所有分类的用户ID；

5）查找出生年份小于1970年或者大于2010年的用户ID、出生年份、所在城市，要求where子句中只能有一个条件表达式；

6）统计每个城市的用户数；

7）统计每个城市的每个出生年份的用户数，并将结果按照城市的升序排列，同一个城市按照出生用户数的降序排列其相应的年份；SELCT

8）查找被点赞数超过1的博文ID号；

9）查找被2000年后出生的用户点赞数超过1的博文ID号；

10）查找被2000年后出生的用户点赞数超过1的每篇博文的进入头条的次数；

11）查找订阅了文学、艺术、哲学、音乐中至少一种分类的用户ID，要求不能使用嵌套查询，且where子句中最多只能包含两个条件；

12）查找标题中包含了“最多地铁站”和“\_华中科技大学”两个词的博文基本信息；

13）查找所有相互关注的用户对的两个ID号，要求不能使用嵌套查询；

14）查找好友圈包含了5号用户好友圈的用户。

15）查找2019年4月20日每一篇头条博文的ID号、标题以及该博文的每一个分类ID，要求即使该博文没有任何分类ID也要输出其ID号、标题；（实现有问题）

16）查找至少有3名共同好友的所有用户对的两个ID号。

17）创建视图：查阅DBMS内部函数，创建前十的微博信息的视图，把博客对应的时间打印出来，其中的属性包括：博文ID、博文标题、发表者ID、发表者姓名、被点赞数。

### 3.1.4 DBMS函数及存储过程

建立关系“点赞排行榜【博文ID，当天点赞人数】”，里面存储系统当天点赞数前十名的博文ID及其点赞人数，尝试编写一个DBMS的存储过程，通过该存储过程更新该表。

## 3.2 完成过程

### 3.2.1 建表

先创建一个叫lab2 的数据库，然后再在这个数据库中创建所有的表格。

1）用户表的创建

创建用户表USER，属性包括

UID：int

NAME：字符串，这里用VARCHAR(64)实现

SEX：一位汉字，用CHAR(2)实现，因为一个汉字编码占两个字节

BYEAR：int

CITY：字符串，这里用VARCHAR(64)实现

主码为 UID。

实现的SQL语句如图13所示：

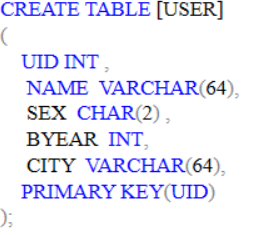


图13：创建USER表

2）分类表的创建

分类表LABEL的属性包括：

LID：int

LNAME：字符串，这里用VARCHAR(64)实现

LID为主码。

创建LABEL表格的SQL语句如图14所示：

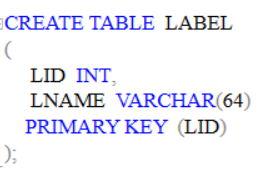


图14：创建LABEL表

3）博文表的创建

博文表MBLOG的属性包括：

BID：int

TITLE：字符串，这里用VARCHAR(128)实现

UID：int

PYEAR：int

PMONTH：int

PDAY：int

CONT：字符串，这里用VARCHAR(1024)实现

BID是主码，UID是外码，应用用户表USER的属性UID

创建MBLOG表的SQL语句如图15所示：

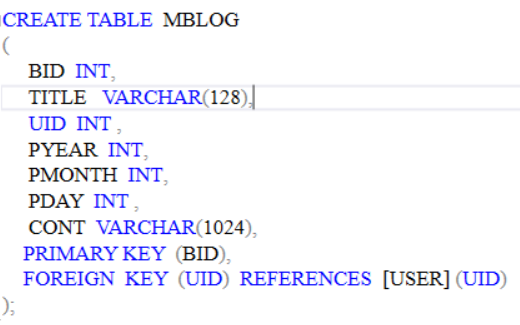


图15：创建MBLOG表

4）博文标注表的创建

博文标注表B\_L表的属性包括：

BID：int

LID：int

主码是(BID，LID)，BID是外码，应用MBLOG表的BID，LID是外码，应用了LABEL表的LID。

创建B\_L表的SQL语句如图16所示：

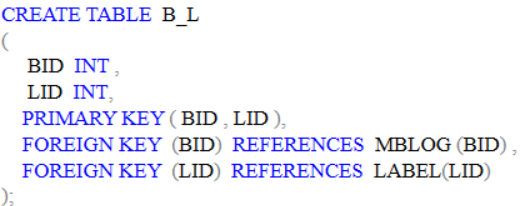


图16：创建B\_L表

5）关注表的创建

关注表FOLLOW的属性包括：  
UID：int

UIDFLD：int

主码是(UID,UIDFLD)，UID是外码，引用表USER的UID，UIDFLD是外码，引用USER表的UID。

创建FOLLOW表的SQL语句如图17所示：

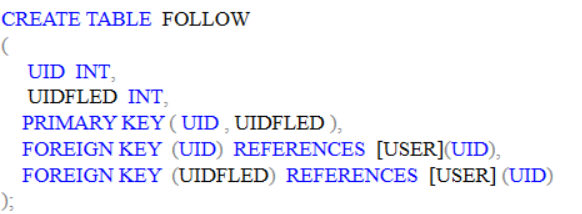


图17：创建FOLLOW表

6）好友表的创建

好友表FRIENDS的属性包括：

UID：int

FUID：int

主码是(UID，FUID)，UID是外码，引用USER表的UID，FUID是外码，引用USER表的UID。

创建FRIENDS表的SQL语句如图18所示：

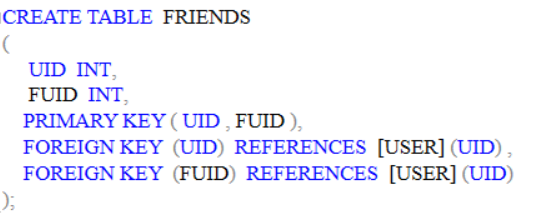


图18：创建FRIENDS表

7）订阅表的创建

订阅表SUB的属性包括：

UID：int

LID：int

主码是(UID，LID)，UID是外码，引用USER表的UID，LID是外码，引用LABEL表的LID

创建SUB表的SQL语句如图19所示：

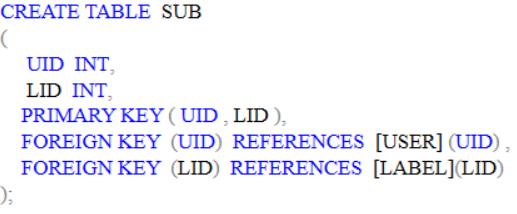


图19：创建SUB表

8）点赞表的创建

点赞表THUMB的属性包括：

UID：int

BID：int

主码是(UID,BID)，UID是外码，引用USER表的UID，BID是外码，引用表MBLOG表的BID。

创建THUMB表的SQL语句如图20所示

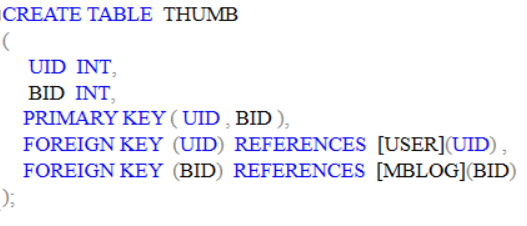


图20：创建THUMB表

9）头条表的创建

头条表TOPDAY的属性包括：

TYEAR：int

TMONTH：int

TDAY：int

BID：int

TNO：int

主码是(TEAR,TMONTH,TDAY,BID)，BID是外码，引用MBLOG表的BID

创建TOPDAY表的SQL语句如图21所示：

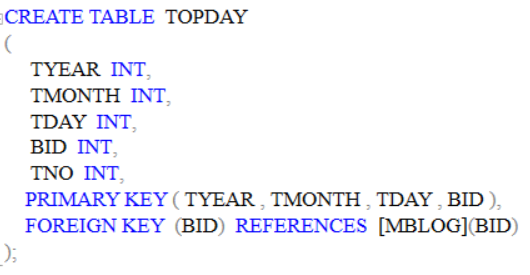


图21：创建TOPDAY表

10）观察性实验：外码是否一定要参照被参照关系的主码。

创建一个表test，其中一个外码参照的不是另外一个表的主码。如图22所示：

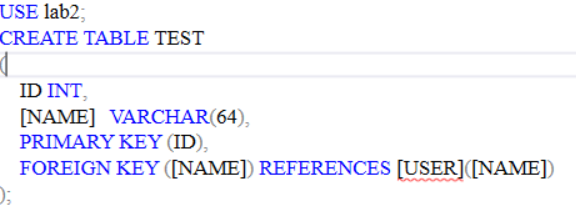


图22：参照非主码

执行时报错，报错信息如图23所示：

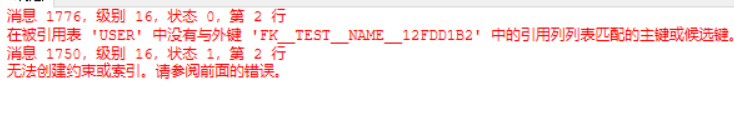


图23：参照非主码报错

由报错信息可知：报错的原因是，TEST表中的外码NAME所参照的USER表的NAME在USER表中不是主码或者候选码。

所以可以知道：外码必须要参照主码或者候选码才可以。

11）数据准备

这里使用实验时老师提供的数据，直接将其导入到所建好的表中。具体操作步骤如下：

选中数据库，右击选择任务，再选择导入数据，选择导入数据源你和目的，即可将给出的数据导入到数据库。

### 3.2.2 数据更新

1）

A 插入博文

之前给出的博文数据中包含有43条博文，BID依次从1到43。这里连续插入两条博文记录，需要用到INSERT INTO … VALUES …的语句，插入博文的SQL语句如图24所示：

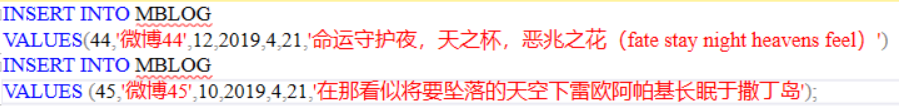


图24：插入博文

插入博文之后MBLOG表中的数据如图25所示：



图25：插入博文之后MBLOG表的数据

B 删除博文

删除博文需要用到DELETE FROM …WHERE …语句来删除符合条件的记录。

删除博文的SQL语句如图26所示：



图26：删除博文

执行完这条删除语句之后MBLOG中的数据如图27所示：



图27：执行删除语句之后MBLOG中的数据

C 修改博文

修改博文需要用到UPDATE…SET … WHERE …语句对满足条件的记录进行修改，修改博文的SQL语句如图28所示：

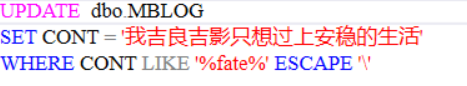


图28：修改博文

dbo.MBLOG也可以直接用MBLOG代替。修改之后MBLOG表中的数据如图29所示：

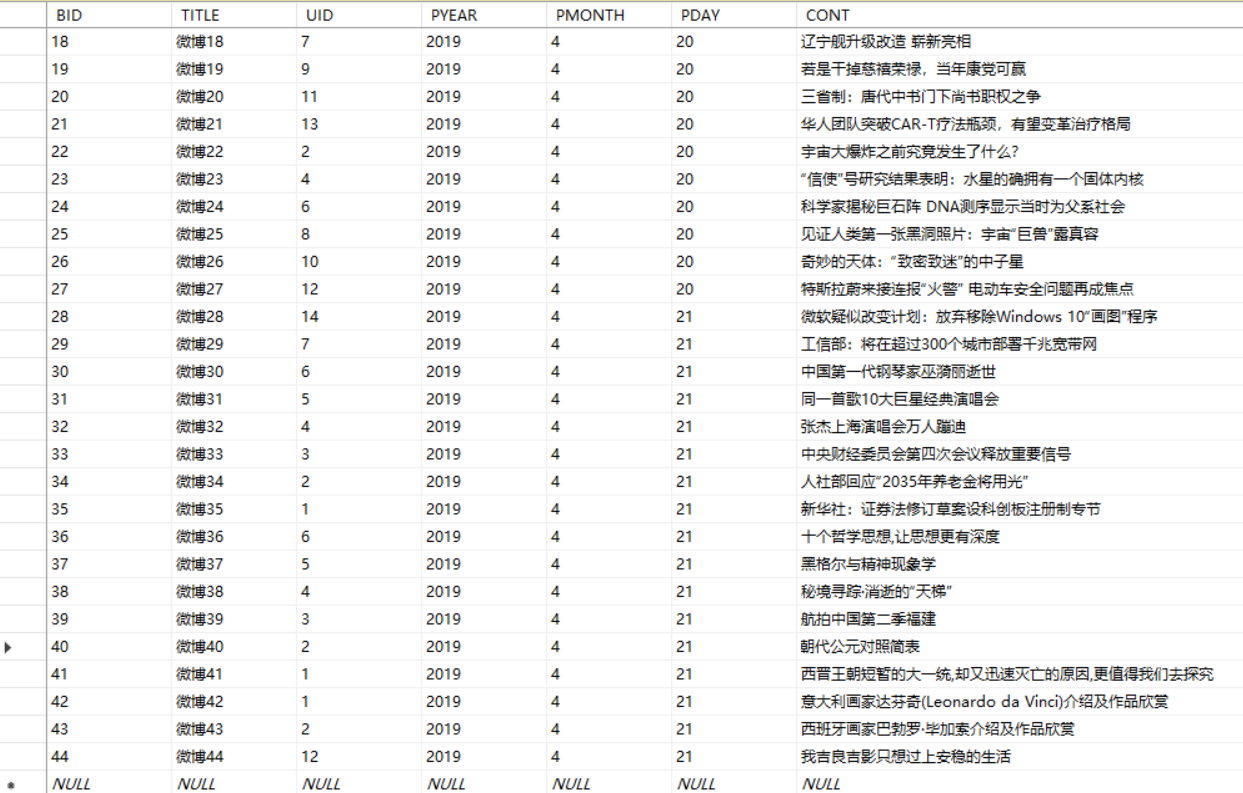


图29：修改之后MBLOG表中的数据

2）批处理操作

需要将关注3号用户的信息插入到一个新表FAN\_3中，所以应该先建立一个叫FAN\_3的新表，表的形式和USER表的形式一样。可以从FOLLOW表中查找到所有关注3号用户的UID，然后再到USER表格中将这些UID对应的记录查询出来，并将查询的结果插入到FAN\_3表中，需要用到插入一个子查询的SQL语句。完成整个功能的SQL语句如图30所示：

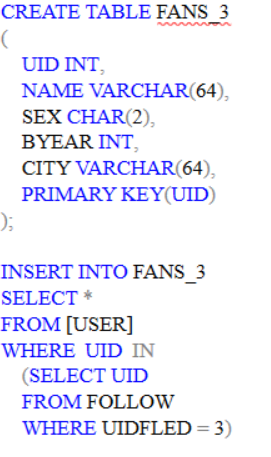


图30：批处理语句

3）数据导入导出

建立一个数据库，命名为lab2backup，来完成数据的导出和导入功能。

A 数据导出

以导出USER表到对应的user\_backup.txt文件为例：

在需要导出数据的数据库上右击，选择任务 -> 导出数据。选择数据源为SQL Server Native Client，下一部选择目标为平面文件目标，文件路径选择对应的user\_backup.txt文件的路径，继续选择下一步，然后选择需要导出的源表或源视图，这里选择USER表，然后点击下一步->完成即可。导出的所有文件如图31所示：

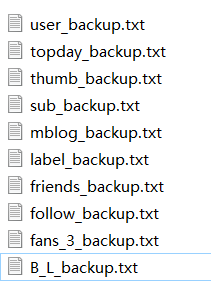


图31：导出的数据文件

B数据导入

在需要导入数据的数据库上点击右键，选择任务->导入数据，数据源选择“平面文件源”，文件路径选择需要导入的user\_backup.txt的路径，根据文件的内容决定是否带分隔符和是否勾选“在第一个数据行中显示列名”，如果勾选则导入时会将第一行视为列名。选择下一步，可以看到导入的效果，下一步，选择导入的目标为SQL Server Native Client，选择需要导入的数据库，下一步选择导入数据到哪一个表，最后点击下一步->完成即可。

4）观察性实验

A建立一个关系，不设置主码，然后向这个关系中插入重复元组。SQL语句如图32所示：

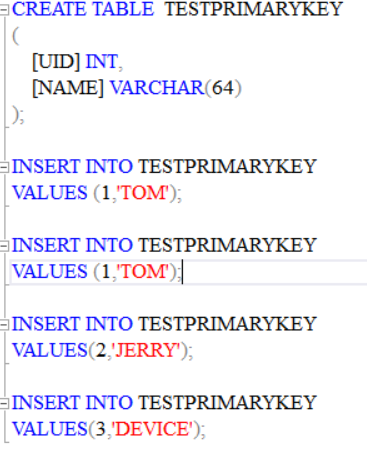


图32：建表但不设置主码

上面的SQL语句建立了一个叫TESTPRIMARYKEY的表，该表没有设置主码。然后向这个表里面插入了两个重复的元组。执行该语句之后可以看到结果如图33所示：

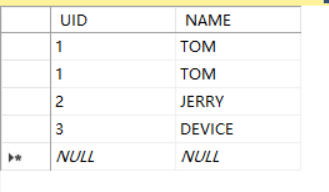


图33：插入重复元组的结果

B修改重复元组：用下面的语句修改TESTPRIMARYKEY中的重复元组，如图34所示：

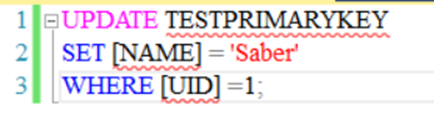


图34：修改TESTPRIMARYKEY表

执行之后的结果如图35所示：

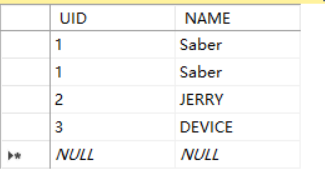


图35：执行修改之后的结果

可见，两个重复的元组同时被修改了。

C删除重复元组：使用下面的语句来删除重复的元素，如图36所示：

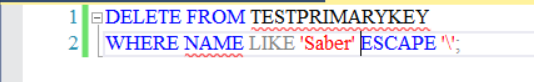


图36：删除重复元组

执行删除操作之后的结果如图37所示：

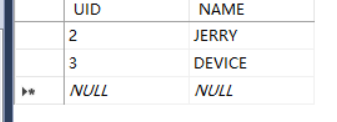


图37：执行删除的结果

可见：两个重复的元组都被删除了。

5）触发器实验

编写一个触发器，用于实现对点赞表的完整性控制规则：当插入或者更新点赞关系时，如果博文作者就是点赞者本人，则拒绝执行。

课堂上讲解时触发器有BEFORE和AFTER两种，分别在相应的操作执行前触发和操作执行后触发，但是在SQL Server中并没有BEFORE类型的触发器，想要实现BEFORE的功能，可以使用回滚函数rollback函数，回滚到执行操作之前的状态。

对一个表进行INSERT和UPDATE操作，插入或者被更新的元组会被记录到一个叫inserted的表中。所以实现所要求的触发器功能的SQL语句如图38所示：



图38：触发器实现

### 3.2.3 查询

1）首先在USER表中找出张三的UID，然后在FOLLOW表中中找出所有该UID对应的UIDFLED，即得到张三关注的所有人，然后再到USER表中找出这些ID对应的基本信息。

对应的SQL语句如图39所示：

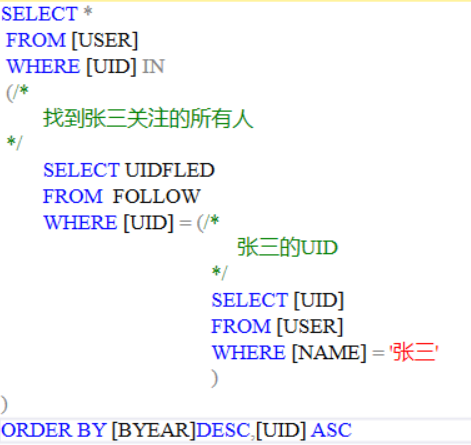


图39：查询1的实现

查询执行的结果如图40所示：



图40：查询一的结果

2）可以从THUMB表中投影出所有被点赞的博文的BID，然后逐条检查博文表MBLOG中的每一条博文，如果某条博文的BID不包含在所有被点赞的博文的BID中，则将此条博文的BID和TITILE以及作者的NAME选出来。并把结果按照TITLE的顺序排序

对应的SQL语句如图41所示：



图41：查询2的实现

执行查询2的结果如图42所示：



图42：查询2的执行结果

3）首先可以从表USER中选择BYEAR>=2000并且CITY = "武汉"的UID，得到所有2000年以后出生的武汉用户，记为集合A，然后在表TOPDAY中投影出所有的BID，即进入过头条的所有的BID，记为集合B。然后逐个检查表MBLOG中的元组，如果某个元组对应的UID在集合A中并且BID在集合B中，则将其挑选出来。

对应的SQL语句如图43所示：



图43：查询3的实现

执行查询3的结果如图44所示：

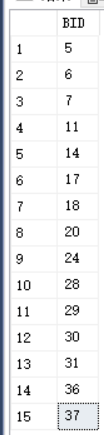


图44：查询3的执行结果

4）订阅了所有分类，即没有一个分类是没有订阅的，即选择一个用户U，依次遍历每一次分类L，一定可以在订阅表中找到一个元组（U，L）

对应的SQL语句如图45所示：

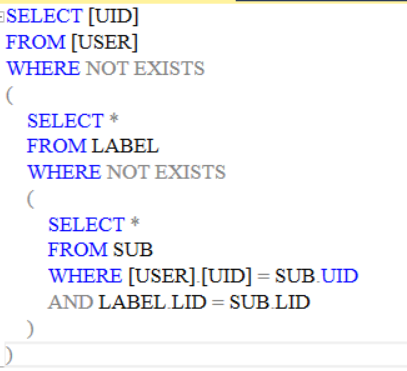


图45：查询4的实现

查询4的执行结果为空

5）用NOT BETWEEN …AND即可满足只用一个条件表达式的要求，

对应的SQL语句如图46所示：

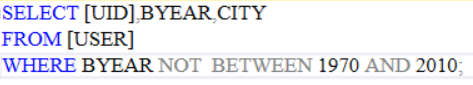


图46：查询5的实现

查询5的执行结果如图47所示：

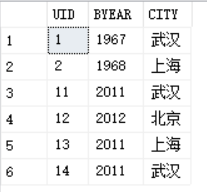


图47：查询5的执行结果

6）将用户表按照城市CITY分组，然后每个组内统计元组个数即可得到每一个城市的用户数。

对应的SQL语句如图48所示：

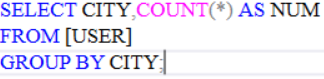


图48：查询6的实现

查询6的执行结果如图49所示：



图49：查询6的执行结果

7）将用户表USER按照CITY和BYEAR分组，统计每个组的元素个数即得到每个城市每个年份出生的人数，然后将结果城市的升序排列，城市相同按照用户数降序排列。

对应的SQL语句如图50所示：

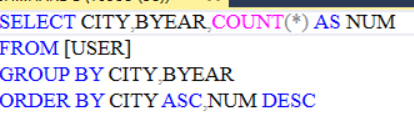


图50：查询7实现

查询7的执行结果如图51所示：



图51：查询7的运行结果

8）将点赞表THUMB按照BID分组，然后每一组内统计元组个数，元组个数等于点赞数，只需要将元组个数大于1的的组的BID选出来即可。

对应的SQL语句如图52所示：

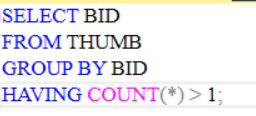


图52：查询8的实现

查询8的执行结果如图53所示：

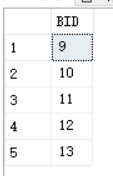


图52：查询8的执行结果

9）先将所有2000年后的用户的点赞记录都选择出来，然后再按照BID分组，统计每一组的元组个数，将元组个数大于1 的BID选择出来。

对应的SQL语句如图53所示：

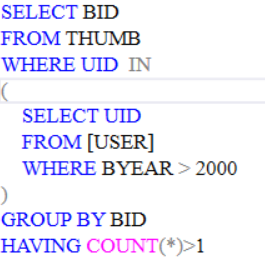


图53：查询9的实现

查询9的执行结果如图54所示：

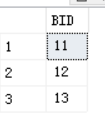


图54:查询9的执行结果

10）所需要实现的功能和查询9差不多，先按照查询9得到所有2000后点赞数超过1的BID，记为集合A，然后将TOPDAY按照BID分组，统计每一组内元组个数，然后将BID在集合A的组选择出来即可。

对应的SQL语句如图55所示：



图55：查询10的实现

查询10的执行结果如图56所示：

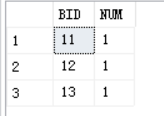


图56：查询10的执行结果

11）只需要将订阅表SUB和分类表LABEL在属性列LID上做等值连接，再将LABEL.LNAME是“文学”“艺术”“哲学”“音乐”，其中之一的元组的UID跳出来即可。

对应的SQL语句如图57所示：

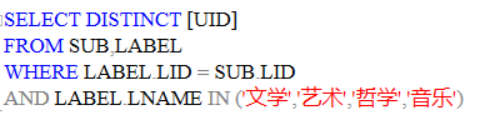


图57：查询11的实现

查询11的运行结果如图58所示：

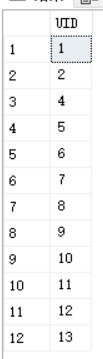


图58：查询11的运行结果

12）直接用字符串查询即可，检查标题中是否含有所要求的字符串

对应的SQL语句如图59所示：



图59：查询12的实现

查询12 的执行结果为空

13）将两个FOLLOW表A和B做连接，连接条件是A中元组a和B中元组b满足：a.UID = b.UIDFLED 并且b.UID=a.UIDFLED 。

对应的SQL语句如图60所示：

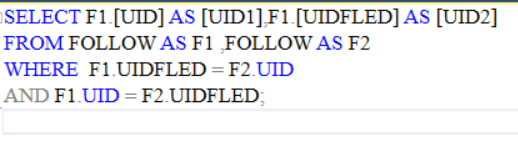


图60：查询13的实现

查询13的执行结果如图61所示：

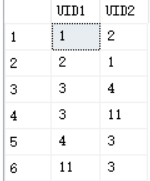


图61：查询13的执行结果

该结果对于（1，2）（2，1）这样的组并没有实现去重，实际上这两个元组表示的意义是一样的

14）这一题和查询4类似，查询朋友圈包含了5号所有好友的用户。即没有任何一个五号的好友不是该用户的好友，即从USER中选择一个元组A，从五号的好友集中逐个选出元组B，(A.UID,B.UID)一定存在于FRIENDS关系中。并且要去掉5号自己

对应的SQL语句如图62所示：

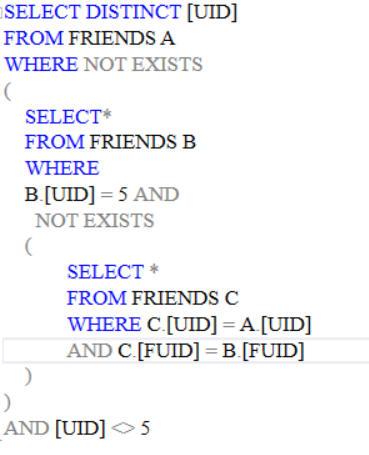


图62：查询14的实现

查询14的结果为空

15）使用外连接，来保存某个表中在连接过程中被舍弃的元组。

对应的SQL语句如图63所示：



图63：查询15的实现

查询15的执行结果如图64所示：

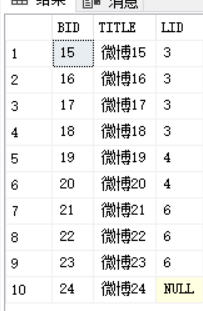


图64：查询15的执行结果

16）先将两个FRIENDS表A和B连接，连接条件是两个元组的FUID相等但是要UID不相等，这样就可以得到一个包含（USER1 USER2 COMMON\_FUID）信息的表格，然后按照USER1，USER2来分组，统计每一组内元组个数即得到任意有共同好友的两人的共同好友数量。再将共同好友数量大于等于1的找出来即可

对应的SQL语句如图65所示：

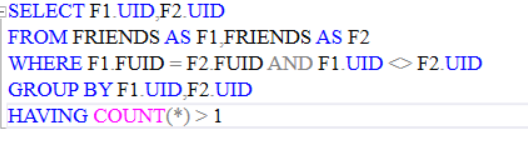


图65：查询16的实现

查询16的执行结果如图66所示：

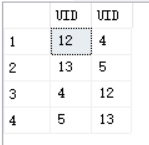


图66：查询16的执行结果

该结果没有实现对（4，12）（12，4）这样的元组的去重，实际上这两个元组的意义是一样的。

17）先从点赞表THUMB中选择出点赞数排名前10的十条微博的BID，记为集合A，然后于博客表MBLOG 以及用户表USER做连接，连接条件是MBLOG.BID = A.BID MBLOG.UID = USER.UID,再将需要的信息给投影出来。

创建视图的SQL语句如图67所示：

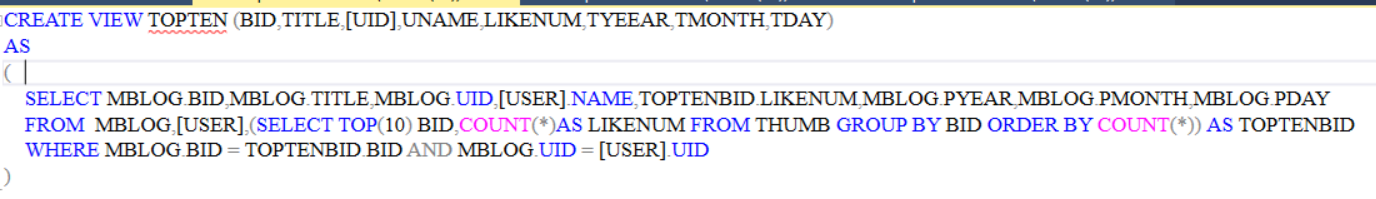


图67：创建视图

执行的结果如图68所示：



图68：视图的结果

## .3任务总结

(1)利用SQL语言创建了USER。MBLOG等表。并通过图形化的功能往表中导入了数据。

(2)验证了关系的外码一定要参照另外一个关系的主码。

(3)使用SQL语句对表的内容进行了基本的增删改，并通过图形化界面直观的看到这些操作的效果。

(4)实现了批量插入数据的操作。

(5)体验了主码在一个关系中的重要性，测试了没有定义主码的关系，在增删改时的执行情况。

(6)通过定义触发器，来加深对触发器工作机制的理解。

(7)通过实现多种不同的查询的需求，运用了多种查询表达式，加深对查询表达式的熟悉和理解，

(8)学习了创建视图。