**数据库系统上机要求**

**数据库系统共5次上机，上机地点为G330,334，具体上机时间为：**

**1、2019.3.27，周三晚上**

**2、2019.4.3，周三晚上**

**3、2019.4.10，周三晚上**

**4、2019.5.15，周三晚上**

**5、2019.5.29，周三晚上**

**上机1**

1. **教材Banking数据库的建库、数据更新及查询。**
2. **课后习题中的数据库建库、数据更新（可参照Banking中的数据进行构造 ）及查询**

**上机1具体要求和步骤**

1. **安装Mysql；**

注意：设置超级用户密码的时候，统一设置为1234

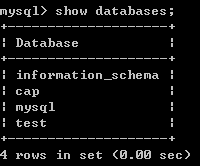
1. **进入Mysql；**

开始->程序->MySQL->MySQL Server 5.0->MySQL Command Line Client

1. **建立数据库BANKxxx；**

create database BANKxxx; --xxx为你的学号后三位

show databases; --验证是否建立成功



1. **使用数据库**BANK**；**

use BANKxxx; --使用数据库BANK

show tables; --数据库中没有表

1. **建立表BRANCHxxx, CUSTOMERxxx, LOANxxx, BORROWERxxx, ACCOUNTxxx, DEPOSITORxxx；**

CREATE TABLE ACCOUNTxxx (

*account\_number* char(10),

*branch\_name* char(30) NOT NULL,

*balance* numeric(12.2),

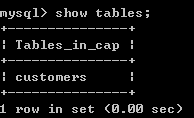
PRIMARY KEY (*account\_number*),

FOREIGN KEY (*branch\_name*)   
 REFERENCES *branch*(*branch\_name*),

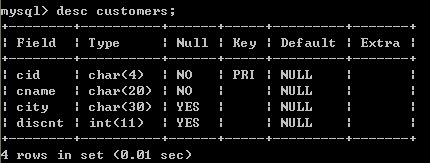
CONSTRAINT *chk\_balance* CHECK (*balance* >= 0 ) ) ;

--建立customersxxx表，xxx同前

show tables; --现在数据库中有了表



desc customers; --查看表的属性

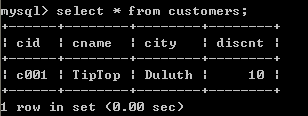


1. **将书上第2章图2-1，2-3，2-4，2-5，2-6，2-7中的数据输入到表中；**

insert into ACCOUNTxxx

values (‘A-101’, ‘Downtown’, 500); --向表中插入数据

select \* from ACCOUNTxxx; --验证是否插入数据成功



1. **创建图3.12的四张表，输入一些数据，查询书上第三章3.2, 3.9的作业题**
2. **上机报告，将实现的源码、截图做成word文档，以学号作为文档的名称，由课代表收齐后统一上交。**

**上机2**

1. **基于第一次上机创建的银行数据库，创建一个视图branch\_detail，能够显示所有支行的存款客户数量、存款总额、贷款客户数量、贷款总额。**
2. **在account的account\_number属性上建立索引，并在account表里插入大量元组，比较有无索引在查询速度上的区别。**
3. **创建角色银行经理branch\_manager，银行经理具有插入、删除和更新branch表的权限。**
4. **自由练习第四章中级SQL的其他特性。**

**上机3**

1. **基于第一次上机创建的银行数据库，创建一个函数，为所有存款账户增加1%的利息。**
2. **创建一个新表branch\_total，用于存储各个支行的存款总额（表中有branch\_name和total\_balance两个属性）。然后在这个表上，创建一个触发器，实现当有用户存款变动（包括增加、删除和更新）时，brach\_total表中的存款总额跟着自动更新。**
3. **自由练习第五章高级SQL的其他特性。**

**上机实验4**

**实验要求：**

基于MySQL，设计并实现一个简单的旅行预订系统。该系统涉及的信息有航班、大巴班车、宾馆房间和客户数据等信息。其关系模式如下：

FLIGHTS (String flightNum, int price, int numSeats, int numAvail, String FromCity, String ArivCity)；

HOTELS(String location, int price, int numRooms, int numAvail)；

BUS(String location, int price, int numBus, int numAvail)；

CUSTOMERS(String custName,custID)；

RESERVATIONS(String custName, int resvType, String resvKey)

为简单起见，对所实现的应用系统作下列假设：

1. 在给定的一个班机上，所有的座位价格也一样；flightNum是表FLIGHTS的一个主码（primary key）。
2. 在同一个地方的所有的宾馆房间价格也一样；location是表HOTELS的一个主码。
3. 在同一个地方的所有大巴车价格一样；location是表 BUS的一个主码。
4. custName是表CUSTOMERS的一个主码。
5. 表RESERVATIONS包含着那些和客户预订的航班、大巴车或宾馆房间相应的条目，具体的说，resvType指出预订的类型（1为预订航班，2为预订宾馆房间，3为预订大巴车），而resvKey是表RESERVATIONS的一个主码。
6. 在表FLIGHTS中，numAvail表示指定航班上的还可以被预订的座位数。对于一个给定的航班（flightNum）,数据库一致性的条件之一是，表RESERVATIONS中所有预订该航班的条目数加上该航班的剩余座位数必须等于该航班上总的座位数。这个条件对于表BUS和表HOTELS同样适用。

**应用系统应完成如下基本功能：**

* 1. 航班，大巴车，宾馆房间和客户基础数据的入库，更新（表中的属性也可以根据你的需要添加）。
  2. 预定航班，大巴车，宾馆房间。
  3. 查询航班，大巴车，宾馆房间，客户和预订信息。
  4. 查询某个客户的旅行线路。
  5. 检查预定线路的完整性。
  6. 其他任意你愿意加上的功能。

**作业检查：**

1. 提交源程序，可执行程序，以及程序运行说明。
2. 系统分析、设计与实现报告。
3. 考试前检查完毕，延迟拒收。

**上机实验5：设计学籍管理系统**

**学籍数据库的有关语义如下：**

* 1. 一个系可以若干专业，每个专业每年可以招若干班，每个班有若干学生。
  2. 系里对每个专业每年都制订了教学计划，学生每年必须按照教学计划修完一定学分的课程（必修课、限选课和任选课），如2000年入学的学生大三上学期必修课30学分，限选课10学分，任选课6学分。
  3. 系里的教师可以给多个班带课，但是不能给一个班带多门课程。
  4. 一门课程最多允许学生一次补考，学生达到如下条件之一的被开除：
     1. 一学期不及格的必修课学分超过10个；
     2. 不及格必修课学分累计超过30个；
     3. 不及格选修课学分累计超过20个；

**处理要求：**

1. 查询学生所选修的课程及成绩，并给出必修课平均成绩和选修课平均成绩；
2. 查某一个学生被哪些教师教过课；
3. 查询应被开除的学生（假定差2学分即被开除）。

**注意事项：**

* 1. 在数据库的设计过程中需要运用规范化理论，避免出现插入异常、删除异常、数据冗余等问题；
  2. 必须设定关系的完整性规则，如实体完整性（例如设置主码），参照完整性（例如设置外码和对应的主码），用户自定义完整性（例如性别只能为“男”或“女”）；
  3. 可以使用索引来加快查询的速度；
  4. 可以使用视图来简化系统的设计；

上机作业报告最好使用电子文稿，但严禁拷贝(**对于内容相近的报告，实验成绩按不及格处理**)。

按照数据库设计的基本步骤，书写上机报告：

* 1. 需求分析（系统数据和功能）
  2. 概念结构设计（E-R图设计）
  3. 逻辑结构设计（E-R图转换为关系模型）
  4. 应用程序设计中遇到的问题及解决方法
  5. 附建立数据库和查询的主要代码（源代码）