**数据库基础**

**前期理论知识：**

数据库系统简称Data Base System,DBC

sa是系统管理员

数据库的组成：

数据库（数据）数据库管理系统（软件）、数据库管理人员（人员）、硬件平台（硬件）、和软件平台（软件）

常见的数据模型：

1、层次模型2、网状模型（视屏中说道这个比较复杂，不推荐使用）3、关系模型

目前地主流数据模型为关系模型

术语：一个关系就是一个二维表

原主，表示一行，就是一条记录

属性，表二维表中的一列，主要是使用类和值来表示

域，表示的是每个值的取值范围，比如说性别只要两个：男、女

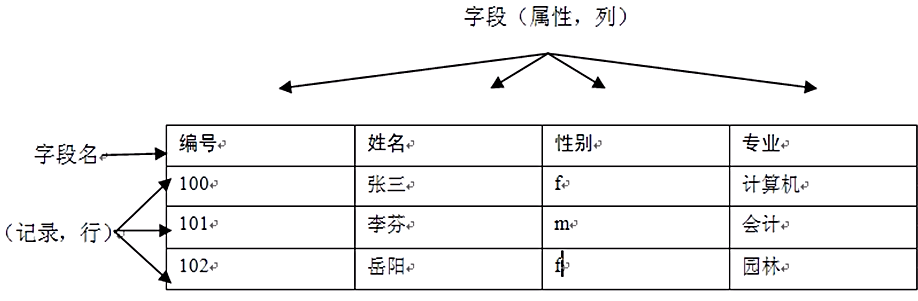
关系数据库的规范化：

就是关系数据库中的每一个关系都应该满足一定的规范，根据满足规范的不同来分成了几个等级

1. 、第一范式（1NF）

它是最基础的范式

1. 、第二范式（2NF）
2. 、第三范式（3NF）

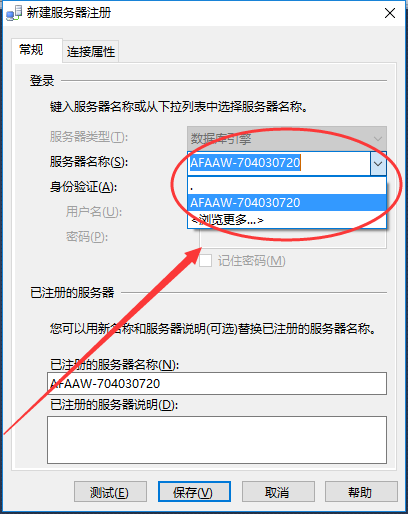




Analysis Service分析服务 reporting Service报表服务 Integration Service整合服务

**1、服务器的注册与删除**

这里值得注意的是



这里值得注意的是“身份验证”框里面的名称不是随意打的，而是通过下拉列表来选择的。

**2、数据库的组成：**

文件：

主要数据文件：存放数据和数据库务的初始化信息。每个数据库有且只有一个主要数据文件，默认扩展名为.mdf

次要数据文件：存放主要数据文件以外的所有数据文件。有些数据库可能没次要数据文件，也可能有多个要数据文件。默认扩展名为.ndf

事物日志文件：存放用于恢复数据库的所有日志信息。每个数据库至少有一个事物日志问价文件，也可以有多个事物日志文件，默认扩展名是.ldf。

**3、标识符的格式**

在SQL2012中，某些处于标识符开始位置的符号具有特殊意义。

以@开始的标识符表示局部变量或参数；

以#开始的标识符表示临时表或过程，如#gzb就是一张临时表

以双##开始的标识符表示全局临时对象，如##gzb就是全局临时表

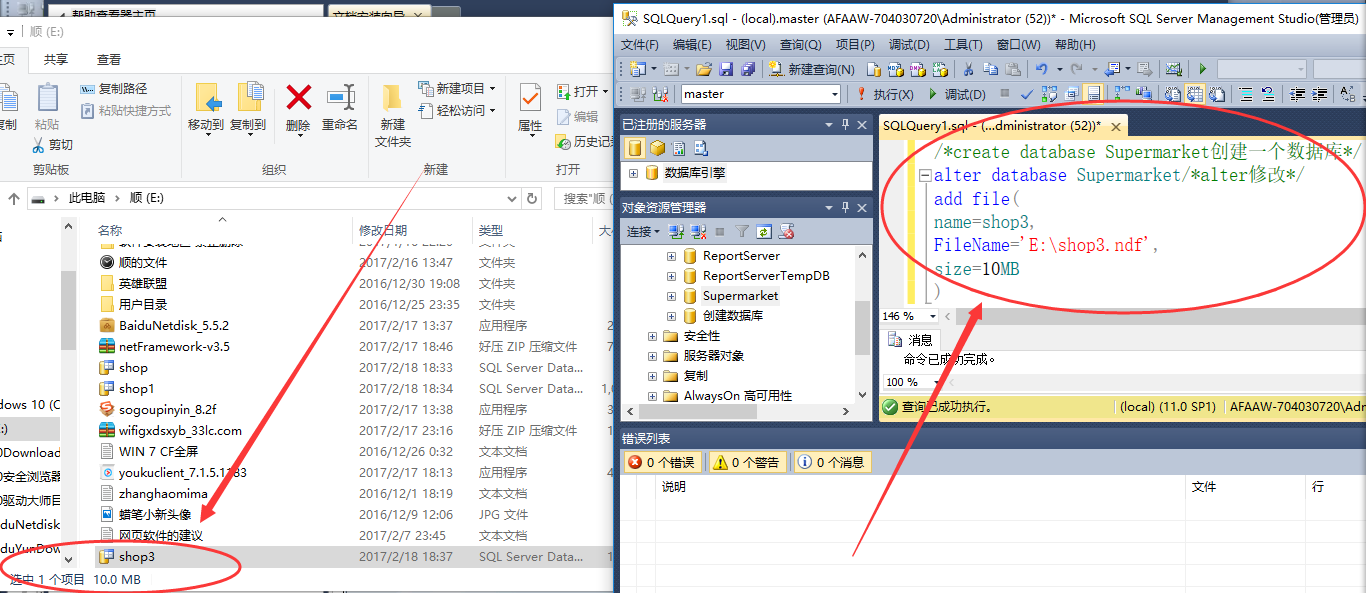
对象名的命名规则：

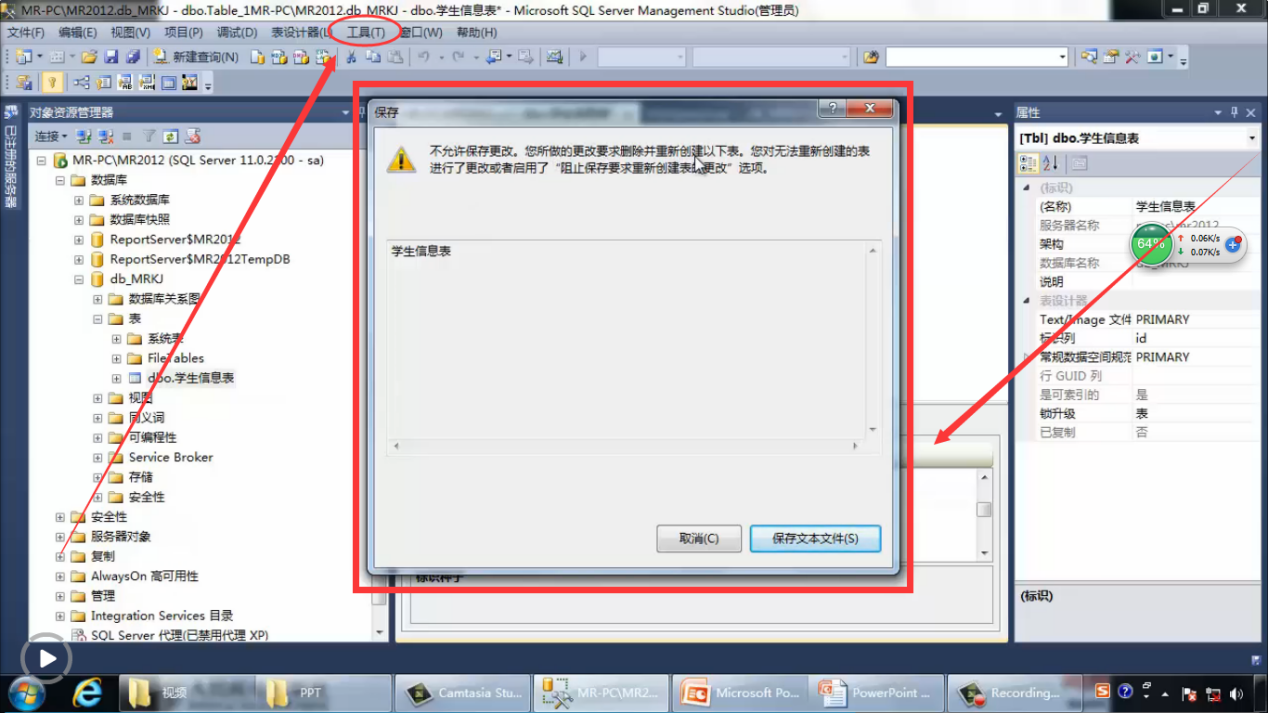
完整的数据库对象名应该由服务器名、数据库名、拥有者名和对象名4部分组成。

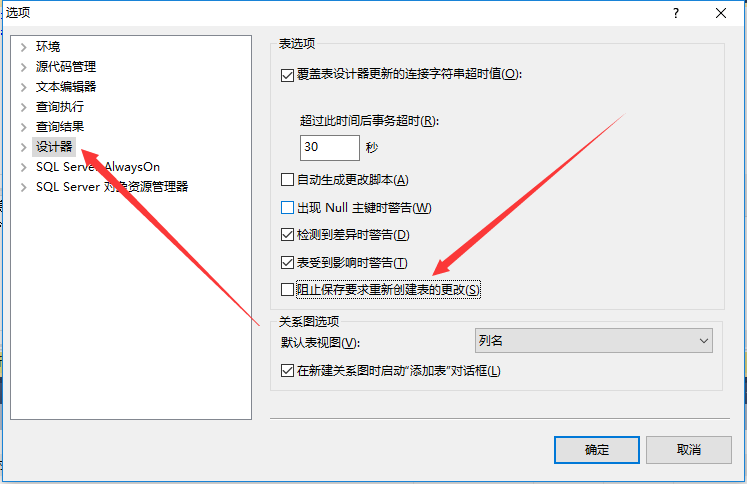
可以是下图的简写格式



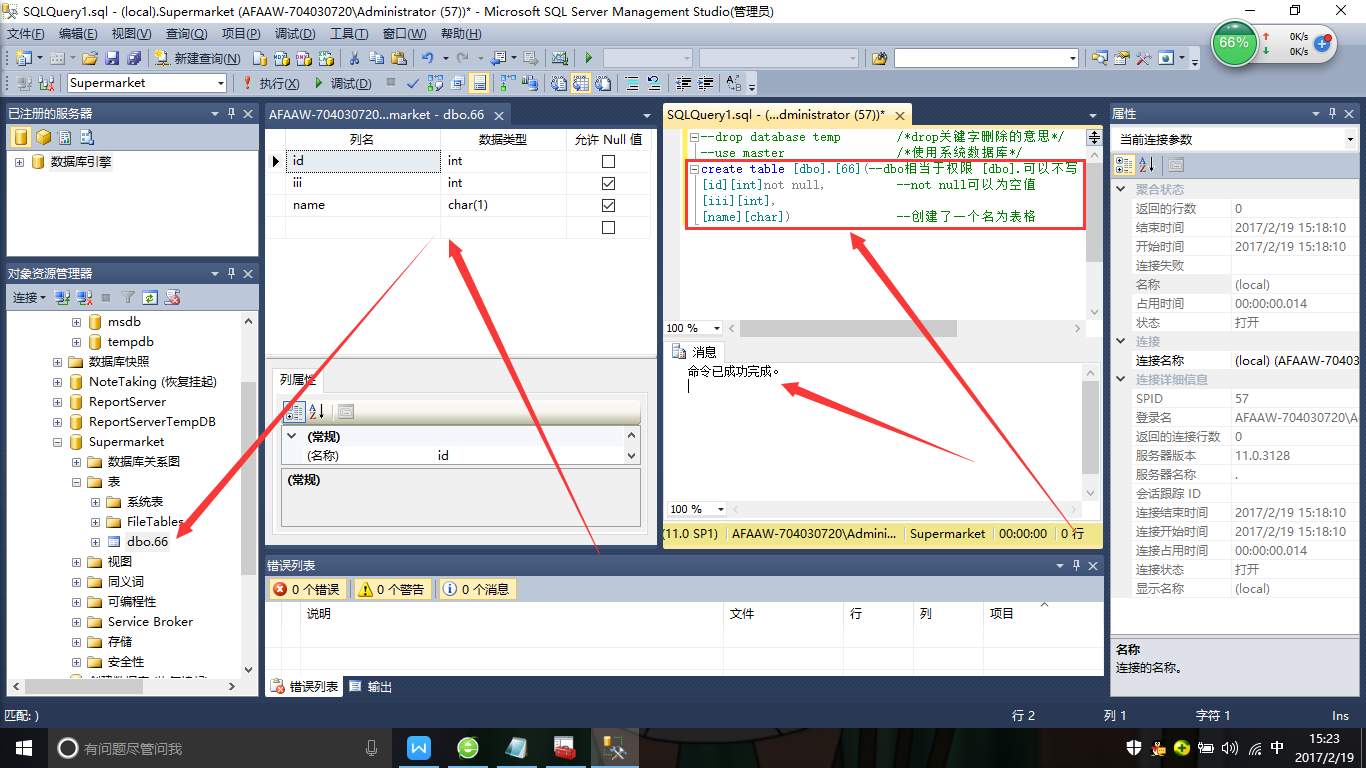
用代码修改

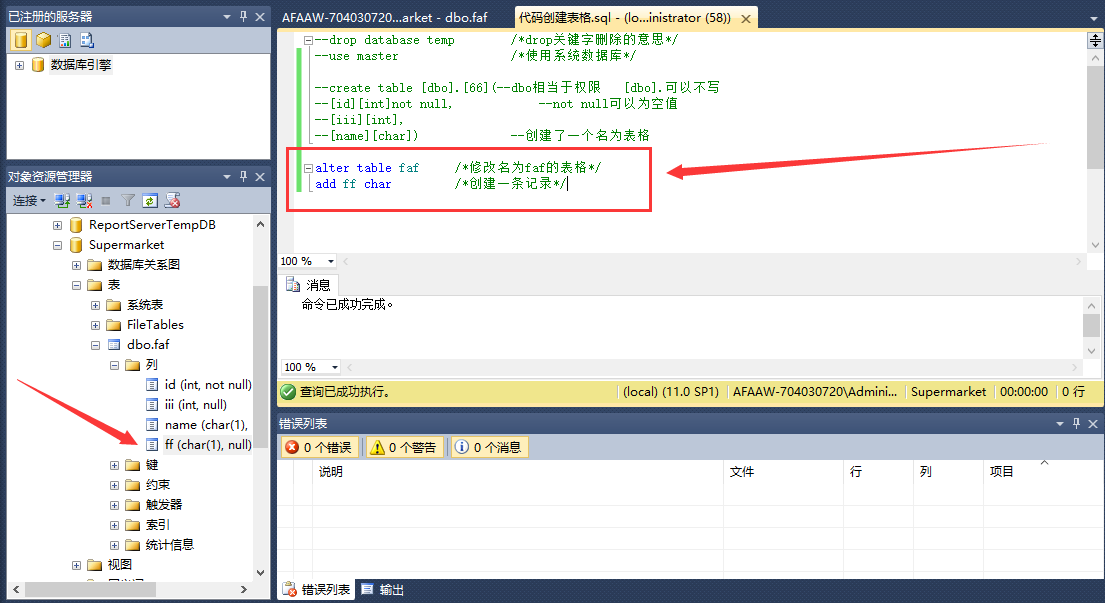






**4、代码创建修改表格**





create table kcb(

课程号 char(3) not null primary key, --设置主键，该字段不允许为空

课程名 char(16) not null,

开课学期 tinyint default 1, --默认为1

学时 tinyint default 0, --tiny：微小

学分 tinyint not null default 0

)

create table cjb

(

学号 char(6) not null,

课程号 char(3) not null,

成绩 int default 0,

primary key(学号,课程号) --设置两个主键

)

create table pjcj

(

课程号 char(3) primary key,

总成绩 real not null,

人数 int not null,

平均成绩 AS 总成绩/人数 persisted --persisted坚持，永远

)

alter table XSB --修改表格

add 奖学金等级 tinyint null, --添加表格字段

add 奖学金等级1 tinyint null,

add 奖学金等级2 tinyint null,

alter column 姓名 char(10) --修改表格字段

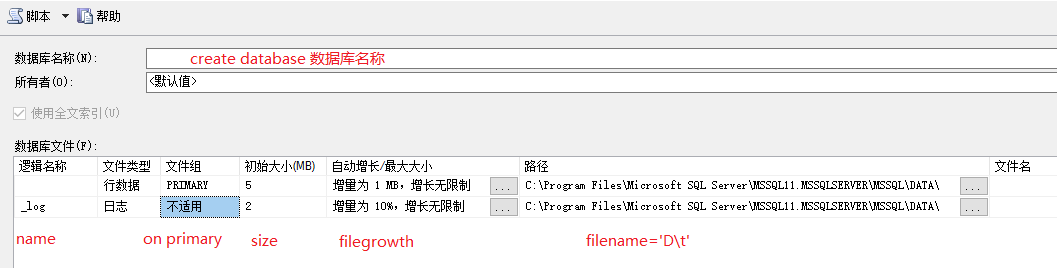
alter column 出生时间 smalldatetime

alter table xsb

drop column 奖学金等级 --删除表格字段 column 柱、列

**5、代码修改创建数据库**





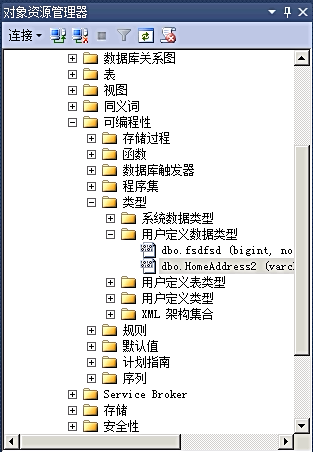
Alter database Mydata

Modify file modify = 修改

(

Name = Mydata, size = 30, maxsize = 50)

**5、用户自定义类型**



Sp\_addtype Mytype,'varchar(128)','not null' --新建数据类型

sp\_droptype Mytype --删除自定义数据类型

**6、select查询语句**

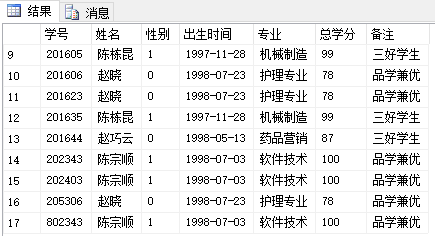
**Select 列名称 from 表名称**

**Select 列名称 from 表名称 where 字段 = ‘ ’ and 字段 = ‘ ’ and 字段 = ‘ ’**

select 姓名,专业 from xsb



select \* from xsb --查询数据表



select 学号, 姓名,总学分, 专业 from xsb where 总学分= '100'

学号、姓名、总学分、专业 从 xsb 中查 并且总学分 = 100的人



**星号（\*）是选取所有列的快捷方式**

select \* from xsb where 专业='计算机' --查到专业为计算机的

select 姓名 from xsb -- 查询xsb中姓名字段的

select \* from xsb where 姓名='王林' -- \* 代表着字段，列

select \* from xsb where 出生时间>'1901-12-17' --查询出生时间大于\*\*日的

select \* from xsb where 学号='081101' --查询出生时间大于\*\*日的

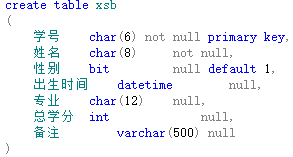
select \* from xsb where 姓名 like '王%' --找到所有姓王的人

select \* from xsb where left(姓名,1)='王' --找到所有姓王的人，调用函数

select \* from xsb where 姓名 like '%王%' --找到字符串中含有王字的人

**7、insert插入**

字体

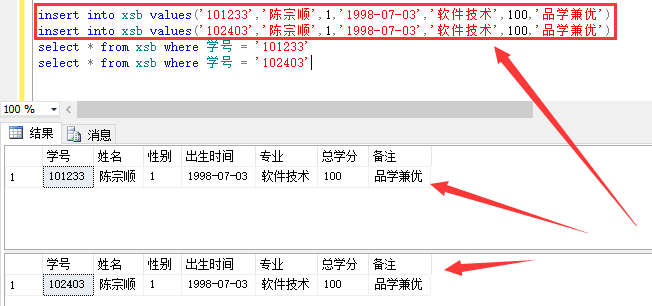


insert into xsb values('201603','陈宗顺',1,'1998-07-03','软件技术',93,'品学兼优')

insert into xsb values('201604','陈宗顺',1,'1998-07-03','软件技术',95,'三好学生')

记住一定要配对好它们的数据类型

insert into xsb values --查询进入之内



**8、delete删除记录**

字体

delete xsb --**删除整个表格里面的记录**

delete xsb where 学号='201611' --删除学生表中学号为的行

**9、update**

update xsb set 总学分=总学分+10 --所有记录的学分都加分

update xsb set 专业='软件工程',学号=081261 where 姓名 = '王林看'

修改 表 设置 \* = \*\*\*

**10、代码设置主外键primar foreign**

注意了，这里

use pxscj --当前表为外键表

alter table cjb

add constraint fk\_cjb\_xsb foreign key(学号)references xsb(学号)

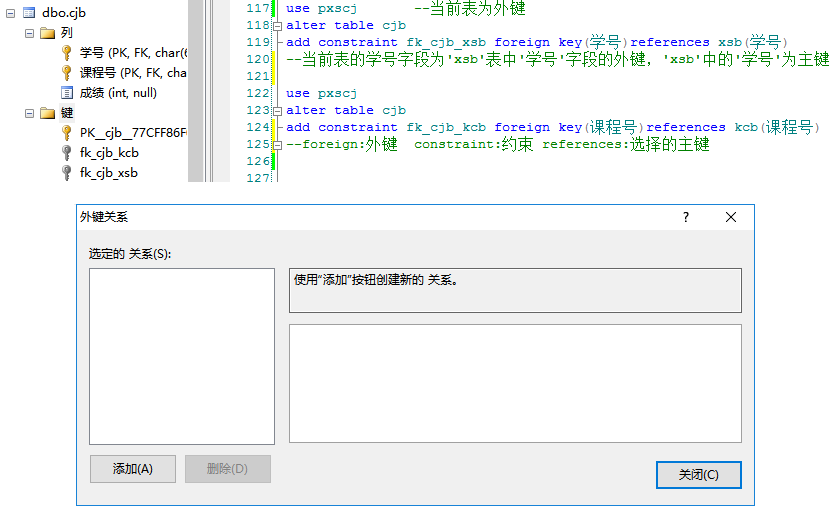
--**当前表的学号字段为'xsb'表中'学号'字段的外键，'xsb'中的'学号'为主键**

use pxscj

alter table cjb

add constraint fk\_cjb\_kcb foreign key(课程号)references kcb(课程号)

--foreign:外键 constraint:约束 references:选择的主键



说明了代码间的联系，工具栏中的关系列表中是没有的。

值得注意的是，建立了联系之后，外键表的主键里的内容一定要在主键表里面有，也就是说主键表有这个东西，外键表里面才能够有，外检表不能无中生有。

**11、唯一键unique**

use pxscj

create table xsb1 --创建代码时就设置

(学号 char(6),

姓名 char(6),

constraint pk primary key(学号),

constraint px unique(姓名)) --设置唯一键

--表格建立完毕之后设置主键为一键

alter table xsb1

add constraint pk primary key(学号) --添加主键

alter table xsb1

add constraint px unique (姓名) --设置唯一键

唯一键，可以为null

**12、check约束**

唯一键

----------check约束----------

create table kcb2 --表格时创建check约束

(

课程号 char(6) not null,

课程名 char(8) not null,

学分 tinyint check (学分>=0 and 学分<=60) null /\*通过check子句定义约束条件\*/

)

alter table kcb2 --表格创建后添加check约束

drop constraint CK\_\_kcb2\_\_学分\_\_0EA330E9

alter table kcb2 --删除表格中的check约束

add constraint ck学分 check(学分>=0 and 学分 <=60)

create table student --创建表格的时候添加一个check约束

(

学号 char(6) not null,

性别 char(1) not null check(性别 in ('男', '女'))--只考虑“学号”和“性别”两列，性别只能包含男或女

)

alter table student

add 姓名 char(10)

alter table student --表格创建好之后添加一个约束

add constraint ck姓名 check(姓名 in ('张','李'))

--8、创建表student2，有“学号”、“最好成绩”和“平均成绩”三列，要求最好成绩必须大于平均成绩。

drop table student2

create table student2

(

学号 char(6) not null,

最好成绩 int not null,

平均成绩 int,constraint sss check (最好成绩>平均成绩)

)

create table student1

(

学号 char(6) not null,

出生时间 datetime not null,

constraint df\_student1\_cjsj check(出生时间>'1980-01-01')

)

**13、case、avg、sum、top、取列名**

--1、查询PXSCJ数据库中XSB表的所有数据。

select \* from xsb

--2、查询PXSCJ数据库的XSB表中各个同学的姓名、专业和总学分。

select 姓名,专业,总学分 from xsb

--3、查询XSB表中计算机专业同学的学号、姓名和总学分，查询XSB表中所有列。

select 学号,姓名,总学分 from xsb where 专业='计算机'

--4、查询XSB表中计算机系同学的学号、姓名和总学分，结果中各列的标题分别指定为number、name和mark。

select 学号 as number,name=姓名,总学分 as mark from xsb where 专业='计算机'--as:和、一样的 “=”和”as”取别名

--5、查询XSB表中计算机系各同学的学号、姓名和总学分，对其总学分按以下规则进行替换：

--若总学分为空值，则替换为“尚未选课”；若总学分小于50，则替换为“不及格”；

--若总学分在50与52之间，则替换为“合格”；若总学分大于52，则替换为“优秀”。列标题更改为“等级”。

select 学号 ,姓名,等级=case --case:案例、情况

when 总学分=null then '尚未选课' --when:什么时候、条件

when 总学分< 50 then '不及格' --then:然后、那么、于是

when 总学分>=50 and 总学分 <=52 then '及格'

when 总学分> 50 then '优秀'

end --其实就是编程语言中的swich开关语句

from xsb

where 专业='计算机'

--6、按120分计算成绩并显示学号为081101的学生的成绩情况。.

select 成绩=成绩\*1.2 from CJB where 学号='081101'

--7、对PXSCJ数据库的XSB表只选择专业和总学分，消除结果集中的重复行

select 专业,总学分 from xsb

select distinct 专业,总学分 from xsb --distinct:不同的 只显示不同的,排除相同的只显示一列

--8、对PXSCJ数据库的XSB表选择姓名、专业和总学分，返回结果集的前6行。

select top 6 姓名,专业,总学分 from xsb

--9、求选修101课程的学生的平均成绩。

select AVG(成绩) from CJB where 课程号=101

--10、求学号为081101的同学所学课程的总成绩。

select SUM(成绩) from CJB where 学号=081101

**14、各种查询count、between、%、\_通配符**

--0、求选修课程的学生的平均成绩、总成绩。

select 平均成绩 = AVG(成绩),SUM(成绩) as 总成绩 from CJB where 课程号='101'

--1、求选修课程的学生的最高分和最低分。

select 最高分 = max(成绩),min(成绩) as 最低分 from CJB where 课程号='101'

--2、求学生的总数

select COUNT(\*) as 总人数 from xsb

--3、统计备注不为空的学生数。

select COUNT(\*) from xsb where 备注 is null

select COUNT(\*) from xsb where 备注 is not null

select \* from xsb where 备注 is not null

--4、统计总学分在分以上的人数。

select \* from xsb where 总学分>50

--5、求选修了课程的学生总数。网络作为选修

select \* from xsb

**select COUNT(专业) as 选了网络选修课的 from xsb where 专业='网络'**

--6、复制新表xsb1,其结构与xsb一样。

select \* into xsb1 from xsb

select \* from xsb1

--7、查询XSB表中计算机专业总学分大于等于的同学的情况。

select \* from xsb where 专业='计算机' and 总学分>42

--8、查询XSB表中姓“王”且单名的学生情况。（刘姓和李姓）

**select \* from xsb where 姓名 like ('王%') --姓王的**

**select \* from xsb where 姓名 like ('%王%') --字符串中只要有王的**

**select \* from xsb where 姓名 like '王%' and 姓名 like '李%' --又姓王又姓李的不存在**

**select \* from xsb where 姓名 like '王%' or 姓名 like '李%' --注意不是and而是or**

--9、查询XSB表中学号倒数第个数字为，且倒数第个数在～之间的学生学号、姓名及专业。

**select \* from xsb where 学号 like '%1\_[1-5]' --通配符‘% ’代表任意多个或单个字符**

**select 学号,姓名,专业 from xsb where 学号 like '%1\_[12345]' --‘\_ ’通配符，代表任意单个字符**

--10、查询XSB表中不在年出生的学生情况。

**select \* from xsb where 出生时间 between '1989-01-01' and '1989-12-31' --between:在…之间**

--11、查询XSB表中专业为“计算机”、“通信工程”或“无线电”的学生情况。

**select \* from xsb where 专业 = '计算机' or 专业 = '通信工程' or 专业 = '无线电'**

**select \* from xsb where 专业 in ('计算机','通信工程','无线电') --in:在…之内**

--12、查询备注尚不定的学生情况。

select \* from xsb

select \* from xsb where 备注 is null

select \* from xsb where 备注 is not null

1. **distinct、order by desc,asc升降、年龄**

--1、在xsb中查找年龄在26~27岁的所有男性记录的学号、姓名、年龄、性别。并按年龄进行降序排列。

**select 学号,姓名,YEAR(GETDATE()) - YEAR(出生时间) 年龄 ,性别 = case--是函数就必须要打括号**

**when 性别='1' then '男'**

**else '女' end**

**from xsb**

**where year(getdate()) - YEAR(出生时间) between 26 and 27 --between配对and配合使用**

**order by YEAR(GETDATE()) - YEAR(出生时间) desc --order by 排序 desc 降序 默认asc顺序**

--2、查询所有女性记录的学号、姓名、专业、出生日期，并按专业升序，出生日期降序排列记录。

select 姓名,专业,出生时间,专业,性别 = case

when 性别='0' then '女'

else '男' end

from xsb

where 性别='0'

order by 出生时间 desc

--3、在学生表中以专业升序排列查询前三个学生的姓名以及与第三个学生同专业的学生的姓名。

select top 3 **with ties** \* from xsb --with ties配合top使用，显示出与四个与第三个相同的记录

order by 专业

--4、查询期末成绩最高的条记录的学生学号和期末成绩以及与第四个学生期末成绩相同的哪些记录。

select top 3 with ties \* from CJB order by 成绩 desc

--5、统计学生表中记录的总数。

select COUNT(\*) 总学生数 from xsb

--6、统计成绩表中所考课程科目的总数。

insert into KCB values (221,'C#',1,100,10)

select \* from KCB

**select COUNT(distinct 课程名) from KCB --distinct差异**

--7、统计成绩表中号课程期中成绩的总分和期中成绩的平均分。

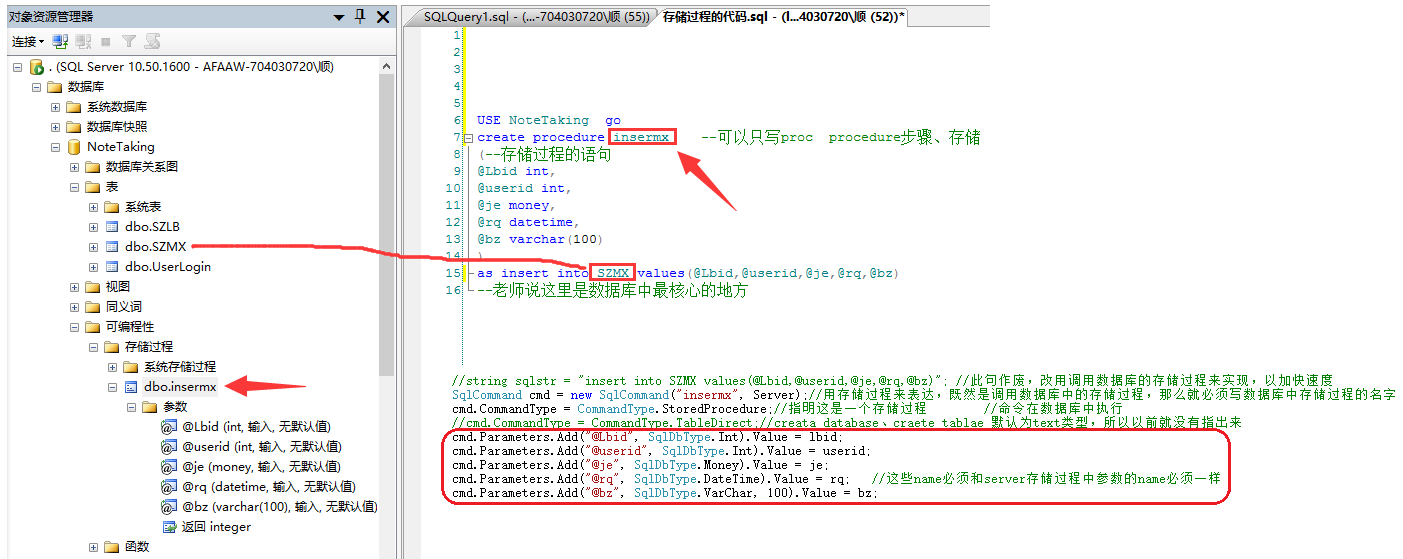
select avg(成绩) 平均分 from CJB where 课程号='101'

select sum(成绩) 总分 from CJB where 课程号='101'

--8、查出档案表中年龄最大和最小的学生的出生日期。

**select MAX(YEAR(出生时间)) 最小年龄,MIN(YEAR(出生时间)) 最大年龄 from xsb**

1. **简单的存储过程**



1. **group by分组**

**字体示例**

--数据的分组查询

--0、统计成绩表中所考课程科目的总数。

select 课程总数=COUNT(distinct 课程号) from CJB --distinct:差异，排除掉相同的

--1、统计各专业学生的总人数。

select COUNT(\*)总人数 from xsb

select 专业,人数=COUNT(\*) from xsb

group by 专业

--2、在cjb中统计参考各门课程的总人数。

delete CJB where 学号='081101' and 课程号='101'

delete CJB where 学号='081104' and 课程号='103'

delete CJB where 学号='081106' and 课程号='103'

select \* from CJB

select 课程号,COUNT(学号) as 人数 from CJB

group by 课程号 --按课程号来进行分组

--3、统计各门课程期末考试的总分、平均分、最高分、最低分与参加考试的人数。

select 课程号, sum(成绩) 总分 ,AVG(成绩) 平均分 ,MAX(成绩) 最高分 ,MIN(成绩) 最低分,COUNT(\*)总人数 from CJB

group by 课程号

--4、分专业分别统计男女生的人数。

select count(专业)人数,专业,性别 from xsb

group by 专业,性别

order by 性别 desc

--5、分别查询、号课程的考试成绩的总分、参考人数、平均分。

select \* from CJB

select 课程号,sum(成绩) 总分 ,AVG(成绩) 平均分 ,MAX(成绩) 最高分 ,MIN(成绩) 最低分,COUNT(\*)人数 from CJB

where 课程号='101' or 课程号='103'

group by 课程号

--6、将总人数大于的专业显示出来。 --从这行开始下面的没有学

Select 专业, 人数=count(\*)

From xsb

Group by 专业

Having count(专业)>5

--7、统计考了两门以上的课程的学生的学号和课程门数。

Select 学号,count(\*)

From cjb

Group by 学号

Having count(课程号)>2

1. **compute汇总语句**

--使用compute 子句汇总数据

--1、列出号课程的明细数据和汇总期中考试的平均分，最高分和最低分。

select \* from CJB

where 课程号='103'

compute avg(cjb.成绩),max(cjb.成绩),min(cjb.成绩)

--2、查询《计算机基础》课程期中考试的明细数据和汇总该课程的考试的总分，参考人数和平均分。

select \* from KCB join CJB

on CJB.课程号=KCB.课程号

where 课程名='计算机基础'

order by 学号

compute sum(成绩),count(成绩),avg(成绩)--计算

--3、按学号分组汇总每个学生的期中成绩的总分，平均分及考试科目数和每个学生考试的明细数据。

select \* from CJB

order by 学号--排序，默认顺

compute sum(成绩),avg(成绩),count(成绩) --重复一行在表的最后计算全部的字段

by 学号

compute sum(成绩),avg(成绩),count(成绩)

--4、查询成绩表中每门课程的明细数据和汇总每门课程期末成绩的总分，

--参考人数和平均分。以及所有科目期末成绩的总分，参考人数和平均分。

select KCB.课程名,xsb.专业,CJB.成绩,xsb.姓名 from

KCB join CJB join xsb

on xsb.学号=CJB.学号

on KCB.课程号=CJB.课程号

order by CJB.课程号

compute sum(成绩),count(成绩),avg(成绩)

1. **IN子查询**

--IN子查询：

--1、查找选修了课程号为的课程的学生情况。

select \* from xsb,CJB where xsb.学号=CJB.学号 and 课程号='103'

select \* from xsb

where 学号 in --因为xsb和cjb中就只有学号字段都共有的

(select 学号 from CJB where 课程号='103') --获得选修了课程的学号，记得只要学号字段

--2、查找未选修《计算机基础》的学生情况。

select \* from xsb

where 学号 not in

(

select 学号 from CJB

where 课程号 in

(select 课程号 from KCB where 课程名='计算机基础')

)

--比较子查询：

--3、查找选修了《计算机基础》的学生学号。

select \* from xsb

where 学号 in --这里不能写等于，因为里面还有in语句，返回值类型不能有多个

(

select 学号 from CJB

where 课程号 = --因为返回值只有一个，所以能够写=而不写in语句

(select 课程号 from KCB where 课程名='计算机基础')

)

--4、查找比所有计算机系的学生年龄都大的学生。

update xsb set 出生时间 = '1980-1-2' where 姓名='胡琴'

select \* from xsb

where 出生时间<all( -- all:所有的 some:一些的 any:任何一个

select 出生时间 from xsb where 专业='计算机'

)

--子查询:

--1、查询号课程期中成绩是分以上的学生姓名。

select \* from xsb join CJB on xsb.学号=CJB.学号 where CJB.成绩>80 --显示条

select distinct xsb.学号,姓名 from xsb join CJB on xsb.学号=CJB.学号 where CJB.成绩>80

select \* from xsb --好像会自带distinct消除学号的差异性 显示条

where 学号 in (

select 学号 from CJB where 成绩 >80

)

--2、列出网络专业的所有学生的成绩情况。

select xsb.\*,CJB.成绩 from xsb join CJB on xsb.学号=CJB.学号 where 专业='网络'

select \* from xsb

where 学号 in (

select 学号 from CJB where 专业='网络'

)

select \* from CJB

where 学号 in (

select 学号 from xsb where 专业='网络'

)

--3、从XSB表中查找所有女学生的姓名、学号及其与号学生的年龄差距。

select 姓名,学号,YEAR(出生时间)-YEAR

(( --注意year函数和子查询的函数括号都要写，这里不需要写in

select 出生时间 from xsb where 学号='081101'

)) as 年龄差异

from xsb

where 性别 = '0'

--4、查询课程号为且期中成绩高于课程号为的所有期中成绩的学生的学号和期中成绩。

select \* from CJB where 课程号='103' and 成绩 > all(

select 成绩 from CJB where 课程号='102'

)

--5、从成绩表中列出号课程的期中成绩

--高于此课程期中成绩平均值

--的记录的学号、姓名、课程号和期中成绩。

select \* from CJB where 课程号='102' and 成绩 > all(

(select AVG(成绩) from CJB where 课程号='102')

)

1. **回顾复习**

------------------------回顾内容------------------------

--创建数据库

create database CPXS

on

(

name='cpxs\_data',

filename='E:\SQLserver\2017 04 06\cpxs.mdf',

size=3,

filegrowth=1

)

log on

(

name='cpxs\_log',

filename='E:\SQLserver\2017 04 06\cpxs.ldf',

size=3,

filegrowth=1

)

use cpxs

--2，创建产品表

create table 产品表

(

产品编号 char(6) not null primary key, --设置主键primary key

产口名称 varchar(20)not null,

价格 numeric(6,1), --数值类型，相当于double (6,1)是代表着保留一位小数点

库存量 int null

)

--3.创建销售商表

create table 销售商表

(

客户编号 char(6) not null primary key,

客户名称 varchar(20)not null,

地区 varchar(20),

负责人 varchar(20),

电话 char(12)

)

--4 创建产品销售表

create table 产品销售表

(

销售日期 date,

产品编号 char(6) not null ,

客户编号 char(6) not null ,

客户名称 varchar(20)not null,

数量 int,

销售额 numeric(20,1)

)

--增加字段

alter table 产品表 --在这里混淆了一下，修改表格用的是alter而不是update，update用于修改记录行

add 产品简介 varchar(50)

use CPXS

select \* from 产品表

--增加字段

alter table 产品表

drop column 产品简介 --这里注意要加column，否则删除的是constraint约束

--插入记录

insert into 产品表 values('132','面包',42,42)

select \* from 产品表

insert into 产品表 values('133222','电脑',5,82),('132132','红酒',100,22)

--修改表格中的价格列,将其打个八折

update 产品表 set 价格=价格\*0.8

select \* from 产品表

--9 将打折后的商品价格低于的删除

delete from 产品表 where 价格 < 50 --这里较为生疏

select \* from 产品表 where 库存量 = 22 --与查询作为对比

--10 增加主键

alter table 产品销售表

add constraint PK\_产品销售表 primary key(产品编号,客户编号,客户名称)

1. **回顾复习**

--以下是针对pxscj数据库中的xsb、cjb、kcb三表的操作，如没有满足条件的记录，请设计数据并插入表中。

--1、查询所有年龄不是岁的学生的姓名、性别、出生日期。

select \* from xsb where YEAR(出生时间)-YEAR(GETDATE())!=27

--2、查询网络和计算机专业学生的学号、姓名和性别。

select 学号,姓名,性别 from xsb where 专业='计算机' or 专业='网络'

--3、在pxscj数据库中将学生的出生日期分为三个阶段：年月日以前出生的为“年以前出生”；年月日－年月日出生的为“年出生”；年月日以后出生为“年以后出生”。请查询学生的姓名、性别、出生时段、专业。

--

select 姓名,性别,出生时段 = case --注意case在显示字段的后面使用

when 出生时间>'1988-12-31' then '89年以前出生'

when 出生时间>'1989-1-1' then '89年出生'

when 出生时间>'1990-1-1' then '89年以后出生'

end,专业 from xsb --使用完case后一定要记得用end结束，否则出错

--4、查询期末成绩最高的条记录的学生学号和期末成绩以及与第四个学生期末成绩相同的哪些记录。

select top 4 \* from CJB order by 成绩 desc --desc倒序 asc顺序

--5、查出档案表中年龄最大和最小的学生的出生日期。

--

select MAX(出生时间) 年龄最大,年龄最小 = MIN(出生时间) from xsb

--6、统计各专业学生的总人数。 --各 group分组查询

--

select 专业,COUNT(\*) as 人数 from xsb group by 专业

--7、分别查询、号课程的期中考试成绩的总分、参考人数、平均分。

--

select 课程号,sum(cjb.成绩) 总分,count(xsb.学号) 参考人数,avg(成绩)平均分

from CJB join xsb

on xsb.学号=CJB.学号

where 课程号 in (102,103)

group by 课程号

--8、从成绩表、学生表、课程表中检索记录，显示输出：学号、姓名、课程名、期平成绩，并根据课程名称排序。

--

select xsb.学号,姓名,KCB.课程名,成绩

from xsb inner join CJB on xsb.学号=CJB.学号

inner join KCB on KCB.课程号=CJB.课程号

order by kcb.课程名

--9、查询选修了计算机基础且期中成绩在分以上的学生的学号、姓名、课程名及期中成绩

--

select x.学号,姓名,课程名,成绩

from xsb x join CJB c on x.学号=c.学号

join KCB k on k.课程号=c.课程号

where 课程名='计算机基础' and 成绩>70

--10、查询哪些专业的学生选修了计算机基础。

--

select \* from xsb join CJB

on xsb.学号=CJB.学号

where 课程号 =

(select 课程号 from KCB where 课程名='计算机基础')

select \* from xsb join CJB

on xsb.学号=CJB.学号

join KCB on CJB.课程号=KCB.课程号

where 课程名='计算机基础'

--11、列出网络专业的所有学生的成绩情况。

select \*,CJB.成绩 from xsb join CJB on xsb.学号=CJB.学号 where 专业='网络'

--12、创建一个网络专业学生的档案表，表名为“wlxsb”。

create table wlxsb

(姓名 varchar(10),

年龄 datetime)

select \* from wlxsb

1. **创建视图**

**作用：就好像是程序中的方法一样，可以省略些许代码**

**缺点：创建的view名字无法通过代码提示出来，需要一个个手敲出来**

--1、创建CS\_KC视图，包括计算机专业各学生的学号、

--其选修的课程号及成绩。要保证对该视图的修改都符合“专业为计算机”这一条件。

create view CS\_KC

as --as:作为 这个给忘了

select a.学号,成绩,b.课程号,c.课程名,专业 from xsb a inner

join CJB b on a.学号=b.学号 --join ：连接

join KCB c on b.课程号=c.课程号

where 专业='计算机'

select \* from CS\_KC

--2、创建计算机专业学生的平均成绩视图CS\_KC\_AVG，

--包括学号（在视图中列名为num）和平均成绩（在视图中列名为score\_avg）。

create view CS\_KC\_AVG(num,score\_avg,课程号,课程名,专业) --如果要取名字的话记得要和下面查询的列数配对上 括号里面的其实就是给字段取名字而已

as --as:作为 当时这个给忘了

select a.学号,成绩,b.课程号,c.课程名,专业 from xsb a inner

join CJB b on a.学号=b.学号

join KCB c on b.课程号=c.课程号

where 专业='计算机'

--3、使用视图CS\_KC查找计算机专业的学生学号和选修的课程号。

select \* from CS\_KC --虽然提示对象名无效,但是不知为何能够出来预想的结果

--4、查找平均成绩在80分以上的学生的学号和平均成绩。

select \* from CS\_KC where 成绩>80

--自己的附加题目 加密

create view CS\_KC1

with encryption --加密

as --as:作为 这个给忘了

select a.学号,成绩,b.课程号,c.课程名,专业 from xsb a inner

join CJB b on a.学号=b.学号

join KCB c on b.课程号=c.课程号

where 专业='计算机'

**注意事项：千万不要把as：作为这个关键字给忘记了**

1. **视图的相关操作**

--1、创建一个加密视图，可以检索各门课程的期中考试参考人数，平均分，最高分，最低分。

create view 成绩加密

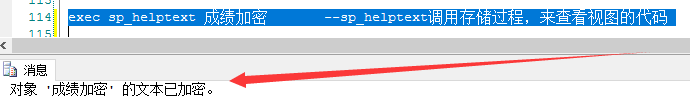
with encryption as --加了这个说明这个视图是加密了的

select COUNT(成绩) 人数,AVG(成绩) 平均分,MAX(成绩) 最高分,MIN(成绩) 最低分 from CJB

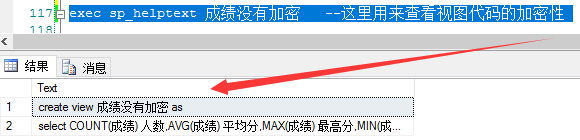
create view 成绩没有加密 as

select COUNT(成绩) 人数,AVG(成绩) 平均分,MAX(成绩) 最高分,MIN(成绩) 最低分 from CJB

exec sp\_helptext 成绩加密 --sp\_helptext调用存储过程，来查看视图的代码



exec sp\_helptext 成绩没有加密 --这里用来查看视图代码的加密性



--2、创建一个视图，可以检索学号、课程号、期末成绩，并且通过该视图修改、插入数据时期末成绩必须成绩>70 and 成绩<80

drop view 成绩修改视图

create view 成绩修改视图 as

select \* from CJB

where 成绩>70 and 成绩<80

with check option --check:检查 option:选项 加了这句的意思是说只能够按照上面的条件去修改数据库

--3、创建视图cs\_xs,可以检索计算机专业学生的基本信息；向CS\_XS视图中插入以下记录：

create view cs\_xs as

select \* from xsb where 专业='计算机'

insert into cs\_xs

values('081115', '刘明仪', 1,'1998-3-2', '计算机',50,null)

--4、将CS\_XS视图中所有学生的总学分增加。

update cs\_xs set 总学分=总学分+ 8

--5、将CS\_KC视图中学号为的学生的号课程成绩改为。

update cs\_kc set 成绩=90

where 学号='081101' and 课程号='101'

--6、删除CS\_XS中女同学的记录。

delete from cs\_xs where 性别 = 0 --删除中记得有个“from”

--7创建一个“课程成绩视图”。要求通过该视图插入一门新课程：

--课程号为‘’、课程名为‘高等数学’；参加考学生的学号为‘’、php的期中成绩为分。

create view 课程成绩视图(学号,课程号,成绩,课程号,课程名)as

select 学号,a.课程号,成绩,b.课程号,课程名

from cjb a join kcb b

on a.课程号=b.课程号

Insert into 课程成绩视图(课程号,课程名)

values ('108','高等数学') --涉及多表只能够用这种方式

Insert into 课程成绩视图(学号,课程号,成绩)

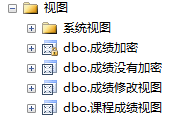
Values('020101','108',89 )

Insert into 课程成绩视图

values('020101','108',89,'108','生物' )

--错误视图或函数'课程成绩视图' 不可更新，因为修改会影响多个基表。

老师说第一题第二题还有最后一题比较重要



1. **视图的相关操作**

--1、创建一个“计算机基础成绩视图”，通过该视图修改王林同学的计算机基础的成绩为分。

create view 计算机基础成绩视图 as

select a.学号,姓名,b.成绩,c.课程名 from xsb a inner

join CJB b on a.学号 = b.学号

join KCB c on b.课程号 = c.课程号

select \* from 计算机基础成绩视图

alter view 计算机基础成绩视图 as

select a.学号,姓名,b.成绩,c.课程名 from xsb a inner

join CJB b on a.学号 = b.学号

join KCB c on b.课程号 = c.课程号

where 课程名 = '计算机基础'

update 计算机基础成绩视图 set 成绩 = 95

where 姓名 = '王林' --你居然忘记了打这句话，后果为把所有的成绩都会修改成为

--2.1建一个包含有“计算机基础”和“ASP”课程的“成绩视图”，该视图含有xsb的学号、姓名、专业字段，cjb的学号、课程号、成绩字段，kcb的课程号、课程名、字段；

create view 成绩视图(a学号,姓名,专业,b学号,b课程号,成绩,c课程号,c课程名) as

select a.学号,姓名,专业,b.学号,b.课程号,成绩,c.课程号,c.课程名 from xsb a

join CJB b on a.学号 = b.学号

join KCB c on b.课程号 = c.课程号

where c.课程名 = '计算机基础' or 课程名 = 'SQL'

--2.2过该视图在xsb中插入一条记录，学号为，姓名为刘自信，专业为网络专业；

select \* from xsb

insert 成绩视图(a学号,姓名,专业) values('081601','刘自信','网络')

--2.3在成绩表中插入该同学的号“VB”这门课程的期末成绩为分；

insert into 成绩视图 (b学号,b课程号,成绩) --注意插入的时候字段名为上面创建视图时的

values('081601','107',89) select \* from 成绩视图

--2.4修改成绩表中王林这个学生计算机基础的期末成绩为分。

update 成绩视图 set 成绩 = 88 where 姓名 = '王林'

1. **declare创建全局变量**

--1、创建局部变量@var1、@var2并赋值，然后输出变量的值。

declare @var1 int,@var2 varchar(10)

set @var1 = 18 set @var2 = '我叫啦啦啦'

select @var1 变量,@var2 变量

--2、创建一个名为sex的局部变量，并在SELECT语句中使用该局部变量查找表XSB中所有女同学的学号、姓名。

declare @sex varchar = '女'

select \* from xsb where 性别 = @sex

--3、使用查询为变量赋值。

declare @shun varchar(10) , @shunn varchar(10) , @shunnn varchar(10)

select @shun='顺',@shunn='顺顺',@shunnn='顺顺顺'

print @shun+@shunn+@shunnn

--4、使用SELECT语句为局部变量赋值。

declare @name varchar(5),@sexx char(1),@yeardate datetime,@xuehao int = 081203

select @name=姓名,@sexx=性别,@yeardate=出生时间 from xsb where 学号= @xuehao

select @name,@sexx,@yeardate --查询版

declare @name varchar(5),@sexx char(1),@yeardate datetime,@xuehao int = 081203

select @name=姓名,@sexx=性别,@yeardate=出生时间 from xsb where 学号= @xuehao

print '姓名:'+@name+'性别:'+@sexx+'出生时间:'+CONVERT(varchar(10),@yeardate)

--扩展拼接字符串版 注意转换为varchar类型才能显示

1. **新知识if、else、end、begin语句，复习case**

**字体示例**

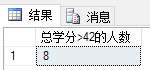
--1、用于查询总学分大于的学生人数。

declare @num int

select @num=(select count(学号) from xsb where 总学分>42)

if @num<>0 --<>在数据库中叫'不等于'

select @num as '总学分>42的人数'



--2、如果“计算机基础”课程的平均成绩高于分，则显示“平均成绩高于分”。

declare @scoue int = 60 --定义并且初始化分数

if (select avg(成绩) from CJB inner join KCB on CJB.课程号=KCB.课程号

where 课程名='计算机基础')>=@scoue

goto a --跳转到a的语句块

else

goto b

a:begin --begin：开始，标志着语句块的开始

print '平均成绩大于等于'+convert(char(10),@scoue)+'分'

return --如果不写这里，如果正好又执行了这一段语句块，

--那么将会把下面的b语句块也执行掉，因为return只能结束当前语句块

end --end：结束，标志着语句块的结束

b:begin

print '平均成绩低于'+convert(char(10),@scoue)+'分'

return

**end --end如果不写的话就会出错，begin一定要配合end来使用**

--3、使用CASE语句，根据性别值输出“男”或“女”。

**select 姓名, 性别 = --在这里为别名 这里磨蹭了半天**

**case**

**when 性别 =1 then '男'**

**when 性别 =0 then '女'**

**end**

**from xsb**

**select 姓名, sex = --sex在这里为别名**

**case 性别 --取性别的值**

**when 1 then '男'**

**when 0 then '女'**

**end**

**from xsb**

1. **标量值函数(函数)**

**其实标量值函数就是returns变量的函数**

**值得注意的是和C#差别还有点大，返回值类型在最后面。并且要写成returns有s加类型加as**

--1、创建一个“日期天数差”函数，用于计算两个日期数据之间相差的天数。

create function 日期天数差(@a datetime,@b datetime) --创建形参

**returns int as**

begin

return datediff(dd,@a,@b) **--dd天 mm月 yy年**

end **--datediff()方法专门用来比较两个年份时间的差距**

go

declare @a int, @b datetime ='2017-08-20',@c datetime = '2007-08-20'

set @a = dbo.日期天数差(@b,@c) --传入两个局部变量来实现

print '年份'+convert(varchar(20),year(@b))+

'和年份'+convert(varchar(20),@c)+'之间相差'

+convert(varchar(20),@a)+'天'

go

--2创建一个“日期年份差”函数，用于计算两个日期数据之间相差的年数。

create function 日期年数差(@a datetime,@b datetime)

returns int as

begin

return datediff(yy,@a,@b) --dd天 mm月 yy年

end

go

declare @a int, @b datetime ='2017-08-20',@c datetime = '2007-08-20'

set @a = dbo.日期年数差(@b,@c)

print '年份'+convert(varchar(20),@b)+

'和年份'+convert(varchar(20),@c)+

'之间相差'+convert(varchar(20),@a)+'年'

go

--3、在学生管理系统数据库中创建一个统计某门课程考试的人数的函数“某门课程参考人数”

create function 某门课程参考人数(@kcm varchar(20))

returns int as --注意是'returns' 并且别忘了varcahr后面有个值大小

begin

declare @rs int

select @rs=COUNT(\*) from KCB where 课程名=@kcm

return @rs

end

declare @a varchar(20)='计算机基础'

print '选修了'+convert(varchar(20),@a)+'的人数有：'

+convert(varchar(20),dbo.某门课程参考人数(@a))+'个人'

--4、应用创建的函数查询档案表中学生的姓名、性别、出生日期、年龄。

select 姓名,性别,出生时间,dbo.日期年数差(出生时间,GETDATE()) as 年龄 from xsb

1. **表值函数**

--1、创建用户定义函数，实现计算全体学生某门功课平均成绩的功能。

create function

某门功课平均成绩 (@kcm varchar(10))

**returns int**

as

begin

declare @pjz int

select @pjz = AVG(成绩) from xsb join CJB

on xsb.学号=CJB.学号 join KCB

on KCB.课程号=CJB.课程号

where 课程名=@kcm

return @pjz

end

print dbo.某门功课平均成绩('计算机基础')

--2、创建内嵌函数fn\_view()，可查询指定专业的学生的学号和姓名。

内嵌表值函数

create function fn\_view(@zy varchar(10))

**returns table as**

return select \* from xsb where 专业=@zy

select \* from dbo.fn\_view('网络')

**--注意前面不需要打印print，因为返回的是表格，表格无法打印**

--3、先创建视图，查询学生的各科成绩（学号、姓名、课程名、成绩）；

create view 各科成绩 as

select xsb.学号,姓名,课程名,成绩

from xsb join CJB on xsb.学号=CJB.学号 join KCB

on CJB.课程号=KCB.课程号

--并利用视图再创建内嵌函数st\_score()，调用st\_score()查询学号为的学生的各科成绩。

create function st\_score(@xh int) returns table as

return select \* from 各科成绩 where 学号=@xh

select \* from dbo.st\_score('081101')

**注意调用函数的时候都需要加上dbo.**

1. **多语句表值函数**

--1、创建一个内嵌表函数能够查看某个专业、某门课程的考试情况。能显示学号、姓名、专业、课程名称、考试成绩；应用创建的函数进行查询。

drop function ckzymmkc --删除函数

create function ckzymmkc(@zy varchar(10) = '计算机')

returns table as

return --仅仅只有一个语句块不写begin否则出错

select xsb.学号,姓名,专业,课程名,成绩 from xsb join CJB

on xsb.学号=CJB.学号 join KCB

on KCB.课程号=CJB.课程号 where 专业=@zy

**select \* from dbo.ckzymmkc(default) where --采用默认参数**

**select \* from dbo.ckzymmkc('网络') where 课程名 = 'C#' --再次筛选**

--2、在PXSCJ数据库中创建返回表的函数score\_table()，通过以学号作为实参调用该函数，可显示该学生各门功课的成绩和学分。

create function score\_table(@xh int)

**returns @ta table --这里是返回一个表的变量，然后跟着表的结构**

(

学号 char(7),

姓名 varchar(10),

课程名 varchar(10),

成绩 int,

xf int

)

as --还是不要忘了as

begin

insert @ta --插入到表变量中去

select xsb.学号,姓名,课程名,成绩,学分 from xsb,KCB,CJB

where xsb.学号=CJB.学号 and KCB.课程号=CJB.课程号 and xsb.学号 = @xh

return --返回一个空的

end

**--要点：开头返回的必须为一个表值变量后面加table后紧跟表结构，字段名会在调用此函数时体现，函数体必须包含在begin end内，begin后面必须为inset+表变量，end上面必须为return，后面不跟任何东西**

select \* from score\_table('081101')

--3、在PXSCJ数据库中创建一个多语句表值函数，可以返回某门课程或所有课程的学号、姓名、课程名、成绩。

create function kcmde(@kcm varchar(10))

returns table as --再次验证仅仅只有一个语句块不写begin否则出错

return select x.学号,姓名,课程名,成绩

from xsb x inner join CJB c

on x.学号=c.学号 join KCB k

on k.课程号=c.课程号 where 课程名=@kcm --课程名等于形参

select \* from kcmde('计算机基础')

1. **练习**

--1、查询总学分在到分之间的学生学号和姓名

select xsb.学号,xsb.姓名,总学分 from xsb

where 总学分 >= 40 and 总学分 <= 50

select 学号, 姓名, 总学分

**from xsb where 总学分 between 40 and 50**

--2、查询总学分在到分之外的学生学号和姓名

select xsb.学号,xsb.姓名,总学分 from xsb where 总学分 <= 40 or 总学分 >= 50

select 学号, 姓名, 总学分

from xsb

where 总学分 not between 40 and 50

--3、查询课程名以“计”或C开头的情况

**select \* from KCB where 课程名 like '[c]%'**

**select \* from KCB where 课程名 like '计%'**

--4、查询所有选课学生的姓名

select distinct 姓名 from xsb x inner join CJB c

on x.学号 = c.学号 join KCB k

on c.课程号=k.课程号

where c.课程号 != 'null'

select distinct 姓名 from xsb

where exists

(select \* from cjb where xsb.学号= cjb.学号)

--5、查询成绩高于“王林”最高成绩的学生姓名以及成绩

select a.学号,a.姓名,a.性别,b.成绩

from xsb a inner join CJB b

on a.学号=b.学号 where

成绩>(select max(c.成绩)

from xsb x join CJB c on x.学号=c.学号

where x.姓名='王林' )

update CJB set 成绩=成绩-10 where CJB.学号=081101

select 姓名, 课程名, 成绩

from xsb, cjb, kcb

where 成绩> all

(select top 2 b.成绩 from xsb a, cjb b where a.学号= b.学号 and a.姓名= '王林')

and xsb.学号=cjb.学号

and kcb.课程号=cjb.课程号

and 姓名<>'王林'

1. **练习**

--1、查询Student表中的所有记录的Sname、Ssex和Class列。

select Sname,Ssex,Class from STUDENT

--2、查询教师所有的单位即不重复的Depart列。

select distinct(DEPART) from TEACHER

--3、查询Student表的所有记录。

select \* from STUDENT

--4、查询Score表中成绩在到之间的所有记录。

select \* from SCORE where DEGREE between 60 and 80

--5、查询Score表中成绩为，或及的记录。in(85,86,88)

select \* from SCORE where DEGREE in (85,86,88)

--6、查询Student表中“95031”班或性别为“女”的同学记录。

select \* from STUDENT where CLASS = '95031' or SSEX = '女'

--7、以Class降序查询Student表的所有记录。

select \* from STUDENT order by SNO desc --desc:倒序 asc:顺序

--8、以Cno升序、Degree降序查询Score表的所有记录。

select \* from SCORE order by SNO asc,DEGREE desc --多个排序用逗号

--9、查询“”班的学生人数。

select COUNT(\*) '95031班的人数为' from STUDENT where CLASS = '95031'

--10、查询Score表中的最高分的学生学号和课程号。

select top 1 \* from SCORE order by DEGREE desc

select \* from SCORE where DEGREE=(select MAX(DEGREE) from SCORE)

--11、查询-105号课程的平均分。

select AVG(DEGREE) '3-105号课程的平均分为' from SCORE where CNO = '3-105'

--12、查询Score表中至少有名以上学生选修的并以开头的课程的平均分数。

select \* from SCORE

**select CNO 课程号,AVG(DEGREE) 平均分,COUNT(cno) 人数 from SCORE**

**where CNO like '3%'**

**group by cno**

**having count(\*)>3 --having count(\*)新内容，分组统计**

--13、查询最低分大于，最高分小于的Sno列。

select SNO from SCORE where DEGREE>70 and DEGREE < 90

--14、查询所有学生的Sname、Cno和Degree列。

select Sname,Cno,Degree from STUDENT join SCORE on STUDENT.SNO=SCORE.SNO

--15、查询所有学生的Sno、Cname和Degree列。

select STUDENT.Sno,Cname,Degree

from STUDENT join SCORE

on STUDENT.SNO=SCORE.SNO join COURSE

on COURSE.CNO=SCORE.CNO

--16、查询所有学生的Sname、Cname和Degree列。

select Sname,Cname,Degree

from STUDENT join SCORE

on STUDENT.SNO=SCORE.SNO join COURSE

on COURSE.CNO=SCORE.CNO

--17、查询“”班所选课程的平均分。

select AVG(DEGREE) from SCORE where SNO in (

select sno from STUDENT where CLASS = '95033')

--18、查询和学号为'109'的同学同年出生的所有学生的Sno、Sname和Sbirthday列。(用year()函数)

select Sno,Sname,Sbirthday

from STUDENT

where YEAR(SBIRTHDAY)=(select year(SBIRTHDAY) from STUDENT where SNO=109)

1. **练习、union连接表查询**

当时没有做完

--1、查询“张旭“教师任课的学生成绩。

select \* from score where cno= --成绩表的课程号=

(select cno from course where tno= --获得课程号，条件为教师号为

(select tno from teacher where tname='张旭'))

--2、查询选修某课程的同学人数多于人的教师姓名。

**select a.tname from teacher a where a.tno in( ---tno教师号**

**select x.tno from course x,score y where x.cno=y.cno --con课程号**

**group by x.tno --tno教师号**

**having count(x.tno)>5)**

--3、查询存在有分以上成绩的课程cno.

select cno from score group by cno having max(degree)>85

--4、查询出“计算机系“教师所教课程的成绩表。

select s.\* from score s join course c

on s.cno=c.cno join teacher t

on t.tno=c.tno where t.depart='计算机系'

--5\*、查询所有教师和同学的name、sex和birthday.（union））

select t.tname,t.tsex,t.tbirthday from teacher t

**union --新内容,如果两个表的字段一样的话就可以进行连接**

select s.sname,s.ssex,s.sbirthday from student s

--6\*、查询成绩比该课程平均成绩低的同学的成绩表。

select a.\* from score a --score成绩表

where degree< --成绩小于

**(select avg(degree) from score b where a.cno=b.cno)--a.cno=b.cno确保课程号对上再对比**

--7、查询所有有课教师的tname和depart 名字和系部

select tname,depart

from teacher join course on teacher.tno=course.tno

--8、查询至少有名男生的班号。

**select class from student where ssex='男' group by class having count(ssex)>1**

--9、查询student表中不姓“王”的同学记录。

select \* from student where sname not like '王%' --重在not

--10、查询student表中每个学生的姓名和年龄。

select s.sname,年龄 = **year(getdate())-year(s.sbirthday)** from student s

--11、查询student表中最大和最小的sbirthday日期值。

select year(max(sbirthday)),year(min(sbirthday)) from student

--12、以班号和年龄从大到小的顺序查询student表中的全部记录。

select \* from student order by sbirthday desc

--13、查询“男”教师及其所上的课程。

select \* from course c join teacher t on c.tno=t.tno where t.tsex='男'

--14、查询最高分同学的sno、cno和degree列。

select top 1 \* from score order by degree desc

--15、查询所有选修“计算机导论”课程的“男”同学的成绩表

select student.\*,score.\* from score join student

on student.sno=score.sno where cno=

(select cno from course where cname='计算机导论') and student.ssex='男'

1. **初识存储过程**

--1、设计简单的存储过程：student\_info，返回号学生的成绩情况。该存储过程不使用任何参数。

create procedure 简单的存储过程 as **--as不要给忘了**

select \* from xsb inner join CJB on xsb.学号=CJB.学号

**exec 简单的存储过程 --执行存储过程**

--2、创建存储过程GetBoyInfo，查询年出生的男学生信息。

create proc GetBoyInfo as

select \* from xsb where convert(int,year(出生时间))=1989 --因为没有年的就了

exec GetBoyInfo

--3、使用带参数的存储过程：student\_info1，从PXSCJ数据库的三个表中查询

--某人指定课程的成绩和学分。该存储过程接收与传递参数精确匹配的值。

create proc student\_info1 @str **varchar(20)** as

select xsb.学号,xsb.性别,xsb.姓名,KCB.课程号,KCB.课程名,xsb.总学分 from xsb join CJB

on xsb.学号=CJB.学号 join KCB

on CJB.课程号=KCB.课程号

where KCB.课程名=@str

exec student\_info1 'C#' --三种调用方式

exec student\_info1 C#

exec student\_info1 @str=C#

1. **初识带参数的存储过程**

--做题之前认识新内容

**select \* from sys.procedures --查看数据库内所有的存储过程**

**select \* from sys.procedures where name = '存储过程名'**

--巧用exists重复创建存储过程

if exists(select \* from sys.procedures where name = '查看xsb')--存在就true

drop proc 查看xsb --存在就删除它

go

create proc 查看xsb as --不管怎样都创建它

select \* from xsb

exec 查看xsb

--1、输入学号，查看该学生的开课门数。（用输入参数）

select \* from xsb

select \* from CJB --先插入改变的与众不同

insert cjb values('081101','010','55')

insert cjb values('081101','110','55')

drop proc 开课门数

create proc 开课门数 @xh int=081101 as

select COUNT(\*) from CJB where 学号=@xh

--2、输入学号，查看该学生的开课门数。（用输出参数）

drop proc 开课门数\_用输出参数

create proc 开课门数\_用输出参数

@xh int=081101, **@num int output** as **--传出参数必须有output**

select @num=COUNT(\*) from CJB where 学号=@xh

--3、分别调用前两个函数查看‘’的开课情况。

exec 开课门数 081102

declare @n int

exec 开课门数\_用输出参数 081101 , **@n output --多个参数记得逗号**

select @n 门数

--4、创建一个存储过程do\_insert，作用是向XSB表中插入一行数据

--('091201', '陶伟', 1, '1990-03-05', '软件工程',50, NULL);

create proc do\_insert as

insert xsb values('091201', '陶伟', 1, '1990-03-05', '软件工程',50, NULL)

select \* from xsb

--创建另外一个存储过程do\_action

--在其中调用第一个存储过程，并根据条件处理该行数据，处理后输出相应的信息

--（输入修改数据，如姓名变量刘英，性别为；输入则删除该记录）。

drop proc do\_action

create proc do\_action @n int as

begin

exec do\_insert

if(@n=0)

begin

update xsb set 姓名='刘英',性别=0 where 学号='091201'

print '修改成功'

end

else if(@n=1)

begin

delete xsb where 学号='091201'

print '该记录已删除'

end

else

print '请输入正确的数字'

end

exec do\_action 0

exec do\_action 1

select \* from xsb

1. **存储过程进阶**

--1、在学生管理系统中创建一个可以返回数据集的存储过程“考试查询”。

--可以查询某专业某姓名学生的学号、姓名、课程名、成绩。

create proc 考试查询 @zy varchar(20),@xm varchar(20) as

select xsb.学号,姓名,专业,课程名,成绩

from xsb join CJB

on xsb.学号=CJB.学号 join KCB

on CJB.课程号=KCB.课程号

where 专业=@zy and 姓名=@xm

exec 考试查询 计算机 , 王林

--2、在学生管理数据库中创建一个“统计考试科目数”的存储过程。

--其中输入参数是某学生的姓名，输出参数是这个学生的考试科目数。

--执行该存储过程，查看输出参数的结果。

create proc 统计考试科目数

@name varchar(10),@num int output as

select @num=COUNT(\*) from xsb join CJB

on xsb.学号=CJB.学号 where 姓名=@name

declare @mz varchar(10),@n int

set @mz = '王燕'

exec 统计考试科目数 @mz,@n output

print @n

--3、在学生管理系统中创建一个“查找学生”的存储过程。

--如果找到有该学生存储过程就返回，否则返回。

drop proc 查找学生

create proc 查找学生 @name varchar(20) as --这里的varchar(20)忘记写后面的范围了

if exists(select \* from xsb where 姓名 = @name) --exists:存在

return 1 --return返回 begin end {}

else

return 0

declare @s int

exec @s=查找学生 '李方方'

print @s --print:打印

exec 查找学生 '李方方'

--4、在学生管理系统中创建一个“考试查询”的存储过程，

--如果查找到该学生的姓名并且参加了一门以上的考试，存储过程就返回，

--如果找到了该学生但没有参加任何考试存储过程就返回，否则就返回。

drop proc 考试查询

create proc 考试查询 @name varchar(10) as

if exists(select 姓名 from xsb,CJB where 姓名=@name and xsb.学号 = CJB.学号)

return 1

else if exists(select \* from xsb where 姓名=@name)

return 2

else

return 0

--下面插入学生表一个字段，作为没有成绩的

insert into xsb values('040203','顺',1,'1998-07-13','软件','100',null)

declare @s int

exec @s = 考试查询 顺

if @s=1

print cast(@s as varchar(20))+'参加了考试并且有成绩'

if @s=2

print convert(varchar(20),@s)+'有这个学生,但是没有成绩'

if @s=0

print cast(@s as varchar(20))+'请确保名字是否输入正确'

--cast(@s as varchar(20)) === convert(varchar(20),@s) 都是转换的意思,注意区分

1. **实训前的练习**

--1. 在数据库supplier中，创建有如下个表：

create database supplier

on (

name='supplier',

filename='E:\SQLserver\2017 05 18\supplier.mdf',

size=3mb,

maxsize=100mb,

filegrowth=1mb

)

log on(

name='supplier\_ldf',

filename='E:\SQLserver\2017 05 18\supplier.ldf',

size=3mb,

maxsize=100mb,

filegrowth=1mb

)

go

--供应商表S(SNO,SNAME,CITY)

use supplier

create table s

(SNO int, --SNO、SNAME、CITY分别表示供应商代码、供应商姓名、城市

SNAME varchar(20),

CITY varchar(20))

--零件表J(JNO,JNAME,COLOR,WEIGHT)

create table j --供应商所在城市，JNO、JNAME、COLOR、WEIGHT分别表示零件代码、零件名、颜色和重量

(JNO int,

JNAME varchar(20),

COLOR varchar(10),

WEIGHTT int)

--工程表P(PNO,PNAME,CITY)

create table p( --PNO、PNAME、CITY分别表示工程代码、工程名、工程所在城市

PNO int,

PNAME varchar(20),

CITY varchar(20))

--供应情况表SPJ(SNO,PNO,JNO,QTY)

create table spj( --SNO供应商代码，PNO工程代码，JNO零件代码，QTY表示某供应商供应某工程某种零件的数量

SNO int,

PNO int,

JNO int,

QTY int)

--然后在每个表中录入或插入若干记录。

select \* from j --零件表 零件代码、零件名、颜色和重量

insert into j values(1,'扳手','sliver',50)

insert into j values(2,'老虎钳','red',20),

(3,'螺丝刀','yellow',10),

(4,'钉子','white',1),

(5,'梯子','green',200)

select \* from p --工程表 工程代码、工程名、工程所在城市

insert into p values(001,'娄底三大桥','娄底'),

(002,'长沙一大桥','长沙'),

(003,'湘潭万达广场','湘潭'),

(004,'株洲汽车工程学院新址','株洲'),

(005,'长沙广电','长沙')

select \* from s --供应商表 分别表示供应商代码、供应商姓名、城市

insert into s values(1,'即可','湘潭'),

(2,'饭店','长沙'),

(3,'早上','株洲'),

(4,'那么','长沙'),

(5,'请我','娄底')

select \* from spj --供应情况表 SNO供应商代码，PNO工程代码，JNO零件代码，QTY表示某供应商供应某工程某种零件的数量

insert into spj values(1,1,1,123),

(2,2,2,45),

(3,3,3,54),

(4,4,4,36),

(5,5,5,12)

--2、分别写出SQL语句，完成如下功能：

--(1)查询出重量大于或颜色为"red"的零件名；

select \* from j where WEIGHTT>30 or COLOR = 'red'

--(2)查询出每个供应商为每个工程供应零件的数量

select sum(QTY) from SPJ group by SNO,PNO

--(3)查询出给"北京"的工程供应"齿轮"零件的供应商名；

select SNAME from S where SNO in (

select SNO from SPJ,P,J

where (CITY='长沙') and (SPJ.PNO=P.PNO)

and (JNAME='老虎钳') and (SPJ.JNO=J.JNO))

select \* from j

select \* from p

select \* from s

select \* from spj

--(4)建一个视图，定义为所有"green"颜色的零件。

create view gre as

select \* from j where COLOR='green'

select \* from gre

--3. 现有关系数据库‘study’如下：

--学生(学号，姓名，性别，专业、奖学金) 课程(课程号，名称，学分) 学习(学号，课程号，分数)

--用SQL实现：

--(0)创建数据库并录入若干记录；

create database study

on (

name='study',

filename='E:\SQLserver\2017 05 18\study.mdf',

size=3mb,

maxsize=100mb,

filegrowth=1mb

)

log on(

name='study\_ldf',

filename='E:\SQLserver\2017 05 18\study.ldf',

size=3mb,

maxsize=100mb,

filegrowth=1mb

)

go

create table student( --学生(学号，姓名，性别，专业、奖学金

sidd int,

sname varchar(10),

ssex varchar(2),

szy varchar(10),

sjxj int)

create table cours( -- 课程(课程号，名称，学分)

cid int,

cname varchar(10),

cxf int)

create table scour( --成绩(学号，课程号，分数)

sidd int,

cid int,

sfs int)

--(1)查询没有获得奖学金、同时至少有一门课程成绩在分以上的学生信息，包括学号、姓名和专业；

select student.sidd,student.sname,student.szy from student join scour on scour.sidd = student.sidd join cours

on cours.cid = scour.cid where scour.sfs > 95

--(2)查询没有任何一门课程成绩在分以下的所有学生的信息，包括学号、姓名和专业；

select student.sidd,student.sname,student.szy from student join scour on scour.sidd = student.sidd join cours

on cours.cid = scour.cid where scour.sfs < 80

--(3)对成绩得过满分(100分)的学生，如果没有获得奖学金的，将其奖学金设为1000元；

update student

set sjxj = 1000 where student.sidd in(

select distinct scour.sidd from student,scour

where (student.sjxj is null) and (student.sidd=scour.cid) and(scour.sfs=100))

--(4)定义学生成绩得过满分(100分)的课程视图AAA，包括课程号、名称和学分；

create view AAA as

select distinct cours.cid,cours.cname,cours.cxf

from cours,scour

where (cours.cid=scour.cid) and(scour.sfs=100)

1. **教师数据库**

--现有关系数据库如下：

--数据库名：教师数据库

create database 教师数据库

on

(name = 教师数据库,

filename = 'E:\SQLserver\2017 05 19\教师数据库.mdf',

filegrowth=1,

size=3)

log on

(name = 教师数据库\_ldf,

filename = 'E:\SQLserver\2017 05 19\教师数据库.ldf',

filegrowth=1,

size=3)

--教师表(编号char(6)，姓名，性别，民族，职称，身份证号)

create table 教师表(

编号 char(6) primary key,

姓名 varchar(10),

性别 varchar(2) check(性别='男' or 性别 ='女'), --仅仅只能输入男和女

民族 varchar(10) not null default '汉族',

职称 varchar(10),

身份证号 int unique,

constraint sfzhaoma unique(身份证号))

--课程表(课号char(6)，名称)

create table 课程表(

课程号 char(6) primary key,

课程名 varchar(10))

alter table 课程表

alter column 课程名 varchar(20) --修改表字段的数据类型

--任课表(ID，教师编号,课号，课时数)

create table 任课表(

ID int identity(1,1), --新知识：标识列identity(1,1)左边是从开始，增量为

教师编号 char(6),

课程号 char(6),

课时数 int check(课时数> 1 and 课时数<100)) --仅仅在---100这个范围内

--用SQL语言实现下列功能的sql语句代码：

--1. 创建上述三表的建库、建表代码；

-- 要求使用：主键(教师表.编号，课程表.课号)、

--外键(任课表.教师编号，任课表.课号)、

alter table 任课表

add constraint 教师表\_编号\_任课表\_教师编号 foreign key(教师编号) references 教师表(编号)

alter table 任课表

add constraint 课程表\_课号\_任课表\_课号 foreign key(课程号) references 课程表(课程号)

-- 默认(民族)、非空(民族，姓名)、唯一(身份证号)、检查(性别、课时数),自动编号(ID)

--2. 将下列课程信息添加到课程表的代码

-- 课号 课程名称

-- 100001 SQL Server数据库

-- 100002 数据结构

-- 100003 VB程序设计

select \* from 课程表

insert into 课程表

values(100001,'SQL Server数据库'),

(100002,'数据结构'),

(100003,'VB程序设计')

-- 修改 课号为的课程名称：Visual Basic程序设计

update 课程表 set 课程名='Visual Basic程序设计' where 课程号=100003

-- 删除 课号为的课程信息

delete 课程表 where 课程号=100003

--3. 写出创建[任课表视图](教师编号，姓名，课号，课程名称，课时数)的代码；

create view 任课表视图 as

select a.编号,a.姓名,c.课程号,c.课程名,b.课时数

from 教师表 a join 任课表 b on a.编号 = b.教师编号 join 课程表 c

on b.课程号=c.课程号

select \* from 任课表视图

--4. 写出创建[某门课任课教师]内嵌表值函数以及检索的代码；

create function 某门课任课教师

(@kename varchar(20)) --局部变量记得打括号

returns table as

return (select a.编号,a.姓名,c.课程号,c.课程名,b.课时数

from 教师表 a

join 任课表 b on a.编号=b.教师编号

join 课程表 c on b.课程号=c.课程号)

select \* from 某门课任课教师 ('数据结构') --传值记得打括号,就跟C#一样

create function [某门课任课教师](@课程名 varchar(15))

returns table as

return (select 课程名称, 课时数, 教师姓名=姓名 from 任课表视图

where 课程名=@课程名)

go

-- 检索：所有代'SQL Server数据库'这门课程的老师姓名；

select \* from 某门课任课教师 ('SQL Server数据库')

--5. 写出创建[统计课时数]：输出最大课时数、最低课时数、平均课时的存储过程以及执行代码；

drop proc 统计课时数

create procedure 统计课时数 as

select 最大课时数 = max(a.课时数),最小课程数 = min(a.课时数),

平均课时数 = avg(a.课时数) from 任课表 as a

exec 统计课时数

--6. 写出创建：计算某教师代课总课时，并将值返回的存储过程以及执行代码。(6分)

--执行：计算“郭老师”的总课时。

drop proc 某教师代课总课时

create procedure 某教师代课总课时

@教师名 nchar(16)

as

begin

declare @总课时 int

select @总课时=sum (课时数) from 任课表视图

where 姓名 = @教师名

print '总课时为' + convert(varchar(20), @总课时)

print convert(varchar(20), @教师名)+ '的总课时为'

end

go

execute 某教师代课总课时 '老师名'

--7. 检索有一门或一门以上课程课时数大于的所有教师的信息，包括编号、姓名。

select \* from 教师表 where 教师表.编号 in(

select distinct 教师编号 from 任课表 where 课时数 > 90) -distinctct差异

--8. 建一个规则，并将其绑定到教师表的职称列上，规定取值为（'教授','副教授','讲师', '助教'）之一。

create rule zhicheng \_rule

as @zhicheng in ('教授','副教授','讲师', '助教')

go

sp\_bindrule zhicheng\_rule, '教师表.职称'

1. **医院数据库**

--现有关系数据库如下：

--数据库名：医院数据库

use master

drop database hospital

go

create database hospital on(

name = hospital,

filename = 'E:\SQLserver\2017 05 24\hospital.mdf',

filegrowth = 1,

size = 3)

log on(

name = hospital\_log,

filename = 'E:\SQLserver\2017 05 24\hospital.ldf',

filegrowth = 1,

size = 1)

go

--医生表(编号，姓名，性别，出生日期，职称)

use hospital

create table yishengbiao(

yid int not null,

name varchar(10),

sex varchar(2),

birthday datetime,

zhicheng varchar(10) not null)

--病人表(编号，姓名，性别，民族，身份证号)

create table bingrenbiao(

bid int not null identity(1,1),

name varchar(10) not null,

sex varchar(2),

mingzu varchar(10),

idcard int)

--病历表(ID，病人编号,医生编号，病历描述)

create table binglibiao(

blid int identity(1,1),

brid int,

ysid int,

binglimiaoshu varchar(100))

go

--用SQL语言实现下列功能的sql语句代码：

--1. 创建上述三表的建库、建表代码；

-- 要求使用：主键(医生表.编号，病人表.编号)、外键(病历表.医生编号，

-- 病历表.病人课号)、非空(职称，姓名)、检查(性别),自动编号(ID)

--yishengbiao bingrenbiao binglibiao

alter table yishengbiao add constraint zj primary key (yid)

alter table bingrenbiao add constraint br primary key (bid)

alter table binglibiao

add constraint ysidbrid foreign key(ysid) references yishengbiao(yid)

alter table binglibiao

add constraint ysidbrid1 foreign key(brid) references bingrenbiao(bid)

alter table yishengbiao add check(sex = '女' or sex = '男')

alter table bingrenbiao add check(sex = '女' or sex = '男')

go

--2. 将下列医生信息添加到医生表的代码

-- 编号 姓名性别出生日期职称

--100001 杜医生男1963-5-18 副主任医师

--100002 郭医生女1950-7-26 副主任医师

--100003 刘医生男1973-9-18 医师

select \* from yishengbiao

insert yishengbiao values(100001 ,'杜医生' ,'男' ,1963-5-18 ,'副主任医师')

insert yishengbiao values(100002 ,'郭医生' ,'女' ,1950-7-26 ,'副主任医师')

insert yishengbiao values(100003 ,'刘医生' ,'男' ,1973-9-18 ,'医师')

-- 修改 编号为的医生职称为‘主任医师’

update yishengbiao set zhicheng = '主任医师' where yid = 100002

-- 删除 编号为的医生信息

delete yishengbiao where yid = 100003

--3. 写出创建：医疗表视图(医生编号，姓名，病人姓名，病历)的代码；

drop view ylsbt

create view ylsbt as(

select y.yid , y.name, br.name bm, b.binglimiaoshu --如果去掉br.name后面的bm就会报在视图或函数'ylsbt' 中多次指定了列名'name'。因为两个表中的名字都是叫name,不能够存在相同的name

from yishengbiao y join binglibiao b

on y.yid = b.ysid join bingrenbiao br

on br.bid = b.brid)

select \* from ylsbt

--4. 写出所有病人编号、姓名、病历、以及病人所对应的医生编号的查询语句；

select br.bid,br.name,b.binglimiaoshu,y.yid

from yishengbiao y join binglibiao b

on y.yid = b.ysid join bingrenbiao br

on br.bid = b.brid

--5. 写出创建： 输出某医生（根据医生编号即可）看病人数存储过程

create procedure kysbh @ysid int as(

select COUNT(\*) 人数 from binglibiao where ysid = @ysid)

-- 以及执行过程（要求输入医生姓名的参数，输出病人数）。

exec kysbh 100001

--6. 写出查询年以前出生的医生。

select \* from yishengbiao

update yishengbiao set birthday = '1998-7-26' where yid = 100002

select \* from yishengbiao where YEAR(birthday)<1970

--7. 检索有病人的医生信息。

select \* from yishengbiao b where b.yid in (select ysid from binglibiao)

--8. 创建一个默认，并将其绑定到医生表的成绩职称列上，默认值为“医师”。

select \* from yishengbiao

create default zhicheng AS '医师'

exec sp\_bindefault 'zhicheng', 'yishengbiao.zhicheng'

--这个意思就是在添加表记录的时候会自动添加的默认值，前提是没有手动添加这个字段的前提下

1. **学生选修课程数据库**

--设有学生选修课程数据库，

create database xueshenbiao on(

name=xueshenbiao,

filename='D:\SQLserver\2017 05 25\xueshenbiao.mdf',

filegrowth=1,

size=3)

log on(

name=xueshenbiao\_ldf,

filename='D:\SQLserver\2017 05 25\xueshenbiao.ldf',

filegrowth=1,

size=3)

go

use xueshenbiao

--学生表（学号，姓名，年龄，性别，所在系，地址，出生日期）

drop table xsb

create table xsb(

id int primary key identity(1001,1),

name varchar(10),

nianling int,

sex varchar(2),

xibu varchar(16),

dizhi varchar(30),

birthday datetime)

--课程表（课程号，课程名称，教师姓名）

drop table kcb

create table kcb(

kch int primary key,

kcm varchar(20),

tcname varchar(20))

--选课表（学号，课程号，成绩）

drop table xkb

create table xkb(

id int primary key,

kch int,

scour int

constraint xkb\_xsb foreign key(id) references xsb(id),

constraint xkb\_kcb foreign key(kch) references kcb(kch))

--开始插入记录

select \* from xsb

insert into xsb values('李方式',23,'男','电信系','湖南郴州','1991-02-05'),

('就浪费',78,'女','足球系','北京大同','1956-05-30'),

('陈发送',45,'男','电信系','湖南邵阳','1987-04-10'),

('猪合格',26,'女','财贸系','湖南湘潭','1971-07-12'),

('李部分',32,'男','农林系','东北范德','1987-02-09')

select \* from xkb

insert into xkb values(1001,54,48),(1002,74,58),(1003,68,78),(1004,45,65),(1005,35,45)

select \* from kcb

insert into kcb values(54,'计算机与生活','陈倒是'),

(74,'C#程序设计','李是的'),

(68,'大学语文','刘骄傲'),

(45,'java程序设计','李没'),

(35,'高职实用英语','李也')

--用SQL语言查询下列问题：

--1）李老师所教的课程号、课程名称。

select \* from kcb where tcname like '李%'

select \* from kcb where tcname like '李\_'

--2）年龄大于岁的女学生的学号和姓名。

select \* from xsb

select \* from xsb where sex = '女' and YEAR(GETDATE()) - YEAR(birthday) > 23

--3）“李是的”所选修的全部课程名称。

select kcm from kcb where tcname = '李是的'

--4）所有成绩都在分以上的学生姓名及所在系。

select xsb.name,xsb.xibu,xkb.scour

from xkb join xsb

on xkb.id = xsb.id

where scour > 60

--5）没有选修“C#程序设计”课的学生姓名。

select \* from kcb join xkb

on xkb.kch=kcb.kch join xsb

on xsb.id=xkb.id where kcm != 'C#程序设计'

--6）至少选修两门以上课程的学生姓名、性别。

select name,sex from xsb where id in

(select id from xkb

group by id having count(id)>1) --分组后再次统计是否超过

--7）选修了李老师所讲课程的学生人数。

select count(id) from xkb, kcb --计算id的数就是人数从xkb、kcb中查

where xkb.kch=kcb.kch and tcname='李是的'

--8）没有选修李老师所讲课程的学生。

select \* from xsb where xsb.id in

(select id from xkb where xkb.kch in

(select kch from kcb where tcname != '李是的'))

--如果用上面的id用=的话那么将会报错

--子查询返回的值不止一个。当子查询跟随在=、!=、<、<=、>、>=

--之后，或子查询用作表达式时，这种情况是不允许的

select distinct xsb.id, xsb.name

from xsb, xkb, kcb

where (xkb.id=xsb.id) and (xkb.kch=kcb.kch)

and (tcname<>'李是的')

--9）“操作系统”课程得最高分的学生姓名、性别、所在系。

select top 1\* from xsb,xkb,kcb

where kcb.kch=xkb.kch and xsb.id=xkb.id and kcm like '计%'

order by xkb.scour --asc 顺序\_默认 desc倒叙

select top 1 xsb.id,xsb.name,xsb.xibu

from xsb,xkb,kcb

where xsb.id=xkb.id and kcb.kch=xkb.kch

and kcm like '计%'

order by scour desc