《数据库》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 21计科 | 姓名 | 方浩楠 | 学号 | 2127405048 |
| 实验布置日期 | |  | | 提交  日期 |  | | 成绩 |  |

# 实验十一 安全性控制实验

**目的:掌握Sql-server 的授权机制.**

在school数据库中，完成以下操作：

1）建立新用户 mary , 密码1234

2) 授予 mary 可以访问 School 数据库的权力

图形用户界面, 应用程序

描述已自动生成

3) 以mary 登录 sql-server ,

执行 select \* from student ,记录执行结果，说明原因。

图形用户界面, 应用程序

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

由于 mary 登录名的默认数据库 是 mary 数据库，因此当访问其他数据库时，会出现错误提示信息。

4）将 course 的查询、更改权限授予 mary

图形用户界面, 文本, 应用程序

描述已自动生成

5）把查询 student 表和修改学生学号的权限授予用户 mary,且他能将此权限转授他人。

图形用户界面, 文本, 应用程序

描述已自动生成

6) 把对 course 表的更改权限从mary 收回

图形用户界面, 文本, 应用程序

描述已自动生成

7) 把第5）小题授予mary的权限收回。

图形用户界面, 文本, 应用程序

描述已自动生成

8）mary 只能查询 ‘1001’ 号课程的学生成绩,请问如何授权

先建立视图view\_select

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

然后调用grant

图形用户界面, 文本, 应用程序

描述已自动生成

9）. 对school数据库分别做完整备份、差异备份和日志备份。（参考14.2.3）

10）. 利用9）中的三种备份，进行数据库的恢复（参考14.2.4）

思考: 1 sp\_addlogin , sp\_grantdbaccess 语句的区别.

sp\_addlogin：创建新的 SQL Server 登录，该登录允许用户使用 SQL Server 身份验证连接到 SQL Server 实例。

2 如有200个人需要授权，SQL-SERVER如何简化授权机制。

可以授权给角色来简化授权机制

# 试验十二 索引

**目的：**掌握索引的建立、删除的方法。

在school数据库中完成以下操作。

一 创建索引

1 建 student 的索引

为姓名建立索引，索引名：Ix\_student\_sname

为系科建立索引，索引名：Ix\_student\_sdept

create index ix\_student\_sname on student(sname);  
create index ix\_student\_sdept on student(Sdept);

2 SC 的索引

为课程号建立索引： ix\_sc\_cno

create index ix\_sc\_cno on sc(cno);

3 Course 的索引

为课程名建立唯一性索引 ：Ix\_course\_cname

create unique index ix\_course\_cname on course(cname);

4 如何 SP\_HELP 查看索引刚才建立的索引？

如何在企业管理器中查看索引？

exec *sp\_helpindex* student;  
exec *sp\_helpindex* sc;  
exec *sp\_helpindex* course;

二 删除索引 course 表的索引 IX\_course\_cname

drop index ix\_course\_cname on Course;

三 思考：如何把索引 IX\_student\_sname 修改为唯一性索引？

drop index ix\_student\_sname on Student;  
create unique index ix\_student\_sname on student(sname);

\*四 思考建立索引的目的

1 输入下列存储过程，该程序生成大量数据供测试：

create procedure usp\_makedata as

declare @nCnt int , @sNo varchar(6) , @sname varchar(8)

set @nCnt =12000 --计数器

while @nCnt<999999

begin

set @nCnt = @nCnt + 1

set @sNo = convert(varchar(6) ,@nCnt)

set @sName = '张'+@sno

insert into student (sno,sname,ssex,sage) values ( @sno,@sname,'男',20)

end

return

2 exec usp\_makedata --生成测试数据

3 输入下述测试程序：

create procedure usp\_test as

declare @nCount int ,@data int

set @nCount=0

while @nCount<100

begin

select @data=count(\*) from student where sname <'张3800' or sname>'张8800'

set @nCount =@nCount + 1

end

4 测试

1）建立姓名的索引，查看运行时间(8秒).

create index ix\_student\_sname on student(sname) --建立索引

exec usp\_test

2) 删除姓名索引，查看运行时间（2分11秒），比较与1）的时间长短。

drop index student.ix\_student\_sname --删除索引

exec usp\_test

# 试验十三 存储过程

**目的: 掌握存储过程的概念、编程及使用**

在school数据库中创建以下存储过程。

1 编写一个存储过程 usp\_avgage , 向客户端返回每个系科的学生平均年龄。

系科 平均年龄

JSJ 21

SX 20

。。。

1) 编写存储过程的代码

2）调试、运行该存储过程。

create procedure *usp\_avgage*as  
begin  
 select Sdept,*avg*(Sage) as '平均年龄'  
 from Student  
 group by Sdept  
end;

2编写一个存储过程 usp\_sdept, 传入一个系科代码，返回该系的平均年龄，人数

create procedure *usp\_sdept* @sdept varchar(15)  
as  
begin  
 select *avg*(sage),*count*(*\**)  
 from Student  
 where Sdept = @sdept;  
end;

3 编写存储过程 usp\_updateGrade , 传入参数为课程号,处理逻辑:

对传入的这门课,进行如下处理:

如某学生该门课成绩>80 , 则加 2 分

如某学生该门课成绩>60 , 则加 1 分

如某学生该门课成绩<=60 ,则减 1分

并且返回此门课的每个学生的最新成绩: 学号 成绩.

create procedure *usp\_updateGrade* @cno char(4)  
as  
 begin  
 update sc  
 set Grade = case  
 when grade>80 then Grade + 2  
 when grade>60 then Grade + 1  
 else Grade - 1  
 end  
 where cno = @cno;  
  
 select sno,grade  
 from sc  
 where cno = @cno;  
 end;

4 编写存储过程 usp\_g1 , 传入参数课程号，对该门课程进行如下处理，低于平均分5分不加分，低于平均分0-5分的加1分，高于平均分0-5加1分，高于平均分5分的加2分。

create procedure *usp\_g1* @cno char(4)  
as  
 begin  
 declare @avg\_grade decimal(12,1)  
 select @avg\_grade = *avg*(grade)  
 from sc  
 where cno = @cno;  
  
 update sc  
 set grade = case  
 when Grade < @avg\_grade - 5 then grade  
 when Grade < @avg\_grade then Grade + 1  
 when Grade < @avg\_grade then Grade + 1  
 when Grade < @avg\_grade then Grade + 2  
 end  
 where cno = @cno  
 end;

5 编写存储过程 usp\_comp\_age , 比较0001，0002学生的年龄的高低，输出： XXXX学生的年龄大

注意： XXXX为学生的姓名

create procedure *usp\_comp\_age* @sno1 char(4),  
 @sno2 char(4)  
as  
 begin  
 declare @age1 int,@age2 int  
 select @age1 = Sage  
 from Student  
 where sno = @sno1;  
  
 select @age2 = Sage  
 from Student  
 where sno = @sno2;  
  
 if @age1 > @age2  
 select sname + '学生的年龄大' as 结果  
 from Student  
 where sno = @sno1;  
 else if @age2 > @age1  
 select sname + '学生的年龄大' as 结果  
 from Student  
 where sno = @sno2;  
 else  
 select '两个学生的年龄相同' as '结果';  
 end;

6 编写存储过程 usp\_comp , 比较1001，1002课程的平均分的高低，输出： XXXX课程的平均分高

CREATE PROCEDURE *usp\_comp* @Cno1 char(4),  
 @Cno2 char(4)  
AS  
BEGIN  
 DECLARE @AvgGrade1 decimal(12, 1), @AvgGrade2 decimal(12, 1);  
 SELECT @AvgGrade1 = *AVG*(Grade)  
 FROM SC  
 WHERE Cno = @Cno1;  
  
 SELECT @AvgGrade2 = *AVG*(Grade)  
 FROM SC  
 WHERE Cno = @Cno2;  
  
 IF @AvgGrade1 > @AvgGrade2  
 SELECT Cname + '课程的平均分高' AS '结果'  
 FROM Course  
 WHERE Cno = @Cno1;  
 ELSE IF @AvgGrade1 < @AvgGrade2  
 SELECT Cname + '课程的平均分高' AS '结果'  
 FROM Course  
 WHERE Cno = @Cno2;  
 ELSE  
 SELECT '两门课程的平均分相同' AS '结果';  
END;

7 编写存储过程 usp\_comp\_age1 , 比较两个学生的年龄的高低，两个学生的学号有参数输入，最后输出： XXXX学生的年龄大。

注意： XXXX为学生的姓名

create procedure *usp\_comp\_age1* @sno1 char(4),  
 @sno2 char(4)  
as  
 begin  
 declare @age1 int,@age2 int  
 select @age1 = Sage  
 from Student  
 where sno = @sno1;  
  
 select @age2 = Sage  
 from Student  
 where sno = @sno2;  
  
 if @age1 > @age2  
 select sname + '学生的年龄大' as 结果  
 from Student  
 where sno = @sno1;  
 else if @age2 > @age1  
 select sname + '学生的年龄大' as 结果  
 from Student  
 where sno = @sno2;  
 else  
 select '两个学生的年龄相同' as '结果';  
 end;

8 利用第8题的存储过程，判断0002，0003学生的年龄大小。

exec *usp\_comp\_age1* '0002','0003';

9 编写存储过程 usp\_comp1 , 传入两参数，课程号1，课程号2；比较这两门课的平均分的高低，输出： XXXX课程的平均分高

CREATE PROCEDURE *usp\_comp1* @Cno1 char(4),  
 @Cno2 char(4)  
AS  
BEGIN  
 DECLARE @AvgGrade1 decimal(12, 1), @AvgGrade2 decimal(12, 1);  
 SELECT @AvgGrade1 = *AVG*(Grade)  
 FROM SC  
 WHERE Cno = @Cno1;  
  
 SELECT @AvgGrade2 = *AVG*(Grade)  
 FROM SC  
 WHERE Cno = @Cno2;  
  
 IF @AvgGrade1 > @AvgGrade2  
 SELECT Cname + '课程的平均分高' AS '结果'  
 FROM Course  
 WHERE Cno = @Cno1;  
 ELSE IF @AvgGrade1 < @AvgGrade2  
 SELECT Cname + '课程的平均分高' AS '结果'  
 FROM Course  
 WHERE Cno = @Cno2;  
 ELSE  
 SELECT '两门课程的平均分相同' AS '结果';  
END;

10 编写存储过程 usp\_comp\_age2 , 比较两个学生的年龄的高低，两个学生的学号有参数输入，最后把年龄大的学生的姓名、性别返回客户端。

create procedure *usp\_comp\_age2* @sno1 char(4),  
 @sno2 char(4)  
as  
 begin  
 declare @age1 int,@age2 int  
 select @age1 = Sage  
 from Student  
 where sno = @sno1;  
  
 select @age2 = Sage  
 from Student  
 where sno = @sno2;  
  
 if @age1 > @age2  
 select sname,Ssex  
 from Student  
 where sno = @sno1;  
 else if @age2 > @age1  
 select sname,Ssex  
 from Student  
 where sno = @sno2;  
 else  
 select '两个学生的年龄相同' as '结果';  
 end;

11 编写存储过程 usp\_comp2 , 传入两参数，课程号1，课程号2；比较这两门课的平均分的高低，最后把平均分高的课程的课程名返回客户端。

CREATE PROCEDURE *usp\_comp2* @Cno1 char(4),  
 @Cno2 char(4)  
AS  
BEGIN  
 DECLARE @AvgGrade1 decimal(12, 1), @AvgGrade2 decimal(12, 1);  
 SELECT @AvgGrade1 = *AVG*(Grade)  
 FROM SC  
 WHERE Cno = @Cno1;  
  
 SELECT @AvgGrade2 = *AVG*(Grade)  
 FROM SC  
 WHERE Cno = @Cno2;  
  
 IF @AvgGrade1 > @AvgGrade2  
 SELECT Cname  
 FROM Course  
 WHERE Cno = @Cno1;  
 ELSE IF @AvgGrade1 < @AvgGrade2  
 SELECT Cname  
 FROM Course  
 WHERE Cno = @Cno2;  
 ELSE  
 SELECT '两门课程的平均分相同' AS '结果';  
END;

12 编写存储过程 usp\_t1，传入参数为学号,把该学号的课程1001的成绩减到58分。每次只能减1分，用循环完成。

create procedure *usp\_t1* @sno char(4)  
as  
 begin  
 declare @grade decimal(12,1);  
 select grade  
 from sc  
 where Sno = @sno and cno = '1001';  
  
 while @grade > 58  
 begin  
 set @grade = @grade - 1;  
 update SC  
 set Grade = @grade  
 where sno = @sno and cno = '1001';  
 end  
 end;

13 编写存储过程 usp\_t2，传入参数为系科，把该系科的学生每次加一岁，只要该系科有一个人的年龄达到28岁，即停止循环。每次只能减加1岁分，用循环完成。

create procedure *usp\_t2* @sdept varchar(15)  
as  
 begin  
 declare @age int;  
 set @age = (  
 select *max*(Sage)  
 from Student  
 where Sdept = @sdept  
 );  
  
 while @age < 28  
 begin  
 update Student  
 set sage = Sage + 1  
 where Sdept = @sdept;  
  
 set @age = (  
 select *max*(Sage)  
 from Student  
 where Sdept = @sdept  
 );  
 end  
 end;

15 编写存储过程 usp\_disp , 传入参数为课程号,处理逻辑: 返回每个学生的成绩等级。

成绩>=90 为优, 成绩>=80为良,成绩>=70 为中,成绩>=60为及格 ,成绩<=60为不及格。

返回结果如下：

学号 课程号 成绩 等第

0001 1001 91 优

0001 1002 78 中

……………..

create procedure *usp\_disp* @cno char(4)  
as  
 begin  
 select sno,cno,grade,  
 case  
 when grade >= 90 then '优'  
 when grade >= 80 then '良'  
 when grade >= 70 then '中'  
 when grade >= 60 then '及格'  
 else '不及格'  
 end as '等第'  
 from SC  
 where cno = @cno  
 end;

16 编写一个存储过程，传入参数为学号，执行后，把该学号的学生按如下格式输出成绩：

（注意：只有一行）

学号 姓名 1001课程 1002课程 1003 课程 平均分

create procedure *usp\_display\_grade* @sno char(6)  
as  
begin  
 select s.sno,s.sname,  
 (select sc.grade from sc where cno = '1001') as '1001课程',  
 (select sc.grade from sc where cno = '1002') as '1002课程',  
 (select sc.grade from sc where cno = '1003') as '1003课程',  
 *avg*(grade)  
 from Student as s join sc on s.sno = sc.Sno  
 where SC.Sno = @sno  
 group by s.sno,s.sname  
end;

17 编写一个存储过程，传入参数为 系科，执行后，把该系科的学生按如下格式输出学生成绩：

学号 姓名 1001 课程 1002课程 1003 课程 平均分

create procedure *usp\_display\_grade2* @sdept varchar(15)  
as  
begin  
 select s.sno,s.sname,  
 (select sc.grade from sc where cno = '1001') as '1001课程',  
 (select sc.grade from sc where cno = '1002') as '1002课程',  
 (select sc.grade from sc where cno = '1003') as '1003课程',  
 *avg*(grade)  
 from Student as s join sc on s.sno = sc.Sno  
 where Sdept = @sdept  
 group by s.sno,s.sname  
end;

18 编写存储过程，统计男女生1001，1002，1003各自的选修人数，输出格式如下：

性别 1001人数 1002人数 1003人数 小计

男 3 5 2 10

女 2 4 1 7

合计 5 9 3 17

（数据为示意数据）

create procedure *usp\_count\_gender\_courses*as  
begin  
 select '男' as '性别',  
 *sum*(case when s.ssex = '男' and sc.cno = '1001' then 1 else 0 end) as '1001人数',  
 *sum*(case when s.ssex = '男' and sc.cno = '1002' then 1 else 0 end) as '1002人数',  
 *sum*(case when s.ssex = '男' and sc.cno = '1001' then 1 else 0 end) as '1003人数',  
 *sum*(case when s.ssex = '男' then 1 else 0 end) as '小计'  
 from Student as s join sc on s.sno = sc.sno  
 union all  
 select '女' as '性别',  
 *sum*(case when s.ssex = '女' and sc.cno = '1001' then 1 else 0 end) as '1001人数',  
 *sum*(case when s.ssex = '女' and sc.cno = '1002' then 1 else 0 end) as '1002人数',  
 *sum*(case when s.ssex = '女' and sc.cno = '1001' then 1 else 0 end) as '1003人数',  
 *sum*(case when s.ssex = '女' then 1 else 0 end) as '小计'  
 from Student as s join sc on s.sno = sc.sno  
 union all  
 select '合计' as '性别',  
 *sum*(case when sc.cno = '1001' then 1 else 0 end) as '1001人数',  
 *sum*(case when sc.cno = '1002' then 1 else 0 end) as '1002人数',  
 *sum*(case when sc.cno = '1003' then 1 else 0 end) as '1003人数',  
 *count*(*\**) as '小计'  
 from SC  
end

19 编写一个存储过程，利用存储过程的参数返回数据库服务器上的日期时间。

create procedure *usp\_getDateTime* @currentDateTime datetime output  
as  
begin  
 set @currentDateTime = *getDate*();  
end

# 试验十四 触发器

**目的: 了解触发器的机制及编程设计、使用**

**在school数据库中完成以下操作：**

一 建立学生表的触发器 usp\_addstudent，当增加学生时，SX系的学生不能超过30岁。

1 写出触发器

create trigger usp\_addstudent on Student  
after insert  
 as  
 begin  
 if *exists*(select 1 from inserted where Sdept = 'SX' and sage >= 30)  
 begin  
 raiserror('数学系的学生不能超过三十岁',11,1)  
 rollback  
 end  
 end;

2 执行下列语句块：

begin tran

insert into student (sno,sname,ssex,sage,sdept)

values (‘0701’,’刘欢’,’男’,26,’SX’)

if @@error=0

commit

else

rollback

end

观察该学生是否加入到 student

能加入到student

3执行下列语句块：

begin tran

insert into student (sno,sname,ssex,sage,sdept) values (‘0702’,’赵欢’,’男’,31,’SX’)

if @@error=0

commit

else

rollback

end

观察该学生是否加入到 student

不能加入到student

二 实现下列触发器

1 不能删除年龄大于25岁的学生记录。

create trigger usp\_nodelstudent on Student  
for delete  
 as  
 begin  
 if *exists*(select 1 from inserted where sage>25)  
 begin  
 raiserror('不能删除年龄大于大于25岁的学生记录',11,2)  
 rollback  
 end  
 end;

2 建立触发器 usp\_delcourse , 使课程表中1001，1002，1003 三门课不会被删除。

注意如何调试。

create trigger usp\_delcourse on Course  
for delete  
 as  
 begin  
 if *exists*(select 1 from deleted where cno in ('1001','1002','1003'))  
 begin  
 raiserror('课程表中1001，1002，1003 三门课不会被删除',11,3)  
 rollback  
 end  
 end;

3 对学生表建立一触发器，使更改后的年龄只能比原值大

create trigger usp\_updateage on Student  
after update  
 as  
 begin  
 if *exists*(  
 select 1  
 from inserted as i join deleted as d on i.Sno = d.Sno  
 where i.Sage < d.Sage  
 )  
 begin  
 raiserror('更改后的年龄只能比原值大',11,4);  
 rollback  
 end  
 end;

4对sc表建立触发器，使‘JSJ’系的学生不可选择 ‘1004’号课程

create trigger usp\_sccheck on sc  
after insert  
 as  
 begin  
 if *exists*(  
 select 1  
 from inserted  
 where Sdept = 'JSJ' and cno = '1004'  
 )  
 begin  
 raiserror('‘JSJ’系的学生不可选择 ‘1004’号课程',11,5)  
 rollback  
 end  
 end;

5 对表 course 建触发器，实现级联删除的功能，但某课选修人数大于3则不能删除。

(先删除 sc 表对course 的外码)

\*三 建立一个触发器，使对sc表成绩的修改自动记录修改日志。

日志文件表(tablog)记录如下：

用户名 学号 课程号 原成绩 修改后成绩 更改日期

create table tablog (  
 用户名 varchar(50),  
 学号 varchar(50),  
 课程号 varchar(50),  
 原成绩 int,  
 修改后成绩 int,  
 更改日期 datetime  
);  
  
create trigger utr\_sclog on sc  
after update  
 as  
 begin  
 insert into tablog  
 select *suser\_name*(),d.sno,d.cno,d.grade,i.grade,*getdate*()  
 from inserted as i join deleted as d on i.sno = d.sno and i.cno = d.cno  
 end;

四 在School数据库中建立一个试验用的发票表bill，然后为发票bill建立触发器 utr\_money ,实现当输入单价和数量后，自动填写金额，即发票金额不输入，由单价、数量相乘后自动填写到金额中。

Create table bill(

billID char(8), --发票编号

date datetime, --开票日期

product char(10), --产品编号

price int , --单价

qty int , --数量

charge int , --金额

primary key (billid) )

create table bill (  
 billid char(8), *--发票编号* date datetime, *--开票日期* product char(10), *--产品编号* price int, *--单价* qty int, *--数量* charge int, *--金额* primary key (billid)  
);  
  
create trigger utr\_money on bill  
after insert ,update  
 as  
 begin  
 update bill  
 set charge = bill.price \* bill.qty  
 from bill join inserted as i on bill.billid = i.billid  
 end;