# 实验 2 SQL Server 数据库的管理

4. 启动查询分析器，在查询分析器中使用 Transact-SQL 语句 CREATE DATABASE 创建 studb 数据库。然后通过系统存储过程 sp_helpdb 查看系统中的数据库信息

```
CREATE DATABASE studb

sp_helpdb
```

5. 在查询分析器中使用 Transact-SQL 语句 ALTER DATABASE 修改 studb 数据库的设置，指定数据文件大小为 5MB，最大文件大小为 20MB，自动递增大小为 1MB。

```
ALTER DATABASE studb
MODIFY FILE
(
NAME=studb,
SIZE=5MB,
MAXSIZE=20MB,
FILEGROWTH=1MB
)
```

7. 使用企业管理器将 studb 数据库的名称更改为 student_db 。

```
ALTER DATABASE studb
MODIFY NAME=student_db
```

8. 使用 Transact-SQL 语句 DROP DATABASE 删除 student_db 数据库。

```
DROP DATABASE student_db
```

# 实验 3 SQL Server 数据表的管理

5. 使用 Transact-SQL 语句 CREATE TABLE 在 studentsdb 数据库中创建 grade 表。

```
CREATE TABLE grade
( 学号 char(4),
课程编号 char(4),
分数 decimal(5)
)
```

8. 使用 Transact_SQL 语句 INSERT INTO...VALUES向 studentsdb 数据库的 grade 表插入以下数据：

| 学号 | 课程编号 | 分数 |
| --- | --- | --- |
| 0004 | 0001 | 80 |

```
USE studentsdb
GO
INSERT INTO grade
VALUES('0004','0001','80')
```

9. 使用 Transact_SQL 语句 ALTER TABLE修改 curriculum 表的“ 课程编号 ”列，使之为非空。

```
ALTER TABLE curriculum
ALTER COLUMN课程编号 char(4)NOT NULL
GO
```

10. 使用 Transact_SQL 语句 ALTER TABLE修改 grade 表的“ 分数 ”列，使其数据类型为 real 。

```
ALTER TABLE grade
ALTER COLUMN分数 real
```

**11.** 分别使用企业管理器和 Transact_SQL 语句 DELETE删除 studentsdb 数据库的 grade 表中学号为 '0004' 的成绩记录。

```
DELETE grade WHERE学号 ='0004'
```

13. 使用 Transact_SQL 语句 UPDATE修改 studentsdb 数据库的 grade 表中学号为 '0003' 、课程编号为 '0005' 、分数为 90 的成绩记录。

```
UPDATE grade SET分数 =90
WHERE 学号 ='0003' and 课程编号 ='0005'
```

14. 使用 Transact_SQL 语句 ALTER...ADD为 studentsdb 数据库的 grade 表添加一个名为“ 备注 ”的数据列，其数据类型为 VARCHAR(20)。

```
ALTER TABLE grade ADD 备注 VARCHAR(20) NULL
```

15. 分别使用企业管理器和 Transact_SQL 语句 DROPTABLE删除 studentsdb 数据库中的 grade 表。

DROP TABLE grade

# 实验 4　数据查询

1.　在 studentsdb 数据库中，使用下列　SQL语句将输出什么？

（1）15

（2）刘卫

张卫

马东

钱达

东方

郭文

肖海

张明

（3）KELLY

（4）kellykellykelly

（5）9.4868329805051381

9.3273790530888157

9.2736184954957039

9.4339811320566032

（6）2　　3　　8

（7）2011　10　　10

2.　在 studentsdb　数据库中使用　SELECT语句进行基本查询。

(1) SELECT　姓名 , 学号 , 姓名 , 出生日期　from student_info

(2) SELECT　姓名 , 家庭住址　from student_info

where　学号 =0002

(3)　刘卫平　　0001

张卫民　　0002

马东　　0003

钱达理　　0004

东方牧　　0005

3. 使用　SELECT语句进行条件查询

(1)　SELECT学号 ,分数　from grade

where　分数 <'90' and　分数 >'80'

(2) SELECT avg( 分数 ) from grade

where　学号 =0003

(3) SELECT课程编号 , count( 课程编号 ) from grade

group by　课程编号

(4) SELECT姓名 ,出生日期　from student_info

order by　出生日期　asc

(5) SELECT学号 ,姓名　FROM student_info WHERE 姓名　LIKE 张%'

4.　嵌套查询

(4)　SELECT课程编号 , 分数 FROM grade

where 学号 =0001 and　分数 >(SELECT max(分数 ) from grade

where 学号 =0002)

5. 多表查询

(3) SELECT s.学号 ,s. 姓名 ,c. 课程名称 ,g. 分数

FROM student_info s,grade g,curriculum c

where s. 学号 =g. 学号 and s. 性别 =' 男 ' and c. 课程编号 =g. 课程编号

(4) select 学号 ,max( 分数 )

from grade

group by 学号

(5) SELECT s.学号 , 姓名 ,sum(g. 分数 ) FROM student_info s left outer join grade g on s. 学号 =g. 学号 group by s. 学号 , 姓名

(6) insert into grade( 学号 ,课程编号 ,分数 )

values('0004','0006','76')

SELECT c. 课程编号 , 课程名称 ,count(g. 学号 ) FROM curriculum c right outer join grade g on g. 课程编号 =c. 课程编号 group by c. 课程编号 , 课程名称

6. 使用 UNION 运算符将 student_info 表中姓 “ 张 ” 的学生的学号、姓名与 curriculum 表的课程编号、课程名称返回在一个表中，且列名为 u_编号、 u_名称，如图 1-8 所示。

select 学号 u_ 编号 , 姓名 u_ 名称 from student_info where 姓名 like ' 张 %'

union

select 课程编号 , 课程名称 from curriculum

7. 数据更新

(4) delete from totalgrade

where 总成绩 =null

# 实验 5 索引和视图

1. 分别使用企业管理器和 Transact-SQL 语句为 studentsdb 数据库的 student_info 表格和 curriculum 表创建主键索引。

ALTER table student_info

ALTER column 学号 char(4) not null

go

ALTER table student_info add constraint PK_student_info primary key( 学号 )

ALTER table curriculum

ALTER column 课程编号 char(4) not null

go

ALTER table curriculum add constraint PK_curriculum primary key( 课程编号 )

5. .分别使用企业管理器和系统存储过程 sp_helpindex 查看 grade 表和 student_info 表上的索引信息。

use studentsdb

go

exec sp_helpindex grade

go

```
use studentsdb
go
exec sp_helpindex student_info
go
```

16. 在 studentsdb 数据库中，使用 Transact-SQL 语句 CREATE VIEW 建立一个名为 v_stu_c 的视图，显示学生的学号、姓名、所学课程的课程编号，并利用视图查询学号为 0003 的学生情况

```
create view v_stu_c
as
select student_info.学号 ,student_info.姓名 ,curriculum.课程名称 ,grade.课程编号
from student_info inner join grade on student_info.  学号 =grade.学号
        inner join curriculum on grade.课程编号 =curriculum.课程编号

select *
from v_stu_c
where  学号 ='0003'
```

17. 基于 student_info 表、curriculum 表和 grade 表，建立一个名为 v_stu_g 的视图，视图中具有所有学生的学号、 姓名、课程名称、分数。使用视图 v_stu_g 查询学号为 0001 的学生的所有课程和成绩

```
create view v_stu_g
as
select student_info.学号 ,student_info.姓名 ,curriculum.课程名称 ,grade.分数
from student_info inner join grade on student_info.  学号 =grade.学号
        inner join curriculum on grade.课程编号 =curriculum.课程编号

select *
from v_stu_g
where  学号 ='0001'
```

18. 分别使用企业管理器和 Transact-SQL 语句修改视图 v_stu_c，使之显示学号、姓名、每个学生所学课程数目。

```
alter view v_stu_c( 学号 ,姓名 ,课程数目 )
as select student_info.学号 ,student_info.姓名 ,count(grade.课程编号 ) as 课程数目
from student_info,grade
where student_info.学号 =grade.学号
group by student_info.学号 ,student_info.姓名
```

21. 利用视图 v_stu_i 为 student_info 表添加一行数据：学号为 0015、姓名为陈婷、性别为女。

```
insert into v_stu_info
values('0015',' 陈婷 ',' 女 ')
```

22. 利用视图 v_stu_i 删除学号为 0015 的学生记录。

```
delete from    v_stu_info
  where  学号 ='0015'
```

23. 利用视图 v_stu_g 修改姓名为刘卫平的学生的高等数学的分数为 84。

```
update v_stu_g
set  分数 =84
where  姓名 ='刘卫平 ' and  课程名称  ='高等数学 '
```

24. 使用 Transact-SQL 语句 DROP VIEW删除视图 v_stu_c 和 v_stu_g 。

```
drop view v_stu_c,v_stu_g
```

# 实验 6 数据完整性

1．为 sutdentsdb 数据库创建一个规则，限制所输入的数据为 7 位 0-9 的数字。

（1）SELECT*INTO stu_phone FROM student_info

ALTER TABLE stu_phone ADD电话号码 CHAR(7)NULL

（2）CREATE RULE phone_rule

AS

@phone LIKE '[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]'

（3）sp_bindrule phone_rule,'stu_phone. 电话号码 '

（4）出错原因：与该列所绑定的规则发生冲突；

需要修改：改 ‘1234yyyy 为 ‘1234567 ’

phone_rule 规则不能对其他操作（如 DELETE）进行规则检查

3. 创建一个规则 stusex_rule，将其绑定到 stu_phone 表的“ 性别 ”列上，保证输入的性别值只能是“ 男 ”或“ 女 ” 。

```
CREATE RULE stusex_rule
AS @sex in('男','女')
sp_bindrule stusex_rule,'stu_phone. 性别 '
```

4．使用系统存储过程 sp_help 查询 stusex_rule 规则列表，使用 sp_helptext 查询 stusex_rule 规则的文本，使用 sp_rename 将 stusex_rule 规则更名为 stu_s_rule。

```
sp_help stusex_rule
sp_helptext stusex_rule
sp_rename stusex_rule,stu_s_rule
```

5．删除 stu_s_rule 规则。

```
Sp_unbindrule    'stu_phone. 性别 '
DROP RULE stu_s_rule
```

6．在 stuedentdb 数据库中，建立日期、货币和字符等数据类型的默认对象。

(1) CREATE DEFAULT df_date

AS '2006-4-12'

GO

CREATE DEFAULT df_char

AS 'unknown'

GO
CREATE DEFAULT df_money
AS $100
GO
（2）CREATE TABLE stu_fee
(学号 char(10)NOT NULL,
姓名 char(8)NOT NULL,
学费 money,
交费日期 datetime,
电话号码 char(7))
（3）Sp_bindefault df_money,'stu_fee. 学费 '
GO
Sp_bindefault df_date,'stu_fee. 交费日期 '
GO
Sp_bindefault df_char,'stu_fee. 电话号码 '
GO
（4）INSERT INTO stu_fee(学号 ,姓名 ) values('0001',' 刘卫平 ')
INSERT INTO stu_fee(学号 ,姓名 ,学费 ) values('0001',' 张卫民 ',$120)
INSERT INTO stu_fee(学号 ,姓名 ,学费 ,交费日期 )
VALUES('0001','马东 ',$110,'2006-5-12')
（5）sp_unbindefault 'stu_fee. 电话号码 '
DROP DEFAULT df_char
GO
sp_unbindefault 'stu_fee. 交费日期 '
DROP DEFAULT df_date
GO
sp_unbindefault 'stu_fee. 学费 '
DROP DEFAULT df_money
GO

8.为 student_info 表添加一列，命名为“ 院系 ” ，创建一个默认对象 stu_d_df，将其绑定到 student_info 表的“ 院系 ” 列上，时期默认值为“ 信息院 ” ，对 student_info 表进行插入操作，操作完成后，删除该默认对象。分别使用企业管理器和查询分析器实现。

ALTER TABLE student_info ADD院系 CHAR(12)NULL
CREATE DEFAULT stu_d_df
AS '信息院 '
sp_bindefault stu_d_df,'student_info. 院系 '
INSERT student_info(学号 ,姓名 ,院系 ) values('0001',' 刘卫平 ', ' 土木工程 ')
sp_unbindefault 'student_info. 院系 '
DROP DEFAULT stu_d_df

9.在 studentsdb 数据库中用 CREATE TABLE 语句创建表 stu_con，并同时创建约束。

(1) CREATE TABLE stu_con
(学号 char(4)
CONSTRAINT pk_sid PRIMARY KEY(学号 ),
姓名 char(8)

CONSTRAINT uk_name UNIQUE,
性别 char(2)
CONSTRAINT df_sex DEFAULT '男',
出生日期 datetime
CONSTRAINT ck_beday CHECK(出生日期 >'1988-1-1'),
家庭住址 varchar(50))

(2) INSERT stu_con(学号 ,姓名 ,出生日期 ) VALUES('0009',张小东 ','1989-4-6')
INSERT stu_con(学号 ,姓名 ,性别 ,出生日期 ) VALUES('0010',李梅 ',' 女','1983-8-5')
INSERT stu_con(学号 ,姓名 ,出生日期 ) VALUES('0011',王强 ','1988-9-10')
INSERT stu_con(学号 ,姓名 ,出生日期 ) VALUES('0012',王强 ','1989-6-3')
结果分析：第一、三条命令顺利执行，第二、四条命令不能执行。
第二行语句 INSERT 语句与 COLUMN CHECK 约束 'ck_beday' 冲突。该冲突发生于数据库 'studentsdb' ，表 'stu_con', column ' 出生日期 '。
第四条语句违反了 UNIQUE KEY 约束 'uk_name' 。不能在对象 'stu_con' 中插入重复键。

（3） ALTER TABLE stu_con
DROP CONSTRAINT pk_sid,uk_name,df_sex,ck_beday

11. 在查询分析器中，为 studentsdb 数据库的 grade 表添加外键约束（ FOREIGN KEY），要求将“ 学号 ”设置为外键， 参照表为 student_info , 外键名为 ufk_sid。使用系统存储过程 sp_help 查看 grade 表的外键信息。

```
ALTER TABLE student_info
ADD PRIMARY KEY(学号 )
GO
ALTER TABLE grade
ADD CONSTRAINT ufk_sid FOREIGN KEY(学号 ) REFERENCES student_info(学号 )
GO
sp_help grade
```

在 grade 表中插入表 1-2 所示记录，观察 SQL Server会做何处理，为什么？如何解决所产生的问题 ？

答：学号 0100 显示为 100；课程编号 0001 显示为 1.

```
ALTER TABLE grade
DROP CONSTRAINT ufk_sid
```

# 实验 7 Transact-SQL 程序设计

1.结果显示：张明华
显示的仅为第二个姓张的记录

```
2. DECLARE @grademax int, @grademin int, @gradesum int
   SELECT @grademax=max(分数 ),@grademin=min( 分数 ),@gradesum=sum(分数 )
   FROM grade
   SELECT @grademax, @grademin, @gradesum
3. DECLARE @row int
   SET @row=(SELECT COUNT(*)FROM grade)
   SELECT @row
4. DECLARE @intCId int, @intErrorCode int
INSERT INTO curriculum(课程编号 ,课程名称 ,学分 )
```

VALUES('0006','VB程序设计 ',2)
SELECT @intCId= @@identity,@intErrorCode= @@error
SELECT @intCId, @intErrorCode
第一次显示： NULL 0
第二次显示： NULL 0
curriculum 表中数据的变化：
第一次： 0006 VB 程序设计 2
第二次： 0006 VB 程序设计 2
5. 在 studentsdb 数据库的 student_info 表中，以 “性别 ”为分组条件，分别统计男生和女生人数。

```
SELECT COUNT(性别 ') FROM student_info
GROUP BY性别
```

6. 在 grade 表中，使用适当函数找出 “高等数学 ”课程的最高分、最低分和平均分。

```
SELECT MAX(分数 )AS 最高分 ,MIN( 分数 )AS 最低分 ,AVG(分数 )AS 平均分
FROM grade
WHERE课程编号 =(SELECT 课程编号 FROM curriculum
        WHERE 课程名称 =' 高等数学 ')
```

7. 定义一个 datetime 型局部变量 @student，以存储当前日期。计算 student_info 表中的学生的年龄，并显示学生的姓名、年龄。 在以下代码的划线部分填入适当内容，以实现上述功能。

```
DECLARE @student datetime
SET @student=getdate()
SELECT姓名 ,year(@student)-year( 出生日期 ) AS 年龄
FROM student_info
```

8. 运行代码，写出运行结果。

运行结果为： 8 233 225

9. 在局部变量 @stu_id 中存储了学号值。编写代码查询学号为 0001 的学生的各科平均成绩，如果平均分 >=60 则显示 “你的成绩及格了，恭喜你！！ ”，否则显示 “你的成绩不及格 ”。

```
 IF ((SELECT AVG( 分数 ) FROM grade where 学号 ='0001')<60)
 PRINT ' 你的成绩不及格 '
ELSE
 PRINT ' 你的成绩及格了 , 恭喜你！！'
```

10. 运行代码段，写出运行的结果。

```
@counter 的值现在为： 1
@counter 的值现在为： 2
@counter 的值现在为： 3
@counter 的值现在为： 4
@counter 的值现在为： 5
@counter 的值现在为： 6
@counter 的值现在为： 7
@counter 的值现在为： 8
@counter 的值现在为： 9
```

11. 查询 grade 表。如果分数大于等于 90，显示 A；如果分数大于等于 80 小于 90，显示 B；如果分数大于等于 70 小于 80，显示 C；如果分数大于等于 60 小于 70，显示 D；其他显示 E。在以下代码的划线部分填入适当内容完成上述功能。

```
SELECT学号 , 分数 , 等级 =
CASE
WHEN分数 >=90 THEN 'A'
WHEN分数 >=80 AND 分数 <90 THEN 'B'
WHEN分数 >=70 AND 分数 <80 THEN 'C'
WHEN分数 >=60 AND 分数 <70 THEN 'D'
ELSE 'E'
END
FROM grade
```

12. 计算 grade 表的分数列的平均值。如果小于 80，则分数增加其值的 5%；如果分数的最高值超过 95，则终止该操作。在以下代码划线处填入适当的内容以完成上述功能。

```
WHILE (SELECT AVG(分数 ) FROM grade)<80
BEGIN
UPDATE grade
SET 分数 =分数 *1.05
If (SELECT MAX( 分数 ) FROM grade)>95
  BREAK
ELSE
  BREAK
END
```

13. 编写代码计算并显示 @ n = 1+2+3+...+20。

```
DECLARE @n int,@a int
SELECT @n=1,@a=1
WHILE @a<=20
BEGIN
  SET @n=@n+@a
  SET @a=@a+1
END
PRINT '@ n = 1+2+3+...+20'
PRINT @n
```

14. 编写代码计算并显示 1~100 之间的所有完全平方数。例如, $81 = 9^2$，则称 81 为完全平方数。

```
DECLARE @a int
SELECT @a=1
WHILE @a<=10
BEGIN
IF 100>=@a*@a
  PRINT CONVERT(char(3),@a*@a)+'  是完全平方数 '
  SET @a=@a+1
END
```

15. 计算 1~100 以内的所有的素数。

```
DECLARE @n int,@i int
SET @n=2
PRINT ' 素数是： '
WHILE @n<=100
BEGIN
SET @i=2
  WHILE @i<SQRT(@n)
  BEGIN
  IF(@n%@i=0)BREAK
  SET @i=@i+1
  END
IF(@i>SQRT(@n))PRINT @n
SET @n=@n+1
END
```

16. 在 studentsdb 数据库中，使用游标查询数据。

（1）DECLARE stu_cursor CURSOR
GLOBAL SCROLL DYNAMIC
FOR
SELECT * FROM student_info
WHERE 性别 =' 男'

（2）OPEN stu_cursor

（3）FETCH NEXT FROM stu_cursor
WHILE @@fetch_status=0
BEGIN
FETCH NEXT FROM stu_cursor
END

（4）CLOSE stu_cursor

17. 使用游标修改数据。

(1)OPEN stu_cursor

(2)UPDATE student_info
SET 出生日期 =Dateadd(Year,1, 出生日期 )
WHERE 姓名 Like ' 马%' AND 性别 =' 男'

(3)CLOSE stu_cursor

18. 声明游标变量 @stu_c，使之关联 stu_cursor 游标，利用 stu_c 查询年龄在 6~9 月份出生的学生信息。

SELECT * FROM student_info
WHERE DATEPART(MONTH, 出生日期 )<=9 AND DATEPART(MONTH, 出生日期 )>=6

19. 使用系统存储过程 sp_cursor_list 显示在当前作用域内的游标及其属性。

```
OPEN stu_cursor
DECLARE @report CURSOR
EXEC sp_cursor_list @cursor_return=@report OUTPUT,
@cursor_scope=2
CLOSE stu_cursor
FETCH NEXT FROM @report
```

# 实验 8　存储过程和触发器

1.　执行代码之后，　studentsdb 数据库的存储过程中出现　lletters_print ，表明命令已经成功执行。

EXECUTE letter_print

2. EXECUTEstu_info　‘马东’

修改为

CREATE PROCEDURE stu_info @name varchar(40)=' 刘卫平 '

AS

SELECT a.学号 , 姓名 , 课程编号 , 分数

FROM student_info a INNER JOIN grade ta

ON a.学号 =ta. 学号

WHERE姓名 =@name

3. （1）CREATE PROC stu_grade

AS

SELECT s. 姓名 ,c. 课程名称 ,g. 分数

FROM student_info s,curriculum c,grade g

WHERE s学号 ='0001' and c.　课程编号 =g. 课程编号 and g. 学号 ='0001'

（2）EXEC stu_grade

（3）sp_rename stu_grade,stu_g

4. （1）CREATE PROC stu_p_g @name varchar(40)

AS

SELECT s. 学号 ,c. 课程名称 ,g. 分数

FROM student_info s,curriculum c,grade g

WHERE s姓名 =@name and c. 课程编号 =g. 课程编号 and g. 学号 =s. 学号

（2）EXEC stu_p_g ' 刘卫平 '

（3）sp_helptext stu_p_g

5. （1）CREATE PROC stu_en

WITH ENCRYPTION AS

SELECT * FROM student_info

WHERE性别 =' 男 '

（2）EXEC stu_en

（3）DROP PROCEDURE stu_en

6. （1）CREATE PROC stu_g_r @xh int

AS

SELECT c. 课程名称 ,g. 分数

FROM student_info s,curriculum c,grade g

WHERE s学号 =@xh and s. 学号 =g. 学号 and c. 课程编号 =g. 课程编号

（2）EXEC stu_g_r 0002

（3）DROP PROC stu_g_r

7. 执行代码，　studentsdb 数据库含有　stu2 表，展开　stu2，其触发器项中含有　stu_str 触发器。

运行代码，因为此过程引动了触发器，使得插入信息的学号改变。

8. CREATE TRIGGER insert_g_tr
ON grade FOR INSERT
AS
IF EXISTS(SELEC课程编号 FROM curriculum WHERE 课程编号 =(SELECT课程编号 FROM Inserted))
BEGIN
PRINT '记录插入成功 '
END
ELSE
BEGIN
PRINT '不能插入记录 '
END

插入第一条记录： INSERT INTO grade(学号 ,课程编号 ,分数 ) VALUES(0004,00003,76)

结果显示：插入记录成功

插入第二条记录： INSERT INTO grade(学号 ,课程编号 ,分数 ) VALUES(0005,00005,69)

结果显示：不能插入记录