

T—SQL 目录

一、T—SQL 的组成.....	1
1、DML(数据操作语言 DATA MANIPULATION LANGUAGE)	1
2、DCL(数据控制语言 DATA CONTROL LANGUAGE)	1
3、DDL(数据定义语言 DATA DEFINITION LANGUAGE)	1
4、变量说明、流程控制、功能函数	1
二、库	1
1、建立库.....	1
2、删除库.....	2
三、表	2
1、建表.....	2
2、删表.....	3
四、约束	3
1、主键(PRIMARY KEY)	3
2、唯一性(UNIQUE)	4
3、默认填写(DEFAULT('.....') FOR).....	4
4、检查(CHECK(.....))	4
5、外键(FOREIGN KEY(列名) REFERENCES 主表名(列名))	4
6、删除约束.....	4
五、通配符	4
六、插入数据	5
1、注意事项.....	5
2、插入多行数据	5
七、更新数据行	6
八、删除数据行	7
1、删除指定的行	7
2、删除所有记录	7
九、查询	7
1、查询全部的行和列	7
2、查询部分行	7
3、自定义命名查询结果中的列名	8
十、模糊查询	10
1、LIKE.....	10
2、IS NULL.....	10
3、BETWEEN.....	11
4、IN	11
十一、聚合函数	11

1、SUM(求和)	11
2、AVG(求平均值)	11
3、MAX、MIN(求最大、最小值)	12
4、COUNT(计数)	12
十二、分组查询	12
1、单列分组查询	12
2、多列分组	12
3、HAVING(追加条件)	13
4、条件比较顺序	13
十三、多表联接查询	13
1、分类	13
2、多表内联结查询	14
3、多表外联接查询	15
4、多表交叉联接查询	15
十四、数据库用户	16
1、创建登录帐户	16
2、创建数据库用户	16
3、给用户分配权限	16
4、系统内置的数据库用户	16
十五、T—SQL 编程	17
1、变量	17
2、输出语句	18
3、逻辑控制语句	18
十六、高级查询	20
1、简单的子查询	20
2、IN (NOT IN) 子查询	20
3、EXISTS 子查询	20
十七、事务	21
1、使用 T-SQL 语句来管理事务	21
2、判断某条语句执行是否出错	21
3、事务必须具备 ACID 四个属性	21
4、事务的分类	21
5、事务例句	22
十八、索引	22
1、索引类型	22
2、使用 T-SQL 语句创建索引	23
3、索引的优缺点	23
4、创建索引的指导原则	23
5、索引例句	24

十九、视图	24
1、什么是视图	24
2、视图的用途	24
3、使用 T-SQL 语句创建视图	25
4、视图例句	25
二十、存储过程	25
1、什么是存储过程（PROCEDURE）	25
2、存储过程的优点	26
3、存储过程的分类	26
4、常用的系统存储过程	26
5、使用 T-SQL 语句创建和调用存储过程	27
6、处理存储过程中的错误	30
二十一、触发器	32
1、创建触发器的语法	32
2、例句	32

T—SQL

一、T—SQL 的组成

1、DML(数据操作语言 Data Manipulation Language)

查询、插入、删除和修改数据库中的数据。SELECT、INSERT、UPDATE、DELETE 等；

2、DCL(数据控制语言 Data Control Language)

用来控制存取许可、存取权限等。GRANT、REVOKE 等。

3、DDL(数据定义语言 Data Definition Language)

用来建立数据库、数据库对象和定义其列。CREATE TABLE 、DROP TABLE 等。

4、变量说明、流程控制、功能函数

定义变量、判断、分支、循环结构等。日期函数、数学函数、字符函数、系统函数等。

二、库

1、建立库

```
--判断是否存在该库,如果有则删除

USE master --设置当前数据库为 master,以便访问 sysdatabases 表

GO

IF EXISTS(SELECT * FROM sysdatabases WHERE name ='stuDB')

    DROP DATABASE stuDB

--建立数据库

CREATE DATABASE stuDB

    ON PRIMARY --默认就属于 PRIMARY 主文件组,可省略
```

```
(  
    NAME='stuDB_data', --主数据文件的逻辑名  
    FILENAME='D:\project\stuDB_data.mdf', --主数据文件的物理名  
    SIZE=5mb, --主数据文件初始大小  
    MAXSIZE=100mb, --主数据文件增长的最大值  
    FILEGROWTH=15% --主数据文件的增长率  
)  
LOG ON  
  
(  
    NAME='stuDB_log',  
    FILENAME='D:\project\stuDB_log.ldf',  
    SIZE=2mb,  
    FILEGROWTH=1MB  
)  
GO
```

2、删除库

```
USE master --设置当前数据库为 master,以便访问 sysdatabases 表  
GO  
IF EXISTS(SELECT * FROM sysdatabases WHERE name ='stuDB')  
    DROP DATABASE stuDB
```

EXISTS() 语句：检测是否存在 stuDB 数据库, 如果存在 stuDB 数据库, 则删除
sysdatabases 表在 master 数据库中, 保存着当前系统中所有的数据库

三、表

1、建表

--判断是否存在该表,有则删除

```
USE stuDB --将当前数据库设置为 stuDB ,以便在 stuDB 数据库中建表

GO

IF EXISTS(SELECT * FROM sysobjects WHERE name='stuInfo')

DROP TABLE stuInfo

--建表

CREATE TABLE stuInfo /*-创建学员信息表-*/

(

stuName VARCHAR(20) NOT NULL, --姓名,非空(必填)

stuNo CHAR(6) NOT NULL, --学号,非空(必填)

stuAge INT NOT NULL, --年龄,INT 类型默认为 4 个字节

stuID NUMERIC(18,0), --身份证号

stuSeat SMALLINT IDENTITY (1,1), --座位号,自动编号

stuAddress TEXT --住址,允许为空,即可选输入

)

GO
```

2、删表

```
USE 库名 --将当前数据库设置为 stuDB ,以便在 stuDB 数据库中建表

GO

IF EXISTS(SELECT * FROM sysobjects WHERE name='表名')

DROP TABLE 表名
```

四、约束

1、主键(primary key)

```
ALTER TABLE stuInfo

ADD CONSTRAINT PK_stuNo PRIMARY KEY (stuNo)
```

2、唯一性(unique)

```
ALTER TABLE stuInfo  
  
    ADD CONSTRAINT UQ_stuID UNIQUE (stuID)
```

3、默认填写(default('.....') for)

```
ALTER TABLE stuInfo  
  
    ADD CONSTRAINT DF_stuAddress  
  
        DEFAULT ('地址不详') FOR stuAddress
```

4、检查(check(.....))

```
ALTER TABLE stuInfo  
  
    ADD CONSTRAINT CK_stuAge  
  
        CHECK(stuAge BETWEEN 15 AND 40)
```

5、外键(foreign key(列名) references 主表名(列名))

```
ALTER TABLE stuMarks  
  
    ADD CONSTRAINT FK_stuNo  
  
        FOREIGN KEY(stuNo) REFERENCES stuInfo(stuNo)
```

6、删除约束

```
ALTER TABLE 有约束的表名 DROP 约束名
```

五、通配符

通配符	解释	示例
'_'	一个字符	A Like 'C_'
'%'	任意长度的字符串	B Like 'CO_%'
'[]'	括号中所指定范围内的一个字符	C Like '9W0[1-2]'
'[^]'	不在括号中所指定范围内的一个字符	D Like '%[A-D][^1-2]'

六、插入数据

INSERT 表名(列名) VALUES (插入的列值)

```
insert stuinfo(stuname,stuno,stuage,stuid,stuaddress)
values('张三',001,20,100,'hello')
```

1、注意事项

A、每次插入一行数据,不可能只插入半行或者几列数据,因此,插入的数据是否有效将按照整行的完整性的要求来检验。

B、每个数据值的数据类型、精度和小数位数必须与相应的列匹配。

C、不能为标识列指定值,因为它的数字是自动增长的。

D、如果在设计表的时候就指定了某列不允许为空,则必须插入数据。

E、插入的数据项,要求符合检查约束的要求。

F、具有缺省值的列,可以使用 DEFAULT(缺省)关键字来代替插入的数值。

2、插入多行数据

(1)对象表存在

INSERT INTO <表名>(列名)

SELECT <列名> FROM <源表名>

INSERT INTO stuinfobak (stuname,stuno,stuage)

SELECT stuname,stuno,stuage FROM stuinfo

(Stuinfobak 表必须在数据库中存在)

(2)对象表不存在

```
SELECT (列名) INTO <表名> FROM <源表名>
```

```
SELECT stuname,stuno,stuage INTO stuinfobak1 FROM stuinfo
```

(Stuinfobak1 表必须在数据库中不存在)

(3)插入新的标识列

IDENTITY(数据类型,标识种子,标识增量)

```
SELECT IDENTITY(数据类型,标识种子,标识增长量) AS 列名
```

```
INTO 新表 FROM 原始表
```

```
SELECT Students.SName,Students.SAddress,Students.SEmail,
```

```
IDENTITY(int,1,1) As StudentID
```

```
INTO TongXunLuEX FROM Students
```

(4)插入多行内容

```
INSERT INTO <表名>(列名)
```

```
SELECT 列内容 UNION
```

```
SELECT 列内容 UNION
```

```
.....
```

七、更新数据行

```
UPDATE <表名> SET <列名 = 更新值>
```

```
[WHERE <更新条件>]
```

```
UPDATE Students
```

```
SET SAddress ='北京女子职业技术学校家政班'
```

```
WHERE SAddress = '北京女子职业技术学校刺绣班'
```

```
UPDATE Scores
```

```
SET Scores = Scores + 5  
  
WHERE Scores <= 95
```

八、删除数据行

1、删除指定的行

```
DELETE FROM <表名> [WHERE <删除条件>]
```

注意：DELETE FROM 不会只删单个字段,要删就是整行

2、删除所有记录

```
TRUNCATE TABLE <表名>
```

注意：不能用于有外键约束引用的表。删除后,表的结构、列、约束、索引不变。

九、查询

```
SELECT <列名>  
  
FROM <表名>  
  
[WHERE <查询条件表达式>]  
  
[ORDER BY <排序的列名>[ASC(升)或 DESC(降)]] --默认为升序
```

1、查询全部的行和列

```
SELECT * FROM Students
```

2、查询部分行

```
SELECT <列名>  
  
FROM <表名>  
  
WHERE <查询条件表达式>
```

3、自定义命名查询结果中的列名

(1)使用 AS 来命名列

```
SELECT 原表中列名 AS 查询结果中显示的列名  
  
FROM <表名>  
  
WHERE <查询条件表达式>  
  
SELECT Stuno AS 学员编号,StuName AS 学员姓名,StuAddress AS 学员地址  
  
FROM stuinfo  
  
WHERE SAddress like '地址不详'
```

(2)合并两列数据,以规定格式输出查询结果

```
SELECT 原表中列名+' '+原表中列名 AS 查询结果中显示的列名  
  
FROM 表名
```

(3)使用=来命名列

```
SELECT 查询结果中显示的列名 = 原表中列名  
  
FROM <表名>  
  
WHERE <查询条件表达式>  
  
SELECT 查询结果中显示的列名 = 原表中列名+' '+原表中列名  
  
FROM 表名
```

(4)查询某几列为空的行

```
SELECT 列名 A  
  
FROM 表名  
  
WHERE 条件列名 B IS NULL
```

显示结果：B 列为空的 A 列内容。

(5)显示结果中加入常量列

```
SELECT 列 A = SName,列 B = SAddress,'常量列内容'AS 常量列名称  
  
FROM Students  
  
SELECT 姓名=SName,地址= SAddress,'河北新龙' AS 学校名称  
  
FROM Students
```

(6)限制查询结果输出的行数

```
SELECT TOP 5 列名  
  
FROM 表名  
  
WHERE 条件
```

显示结果：符合条件的前五行。

(7)返回百分之多少行

```
SELECT TOP 20 PERCENT 列名  
  
FROM 表名  
  
WHERE 条件
```

显示结果：符合条件的行数的前 20%行。

(8)升序排列(默认)ASC

```
SELECT StudentID As 学员编号,(Score*0.9+5) As 综合成绩  
  
FROM Score  
  
WHERE (Score*0.9+5)>60  
  
ORDER BY Score
```

(9)降序排列 DESC

```
SELECT Au_Lname +'.'+Au_fName AS EMP  
  
From 表 A Union  
  
SELECT fName +'.'+ LName AS EMP
```

```
From 表 B

ORDER BY EMP DESC
```

显示结果：混合查找两张表中的列,并按格式输出到新列中,并按新列排序输出。

(10)按多列排序

```
SELECT StudentID As 学员编号, Score As 成绩

FROM 表名

WHERE Score>60

ORDER BY Score,studentID
```

显示结果：先按 Score 排序，再按 studentID 排序。

十、模糊查询

1、LIKE

查询时,字段中的内容并不一定与查询内容完全匹配,只要字段中含有这些内容。

```
SELECT StuName AS 姓名

FROM Stinfo

WHERE stuname LIKE '徐%'
```

显示结果：姓为“徐”的人的名字。

2、IS NULL

把某一字段中内容为空的记录查询出来。

```
SELECT StuName AS 姓名,StuAddress AS 地址。

FROM Stinfo

WHERE StuAddress IS NULL
```

显示结果：把地址栏为空的显示出来。

3、BETWEEN

把某一字段中内容在特定范围内的记录查询出来。

```
SELECT  StuNo, Score  
  
FROM    Stumarks  
  
WHERE Score BETWEEN 60 AND 80
```

显示结果：把分数 $80 \geq \text{Score} \geq 60$ 的显示出来。

4、IN

把某一字段中内容与所列出的查询内容列表匹配的记录查询出来。

```
SELECT  StuName AS 学员姓名, StuAddress As 地址  
  
FROM    Stuinfo  
  
WHERE   StuAddress IN ('北京','广州','上海')
```

显示结果：把地址在('北京','广州','上海')里的显示出来。

十一、聚合函数

1、SUM(求和)

```
SELECT  SUM(Score)  
  
FROM    Stumarks  
  
WHERE   条件
```

显示结果：把符合条件的 Score 求和,然后显示结果。

2、AVG(求平均值)

```
SELECT  AVG(Score) AS 平均成绩  
  
From    Score  
  
WHERE   Score >=60
```

显示结果：把 $\text{Score} \geq 60$ 的成绩求平均值,然后显示结果,显示的列名为“平均成绩”

3、MAX、MIN(求最大、最小值)

```
SELECT  MAX (Score) AS 最高分, MIN (Score) AS 最低分  
  
From  Score  
  
WHERE  Score >=60
```

显示结果：把 Score>=60 中的最高分和最低分显示出来。

4、COUNT(计数)

```
SELECT  COUNT (*) AS 及格人数  
  
From  Score  
  
WHERE  Score>=60
```

显示结果：把 Score 列中,>=60 的个数统计出来,然后显示统计数目。

十二、分组查询

1、单列分组查询

```
SELECT  CourseID, AVG(Score) AS 课程平均成绩  
  
FROM  Score  
  
GROUP BY  CourseID
```

显示结果：按 CourseID 组求 Score 的平均值,然后将 CourseID 和平均值显示出来。

2、多列分组

```
SELECT  StudentID AS 学员编号, CourseID AS 内部测试, AVG(Score) AS 平均成绩  
  
FROM  Score  
  
GROUP BY  StudentID, CourseID
```

显示结果：显示所有学员的：“学员编号”(StudentID),“内部测试”(CourseID),“平均成绩”(AVG(Score))。

如果同一 CourseID 组中出现了相同的 StudentID,则显示出来的是这一 CourseID 组中相同的 StudentID 的平均成绩。

3、HAVING(追加条件)

```
SELECT StudentID AS 学员编号, CourseID AS 内部测试, AVG(Score) AS 平均成绩
FROM Score
GROUP BY StudentID, CourseID
HAVING COUNT(Score) > 1
```

显示结果：显示补考学员的：“学员编号”(StudentID)，“内部测试”(CourseID)，“平均成绩”(AVG(Score))。
如果同一 CourseID 组中的同一 StudentID 组中记录 Score 的次数，如果次数>1，则显示出来“学员编号”(StudentID)，“内部测试”(CourseID)，“平均成绩”(AVG(Score))。

4、条件比较顺序

WHERE 子句从数据源中去掉不符合其搜索条件的数据。

GROUP BY 子句搜集数据行到各个组中，统计函数为各个组计算统计值。

HAVING 子句去掉不符合其组搜索条件的各组数据行。

WHERE——>GROUP BY——>HAVING

十三、多表联接查询

1、分类

(1)内联接(INNER JOIN)

(2)外联接

- A、左外联结 (LEFT JOIN)
- B、右外联结 (RIGHT JOIN)
- C、完整外联结 (FULL JOIN)

(3)交叉联接(CROSS JOIN)

2、多表内联结查询

(1)建立联接

```
SELECT S.SName,C.CourseID,C.Score  
  
From Score AS C  
  
INNER JOIN Students AS S  
  
ON C.StudentID = S.SCode
```

显示结果：把 Score 表和 Students 表建立内联结,查询 C.StudentID = S.SCode 时,显示 S.SName,C.CourseID,C.Score 的内容。

(2)未建立联接

```
SELECT Students.SName, Score.CourseID, Score.Score  
  
FROM Students,Score  
  
WHERE Students.SCode = Score.StudentID
```

显示结果：同上。

(3)多表联接查询—三表联接

```
SELECT S.SName AS 姓名, CS.CourseName AS 课程, C.Score AS 成绩  
  
FROM Students AS S  
  
INNER JOIN Score AS C  
  
ON (S.SCode = C.StudentID)  
  
INNER JOIN Course AS CS  
  
ON (CS.CourseID = C.CourseID)
```

(4)区别

建立联接的查询速度比没有建立的快得多。

3、多表外联接查询

(1)左外联接

(LEFT JOIN 或 LEFT OUTER JOIN)

```
SELECT S.SName,C.CourseID,C.Score
```

```
From Students AS S
```

```
LEFT JOIN Score AS C
```

```
ON C.StudentID = S.SCode
```

显示结果：Students 为左表(left join)Score 为右表。左表中有的,右表中没有的显示空值(NULL)。

(2)右外联接

(RIGHT JOIN 或 RIGHT OUTER JOIN)

```
SELECT Titles.Title_id, Titles.Title, Publishers.Pub_name
```

```
FROM titles
```

```
RIGHT OUTER JOIN Publishers
```

```
ON Titles.Pub_id = Publishers.Pub_id
```

显示结果：与左外连接相反。

(3)完整外联接

(FULL JOIN 或 FULL OUTER JOIN)

显示结果：左表和右表中的所有行。当某行在另一个表中没有匹配行时,则另一个表的选择列为空值；如果有匹配行,则显示结果包括左右表中的所有列值。

4、多表交叉联接查询

(CROSS JOIN)

交叉联接返回左表中的所有行,左表中的每一行与右表中的所有行一一组合,相当于两个表“相乘”。

十四、数据库用户

1、创建登录帐户

(1)添加 Windows 登录帐户

```
EXEC sp_grantlogin 'jbtraining\S26301'(域名\用户名)
```

(2)添加 SQL 登录帐户

```
EXEC sp_addlogin 'zhangsan','1234'
```

EXEC 表示调用存储过程,存储过程类似 C 语言的函数。

内置的系统管理员 帐户 sa,密码默认为空,建议修改密码

2、创建数据库用户

```
USE 库名
```

```
GO
```

```
EXEC sp_grantdbaccess '登录帐户名','数据库用户名'
```

其中,“数据库用户”为可选参数,默认为登录帐户,即数据库用户默认和登录帐户同名。

3、给用户分配权限

```
GRANT 权限 [ON 表名 ] TO 数据库用户
```

权限: select、insert、update、delete、create table

4、系统内置的数据库用户

(1)dbo 用户

A、表示数据库的所有者(DB Owner)

B、无法删除 dbo 用户,此用户始终出现在每个数据库中

(2)guest 用户

- A、适用于没有数据库用户的登录帐号访问
- B、每个数据库可有也可删除

十五、T—SQL 编程

1、变量

(1)局部变量

- A、局部变量必须以标记@作为前缀 ,如@age。
- B、局部变量的使用也是先声明,再赋值。
- C、声明局部变量

DECLARE @变量名 数据类型

- D、赋值

SET @变量名 = 值

SELECT @变量名 = 值

(2)全局变量

- A、全局变量必须以标记@@作为前缀,如@@version
- B、全局变量由系统定义和维护,我们只能读取,不能修改全局变量的值
- C、全局变量的类型与含义

变 量	含 义
@@ERROR	最后一个 T-SQL 错误的错误号
@@IDENTITY	最后一次插入的标识值
@@LANGUAGE	当前使用的语言的名称
@@MAX_CONNECTIONS	可以创建的同时连接的最大数目
@@ROWCOUNT	受上一个 SQL 语句影响的行数
@@SERVERNAME	本地服务器的名称
@@TRANSCOUNT	当前连接打开的事务数
@@VERSION	SQL Server 的版本信息

2、输出语句

(1)print 局部变量或字符串

结果在消息窗口以文本方式显示。

(2)SELECT 局部变量 AS 自定义列名

结果在网格窗口以表格方式显示。

3、逻辑控制语句

(1)IF-ELSE 条件语句

```
IF(条件)
    BEGIN
        语句块.....
    END
ELSE
    .....
```

(2)WHILE 循环语句

WHILE(条件)

BEGIN

语句块……

(BREAK)-- 跳出循环,可选语句。

END

(3)CASE—END

十六、高级查询

1、简单的子查询

```
SELECT * FROM stuInfo  
WHERE stuAge > (SELECT stuAge FROM stuInfo where stuName='姓名')
```

(1) 子查询的一般用法

- A、SELECT ... FROM 表1 WHERE 字段1 > (子查询)。
- B、外面的查询称为父查询，括号中嵌入的查询称为子查询。
- C、UPDATE、INSERT、DELETE 一起使用，语法类似于 SELECT 语句。
- D、将子查询和比较运算符联合使用，必须保证子查询返回的值不能多于一个。

(2) 子查询替换表连接

- A、一般来说，表连接都可以用子查询替换，但有的子查询却不能用表连接替换。
- B、子查询比较灵活、方便，常作为增删改查的筛选条件，适合于操纵一个表的数据。
- C、表连接更适合于查看多表的数据。

2、IN (NOT IN) 子查询

```
SELECT stuName FROM stuInfo  
WHERE stuNo IN (SELECT stuNo FROM stuMarks WHERE writtenExam=60)
```

IN (在子查询范围内)，NOT IN (不在子查询范围内)

IN (NOT IN) 后面的子查询可以返回多条记录，常用 IN 替换等于 (=) 的比较子查询。

3、EXISTS 子查询

```
IF EXISTS (子查询)  
语句.....
```

- A、如果子查询的结果非空，即记录条数 1 条以上，则 EXISTS (子查询) 将返回真 (true)，否则返

回假(false)。

B、EXISTS 也可以作为 WHERE 语句的子查询，但一般都能用 IN 子查询替换。

C、EXISTS 和 IN 一样，允许添加 NOT 取反，表示不存在。

十七、事务

1、使用 T-SQL 语句来管理事务

开始事务：BEGIN TRANSACTION

提交事务：COMMIT TRANSACTION

回滚（撤销）事务：ROLLBACK TRANSACTION

2、判断某条语句执行是否出错

使用全局变量 @@ERROR。@@ERROR 只能判断当前一条 T-SQL 语句执行是否有错，为了判断事务中所有 T-SQL 语句是否有错，我们需要对错误进行累计：

如： SET @errorSum=@errorSum+@@error

3、事务必须具备 ACID 四个属性

原子性 (Atomicity)：事务是一个完整的操作。事务的各步操作是不可分的（原子的）；要么都执行，要么都不执行

一致性 (Consistency)：当事务完成时，数据必须处于一致状态

隔离性 (Isolation)：对数据进行修改的所有并发事务是彼此隔离的，这表明事务必须是独立的，它不应以任何方式依赖于或影响其他事务

永久性 (Durability)：事务完成后，它对数据库的修改被永久保持，事务日志能够保持事务的永久性

4、事务的分类

显示事务：用 BEGIN TRANSACTION 明确指定事务的开始，这是最常用的事务类型

隐性事务：通过设置 SET IMPLICIT_TRANSACTIONS ON 语句，将隐性事务模式设置为打开，下一个语句自动启动一个新事务。当该事务完成时，再下一个 T-SQL 语句又将启动一个新事务

自动提交事务：这是 SQL Server 的默认模式，它将每条单独的 T-SQL 语句视为一个事务，如果成功执行，则自动提交；如果错误，则自动回滚

5、事务例句

```
BEGIN TRANSACTION  -- 开始事务

    DECLARE @errorSum INT      --定义变量接收错误语句数

    SET @errorSum = 0  --初始化变量，无错误

    SQL 语句 1

    SET @errorSum=@errorSum+@@error

    SQL 语句 2

    SET @errorSum=@errorSum+@@error

    .....

    IF @errorSum <> 0      --产生错误，回滚事务

        BEGIN

            ROLLBACK TRANSACTION

        END

    ELSE  --如果无错误，提交事务

        BEGIN

            COMMIT TRANSACTION

        END

GO
```

十八、索引

1、索引类型

唯一索引：唯一索引不允许两行具有相同的索引值

主键索引：为表定义一个主键将自动创建主键索引，主键索引是唯一索引的特殊类型。主键索引要求主键中的每个值是唯一的，并且不能为空

聚集索引(Clustered)：表中各行的物理顺序与键值的逻辑（索引）顺序相同，每个表只能有一个

非聚集索引(Non-clustered)：非聚集索引指定表的逻辑顺序。数据存储在一个位置，索引存储在另一个位置，索引中包含指向数据存储位置的指针。可以有多个，小于 249 个

2、使用 T-SQL 语句创建索引

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED] INDEX index_name  
ON table_name (column_name...)  
[WITH FILLFACTOR=x]
```

UNIQUE 表示唯一索引，可选。

CLUSTERED、NONCLUSTERED 表示聚集索引还是非聚集索引，可选。

FILLFACTOR 表示填充因子，指定一个 0 到 100 之间的值，该值指示索引页填满的空间所占的百分比。

3、索引的优缺点

(1) 优点

- A、加快访问速度
- B、加强行的唯一性

(2) 缺点

- A、带索引的表在数据库中需要更多的存储空间
- B、操纵数据的命令需要更长的处理时间，因为它们需要对索引进行更新

4、创建索引的指导原则

(1) 请按照下列标准选择建立索引的列

- A、该列用于频繁搜索
- B、该列用于对数据进行排序

(2) 请不要使用下面的列创建索引

- A、列中仅包含几个不同的值。

B、表中仅包含几行。为小型表创建索引可能不太划算，因为 SQL Server 在索引中搜索数据所花的时间比在表中逐行搜索所花的时间更长

5、索引例句

```
USE stuDB

GO

IF EXISTS (SELECT name FROM sysindexes WHERE name = 'IX_writtenExam')

    DROP INDEX stuMarks.IX_writtenExam

/*--笔试列创建非聚集索引：填充因子为 30%--*/

CREATE NONCLUSTERED INDEX IX_writtenExam

    ON stuMarks(writtenExam)

    WITH FILLFACTOR= 30

GO

/*-----指定按索引 IX_writtenExam 查询-----*/

SELECT * FROM stuMarks (INDEX=IX_writtenExam)

    WHERE writtenExam BETWEEN 60 AND 90
```

十九、视图

1、什么是视图

A、视图是一张虚拟表，它表示一张表的部分数据或多张表的综合数据，其结构和数据是建立在对表的查询基础上

B、视图中并不存放数据，而是存放在视图所引用的原始表（基表）中

C、同一张原始表，根据不同用户的不同需求，可以创建不同的视图

2、视图的用途

A、筛选表中的行

B、防止未经许可的用户访问敏感数据

- C、降低数据库的复杂程度
- D、将多个物理数据库抽象为一个逻辑数据库

3、使用 T-SQL 语句创建视图

```
CREATE VIEW view_name  
  
AS  
  
<select 语句……>  
  
GO
```

4、视图例句

```
IF EXISTS (SELECT * FROM sysobjects WHERE name = 'view_stuInfo_stuMarks')  
  
DROP VIEW view_stuInfo_stuMarks  
  
GO  
  
CREATE VIEW view_stuInfo_stuMarks  
  
AS  
  
SELECT 姓名=stuName,学号=stuInfo.stuNo,  
  
        笔试成绩 =writtenExam, 机试成绩=labExam,平均分=(writtenExam+labExam)/2  
  
FROM stuInfo LEFT JOIN stuMarks  
  
ON stuInfo.stuNo=stuMarks.stuNo  
  
GO  
  
SELECT * FROM view_stuInfo_stuMarks
```

二十、存储过程

1、什么是存储过程 (procedure)

一组预编译的 SQL 语句，它可以包含数据操纵语句、变量、逻辑控制语句等。

2、存储过程的优点

- A、执行速度更快
- B、允许模块化程序设计
- C、提高系统安全性
- D、减少网络流量

3、存储过程的分类

(1)系统存储过程

由系统定义，存放在 master 数据库中，类似 C 语言中的系统函数，系统存储过程的名称都以“sp_”开头或“xp_”开头。

(2)用户自定义存储过程

由用户在自己的数据库中创建的存储过程，类似 C 语言中的用户自定义函数。

4、常用的系统存储过程

扩展存储过程：xp_cmdshell

可以执行 DOS 命令下的一些的操作，以文本行方式返回任何输出

调用语法：

EXEC xp_cmdshell DOS 命令 [NO_OUTPUT]

例句：

EXEC xp_cmdshell 'mkdir d:\bank', NO_OUTPUT

EXEC xp_cmdshell 'dir D:\bank\' --查看文件

系统存储过程	说明
sp_databases	列出服务器上的所有数据库。
sp_helpdb	报告有关指定数据库或所有数据库的信息
sp_renamedb	更改数据库的名称
sp_tables	返回当前环境下可查询的对象的列表

sp_columns	回某个表列的信息
sp_help	查看某个表的所有信息
sp_helpconstraint	查看某个表的约束
sp_helpindex	查看某个表的索引
sp_stored_procedures	列出当前环境中的所有存储过程。
sp_password	添加或修改登录帐户的密码。
sp_helptext	显示默认值、未加密的存储过程、用户定义的存储过程、触发器或视图的实际文本。

5、使用 T-SQL 语句创建和调用存储过程

创建：

```
CREATE PROC[EDURE] 存储过程名
```

```
@参数 1 数据类型 = 默认值,
```

```
.....,
```

```
@参数 n 数据类型 = 默认值
```

```
AS
```

```
SQL 语句
```

```
GO
```

调用：

```
EXEC 过程名 [参数]
```

注意：

A、和 C 语言的函数一样，参数可选

B、参数分为输入参数、输出参数

C、输入参数允许有默认值

D、存储过程的最大大小为 128 MB

(1) 不带参数的存储过程

创建：

```
CREATE PROCEDURE proc_stu

AS

DECLARE @writtenAvg float,@labAvg float

SELECT @writtenAvg=AVG(writtenExam),

        @labAvg=AVG(labExam) FROM stuMarks

print '笔试平均分: '+convert(varchar(5),@writtenAvg)

print '机试平均分: '+convert(varchar(5),@labAvg)

IF (@writtenAvg>70 AND @labAvg>70)

    print '本班考试成绩: 优秀'

ELSE

    print '本班考试成绩: 较差'

print '-----'

print ' 参加本次考试没有通过的学员: '

SELECT stuName,stuInfo.stuNo,writtenExam,labExam

        FROM  stuInfo  INNER JOIN stuMarks ON stuInfo.stuNo=stuMarks.stuNo

        WHERE writtenExam<60 OR labExam<60

GO

调用:

EXEC proc_stu
```

(2) 带输入参数的存储过程

创建:

```
CREATE PROCEDURE proc_stu

    @writtenPass int=60, --输入参数可带默认值

    @labPass int=60

AS

print '-----'

print '          参加本次考试没有通过的学员: '

SELECT stuName,stuInfo.stuNo,writtenExam,
```

```
labExam FROM stuInfo

INNER JOIN stuMarks ON stuInfo.stuNo=stuMarks.stuNo

WHERE writtenExam<@writtenPass OR labExam<@labPass

GO
```

调用：

```
EXEC proc_stu --都采用默认值

EXEC proc_stu 64 --第二个参数采用默认值

EXEC proc_stu @labPass=55 --第一个参数采用默认值

EXEC proc_stu 60,55 --都不采用默认值
```

(3) 带输出参数的

创建：

```
CREATE PROCEDURE proc_stu

    @notpassSum int OUTPUT, --输出（返回）参数

    @writtenPass int=60, --推荐将默认参数放后面

    @labPass int=60

AS

.....

SELECT stuName,stuInfo.stuNo,writtenExam,

labExam FROM stuInfo INNER JOIN stuMarks

ON stuInfo.stuNo=stuMarks.stuNo

WHERE writtenExam<@writtenPass OR labExam<@labPass

SELECT @notpassSum=COUNT(stuNo)

FROM stuMarks WHERE writtenExam<@writtenPass OR labExam<@labPass

GO

调用：

/*---调用存储过程----*/

DECLARE @sum int

EXEC proc_stu @sum OUTPUT ,64
```



```
print '-----'

IF @sum>=3

    print '未通过人数: '+convert(varchar(5),@sum)+ '人,
    超过 60%,及格分数线还应下调'

ELSE

    print '未通过人数: '+convert(varchar(5),@sum)+ '人,
    已控制在 60%以下,及格分数线适中'

GO
```

6、处理存储过程中的错误

(1) PRINT 语句显示错误信息

可以使用 PRINT 语句显示错误信息，但这些信息是临时的，只能显示给用户

(2) RAISERROR 显示用户定义的错误信息

可指定严重级别，设置系统变量@@ERROR，记录所发生的错误等。

(3) RAISERROR 语句的用法

```
RAISERROR (msg_id | msg_str,severity,state WITH option[,...n])
```

msg_id: 在 sysmessages 系统表中指定用户定义错误信息。

msg_str: 用户定义的特定信息，最长 255 个字符。

severity: 定义严重性级别。用户可使用的级别为 0-18 级。

state: 表示错误的状态，1 至 127 之间的值。

option: 指示是否将错误记录到服务器错误日志中。

(4) 例句

```
CREATE PROCEDURE proc_stu

@notpassSum int OUTPUT, --输出参数

@writtenPass int=60, --默认参数放后
```

```
@labPass int=60      --默认参数放后

AS

IF (NOT @writtenPass BETWEEN 0 AND 100)

    OR (NOT @labPass BETWEEN 0 AND 100)

BEGIN

    RAISERROR ('及格线错误，请指定 0—100 之间的分 数，统计中断退出',16,1)

    RETURN  ---立即返回，退出存储过程

END

.....其他语句同上例，略

GO


/*---调用存储过程，测试 RAISERROR 语句----*/

DECLARE @sum int,  @t int

EXEC proc_stu @sum OUTPUT ,604

SET @t=@@ERROR

print '错误号：'+convert(varchar(5),@t )

IF @t<>0

    RETURN  --退出批处理，后续语句不再执行

    print '-----'

IF @sum>=3

    print '未通过人数：'+convert(varchar(5),@sum)+'人,超过 60%,及格分数线还应下调'

ELSE

    print '未通过人数：'+convert(varchar(5),@sum)+'人,已控制在 60%以下，

        及格分数线适中'

GO
```

二十一、触发器

1、创建触发器的语法

```
CREATE TRIGGER trigger_name ON table_name  
  
[WITH ENCRYPTION]  
  
FOR [DELETE, INSERT, UPDATE]  
  
AS  
  
    T-SQL 语句……  
  
GO
```

2、例句

(1) Insert 触发器

```
CREATE TRIGGER trig_transInfo ON transInfo  
  
FOR INSERT  
  
AS  
  
    DECLARE @type char(4),@outMoney MONEY  
  
    DECLARE @myCardID char(10),@balance MONEY  
  
    SELECT @type=transType,@outMoney=transMoney,  
  
    @myCardID=cardID FROM inserted  
  
    IF (@type='支取')  
  
        UPDATE bank SET currentMoney=currentMoney-@outMoney  
  
        WHERE cardID=@myCardID  
  
    ELSE  
  
        UPDATE bank SET currentMoney=currentMoney+@outMoney  
  
        WHERE cardID=@myCardID  
  
GO
```

(2) Delete 触发器

```
CREATE TRIGGER trig_delete_transInfo ON transInfo
FOR DELETE
AS
    print '开始备份数据，请稍后.....'

    IF NOT EXISTS(SELECT * FROM sysobjects WHERE name='backupTable')

        SELECT * INTO backupTable FROM deleted

    ELSE

        INSERT INTO backupTable SELECT * FROM deleted

    print '备份数据成功，备份表中的数据为:'

    SELECT * FROM backupTable

GO
```

(3) Update 触发器

```
CREATE TRIGGER trig_update_bank ON bank
FOR UPDATE
AS
    DECLARE @beforeMoney MONEY,@afterMoney MONEY

    SELECT @beforeMoney=currentMoney FROM deleted

    SELECT @afterMoney=currentMoney FROM inserted

    IF ABS(@afterMoney-@beforeMoney)>20000

    BEGIN

        print '交易金额:' + convert(varchar(8),

            ABS(@afterMoney-@beforeMoney))

        RAISERROR ('每笔交易不能超过 2 万元，交易失败',16,1)

        ROLLBACK TRANSACTION

    END

GO
```

```
CREATE TRIGGER trig_update_transInfo ON transInfo
FOR UPDATE
AS
IF UPDATE(transDate) --判断是否被更改
BEGIN
    print '交易失败.....'
    RAISERROR ('安全警告：交易日期不能修改，由系统自动产生',16,1)
    ROLLBACK TRANSACTION
END
GO
```