

SQLServer的两种验证方式：用户名(sa)验证和Windows验证，开发时用Windows验证就行。

除了Access、SQLServerCE、SQLite等文件型数据库之外，大部分数据库都需要数据库服务器才能运行。学习、开发时是连接本机的数据库，上线运行时是数据库运行在单独的服务器。用户名可以写 `localhost` 或 `127.0.0.1` 或 `.` 或 `ZZMF-20190903EB`

如果要使用sa登录，修改 `localhost(SQL Server ...)` - 属性 - 安全性 - 服务器身份验证 - 勾选 `SQL Server` 和 `Windows` 身份验证模式，将 `服务 - SQL Browser` 开启，然后将 `SQL Server (MSSQLSERVER)` 重新启动

MS SQLServer的每个数据库包含：

- 1个主数据文件(.mdf)必须。
- 1个事务日志文件 (.ldf) 必须。

可以包含：

- 任意多个次要数据文件(.ndf)
- 多个事务日志文件

主键就是数据行的唯一标识，一个表可以没有主键，但是会非常难以处理。

约束-保证数据完整性（数据检查）

- 主键约束(**PK**) primary key constraint 唯一且不为空
- 唯一约束 (**UQ**)unique constraint 唯一，允许为空，**但只能出现一次**
- 默认约束 (**DF**)default constraint 默认值
- 检查约束 (**CK**)check constraint 范围以及格式限制
- 外键约束 (**FK**)foreign key constraint 表关系：保证外键值来源于主键
- 增加外键约束时，设置级联更新、级联删除：
  - `[ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]`
  - `[ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]`

**只有整数才可以做标识( IDENTITY )**

bit控制 `true` 或 `false`，可以写 `1` 或 `0`，但是**手动输入表格**时不能写 `1` 或 `0`，只能写 `true` 或 `false`，**insert**时或 `insert into Department values(0)` 或 `insert into Department values('false')`，但是在 `if` 语句中，只能用 `0` 或 `1`

`char` 当存储的字符数量小于指定的空间的时候，空间不会收缩，但是大于的时候，会报错(截断二进制数据)

`varchar` 说明分配的空间是一个可以动态变化的空间。当存储的字符长度小于分配的空间的时候，多余的空间会自行回收，但是大于的时候还会报错

`n` 是unicode，不管哪种字符都会使用2个字节进行存储

修改日期值时，记得包含在单引号以内

N前缀： `N'字符串'`，在服务器上执行的代码中（例如在存储过程和触发器中）显示的 Unicode 字符串常量必须 大写字母 `N` 为前缀。即使所引用的列已定义为 Unicode 类型，也应如此。如果不使用 `N` 前缀，可能导致不识别某 字符。

- `select *`、`SeLeCT *`
  - SQL 对大小写不敏感！不敏感指的是SQL关键字，对于值也不区分
- 如果您使用的是 MS Access 和 SQL Server 2000，则不必在每条 SQL 语句之后使用分号，不过某些数据库软件要求必须使用分号。
- 建库、删除数据库、创建表、删除表不仅可以手工完成，还可以执行SQL语句完成，在自动化部署、数据导入中用的很多
- 没有 `==`，赋值和逻辑都是使用 `=`
- (\*) SQL主要分DDL（数据定义语言,建表、建库等语句。）、DML（数据操作语言 multipulation）和DCL（数据库控制语言）。`Create Table`、`Drop Table`、`Alter Table` 等属于DDL，`Select`、`Insert`、`Update`、`Delete` 等属于DML，`GRANT` 授权、`REVOKE` 取消授权属于DCL
- 在sql中没有 `""`，所有的字符值都使用 `' '` 包含
- 任何类型的值都可以使用 `' '` 包含。
  - `+` 首先是一个算术运算符，只有 `+` 两边都是字符串 `+` 才是一个连接符，如果有一边是数值类型，那么系统会：将另外一个值做隐式的类型转换，如果可以转换就进行转换，如果不可以转换就报错
  - `print '1' + 2` 的结果是3
  - `print 1 + 'a'` 会报错
- `select` 可以以文本模式显示

## 常用函数

- SQL的字符串是从1开始计数的
  - 关于GO
    - GO 不是 Transact-SQL 语句；
    - 它是 sqlcmd 和 osql 实用工具以及 SQL Server Management Studio 代码编辑器识别的命令。
    - SQL Server 应用程序可以将多个 Transact-SQL 语句作为一个批发送到 SQL Server 的实例来执行.然后,该批中的语句被编译成一个执行计划.程序员在 SQL Server 实用工具中执行特殊语句,或生成 Transact-SQL 语句的脚本在 SQL Server 实用工具中运行时,使用 GO 作为批结束的信号。
    - 如果基于 ODBC 或 OLE DB API 的应用程序试图执行 GO 命令,会收到语法错误.SQL Server 实用工具从不向服务器发送 GO 命令。
    - GO 命令和 Transact-SQL 语句不能在同一行中.但在 GO 命令行中可包含注释。
    - 用户必须遵照使用批处理的规则.例如,在同一批处理中,创建数据库之后不能直接使用其新建的数据库.局部（用户定义）变量的作用域限制在一个批处理中,不可在 GO 命令后引用。 **所以我在申明一个变量之后 GO，再print这个变量会报错**
- ```

1  Insert [Roc] Select 'aking'
2  GO 10

```
- 批处理执行10次,向表Roc中插入10行记录aking
- 不同批处理是分开执行的,一个查询失败不会影响另外一个查询。

- 用户必须遵照使用批处理的规则。例如，在批处理中的第一条语句后执行任何存储过程必须包含 EXECUTE 关键字。局部（用户定义）变量的作用域限制在一个批处理中，不可在 GO 命令后引用。

```
1 UNIQUEIDENTIFIER
2 DECIMAL
3 declare @num decimal(10, 3)
4 set @num = 13.534535432
5 print @num
6 go
```

[decimal, float, real和numeric](#)

[sql server数据类型](#)

```
1 --尝试使用SQLServer的帮助。选中关键字，按F1查看文档
2 Go: --将T-SQL语句分批发送到数据库实例执行。
3
4 --字符串函数
5 CHARINDEX() --寻找指定的字符串在源字符串中的起始位置:位置从1开始计算，如果找不到就返回0
6 --第一个参数是指你需要查询的字符串 第二个参数是源字符串，第三个参数是指定的查询起始位置
7 SELECT CHARINDEX('JBNS','My Jbns Course',1) --返回: 4
8 LEN() --计算字符串长度（字符的个数），不区分英文
9 SELECT LEN('SQL Server课程') --返回: 12
10 datalength() --计算指定参数的所占据字符串长度。一个英文一个字节，但是一个中文两个字节
11 LOWER()/UPPER ()--转小写/大写
12 SELECT UPPER('sql server课程') --返回: SQL SERVER课程
13 LTRIM() --字符串左侧的空格去掉
14 SELECT LTRIM(' 涛哥 ') --返回: 涛哥（后面的空格保留）
15 RTRIM() --字符串右侧的空格去掉
16 SELECT RTRIM('涛哥 ') --返回: 涛哥（前面的空格保留）
17 LTRIM(RTRIM(' bb '))
18 LEFT()、RIGHT() --可以从指定的字符串右边开始返回指定数目的字符串.数量可以指定任意的正值，但是不可是负值
19 SELECT RIGHT('买卖提.吐尔松',3) --返回: 吐尔松
20 SUBSTRING(string,start_position,length) --索引从1开始。 参数string为主字符串，
start_position为子字符串在主字符串中的起始位置，length为子字符串的最大长度。
21 SELECT SUBSTRING('abcdef111',2,3) --返回: bcd
22 REPLACE() --替换一个字符串中的字符
23 SELECT REPLACE('莫乐可切.杨可','可','兰') --返回: 莫乐兰切.杨兰
24 STUFF() --在一个字符串中，删除指定长度的字符，并在该位置插入一个新的字符串
25 SELECT STUFF('ABCDEFGF', 2, 3, '我的音乐我的世界') --返回: A我的音乐我的世界EFG，从
第2个起替换3个
26
27 --数学函数
28 RAND() ----RAND可以生成一个0~1之前的随机数，包含0，但是不包含1，里面可以写种子值
29 SELECT RAND() --返回: 0.79288062146374
30 ABS() --取数值表达式的绝对值
31 SELECT ABS(-43) --返回: 43
32 CEILING() --取大于或等于指定数值、表达式的最小整数
33 SELECT CEILING(43.5) --返回: 44
34 FLOOR() --取小于或等于指定表达式的最大整数
35 SELECT FLOOR(43.5) --返回: 43
36 POWER() --取数值表达式的幂值
37 SELECT POWER(5,2) --返回: 25
38 ROUND() --将数值表达式四舍五入为指定精度，只关注你指定的小数位后一位数值
39 SELECT ROUND(43.543,1) --返回: 43.500
```

```

40 SIGN() --对于正数返回+1, 对于负数返回-1, 对于0则返回0
41 SELECT SIGN(-43) --返回: -1
42 SQRT() --取浮点表达式的平方根
43 SELECT SQRT(9) --返回: 3
44
45 --时间函数
46 GETDATE() --取得当前的系统日期
47 SELECT GETDATE() --返回: 今天的日期
48 DATEADD() --将指定的数值添加到指定的日期部分后的日期
49 SELECT DATEADD(mm,4,'01/01/2009') --返回: 以当前的日期格式返回05/01/2009
50 DATEDIFF() --两个日期之间的指定日期部分的间隔
51 SELECT DATEDIFF(mm, '01/01/2009', '05/01/2009') --返回: 4
52 DATENAME() --日期中指定日期部分的字符串形式
53 SELECT DATENAME(dw, '01/01/2000') --返回: Saturday或星期六
54 DATEPART() --日期中指定日期部分的整数形式
55 SELECT DATEPART(day, '01/15/2000')返回: 15
56
57 /*datepart 缩写
58 year yy, yyyy
59 quarter qq, q
60 month mm, m
61 dayofyear dy, y
62 day dd, d
63 week wk, ww
64 weekday dw, w
65 hour hh
66 minute mi, n
67 second ss, s
68 millisecond ms
69 microsecond mcs
70 nanosecond ns */
71
72 /*MS SQL时间格式
73 DATE - 格式 YYYY-MM-DD
74 DATETIME - 格式: YYYY-MM-DD HH:MM:SS
75 SMALLDATETIME - 格式: YYYY-MM-DD HH:MM:SS
76 TIMESTAMP - 格式: 唯一的数字
77 */
78 --其他
79 concat()
80 isnull()

```

## 日期函数补充

- `GETDATE()` : 取得当前日期时间
- `DATEADD (datepart , number, date )` , 计算增加以后的日期。参数 `date` 为待计算的日期; 参数 `number` 为增量; 参数 `datepart` 为计量单位, 可选值见备注。 `DATEADD(DAY, 3,date)` 为计算日期`date`的3天后的日期, 而 `DATEADD(MONTH ,-8,date)` 为计算日期`date`的8个月之前的日期。(入职一年以上的员工发1000\$)
- `DATEDIFF (datepart , startdate , enddate )` : 计算两个日期之间的差额。 `datepart` 为计量单位, 可取值参考 `DateAdd`。
- 统计不同入学年数的学生个数:  

```
select DateDiff(year,sInDate,getdate()),count(*) from student Group by DateDiff(year,sInDate,getdate())
```
- `DATEPART (datepart,date)` : 返回一个日期的特定部分

- `Month()`、`year()`、`day()` 来代替。
- 统计学生的生日年份个数：  
`select DatePart(year,sBirthday),count(*) from student group by DatePart(year, sBirthday)`
- 1990年出生的人的个数? `select count(*) from P where year(birthday)=1990`

## create database and table

```

1  --使用sql语句创建数据库和表
2  /*语法
3  create database 数据库名称
4  on primary --在那个文件组上创建.默认是在主文件组上创建 主数据文件
5  (
6      ,: 当它不是一句可以独立执行的sql命令的时候,同时它是一个结构中的某一句就需要添加,
7      name='逻辑名称_data', --逻辑名称一般会有一个后缀,数据文件--data 日志文件--log
8      size=初始大小 --数值不应该包含在''以内
9      fileGrowth=增长方式 --也不能添加''包含
10     maxsize=最大容量,
11     filename='全路径' --最后一句不添加,扩展名: mdf
12 )
13 log on --日志文件
14 (
15     name='逻辑名称_log' --逻辑名称一般会有一个后缀,数据文件--data 日志文件--log
16     size=初始大小 --数值不应该包含在''以内
17     fileGrowth=增长方式 --也不能添加''包含
18     maxsize=最大容量,
19     filename='全路径' --最后一句不添加,扩展名: ldf
20 )*/
21 --先切换当前数据库
22 use master
23 --先判断数据库是否存在,如果存在就先删除 exists就是判断()里面的语句是否返回值,如果有值
  就返回true,否则就是false, sysdatabases是系统所有的数据库,包括master, model, msdb,
  temperdb
24 if exists(select * from sysdatabases where name='TestSchool')
25 --删除数据库
26 drop database TestSchool
27 go
28 --自动创建文件夹 调用存储过程xp_cmdshell,让其帮助我们创建一个文件夹
  d:\mydir\database
29 execute sp_configure 'show advanced options' ,1
30 RECONFIGURE --安装
31 exec sp_configure 'xp_cmdshell',1
32 RECONFIGURE
33 go
34 exec xp_cmdshell 'mkdir d:\mydir\database'
35
36 create database TestSchool
37 on primary --primary可以省略
38 (
39     name='TestSchool_data',
40     size=3mb,
41     filegrowth=10%,
42     maxsize=100mb,
43     filename='d:\mydir\database\TestSchool_data.mdf'
44 ),

```

```

45 filegroup userDe
46 (
47     name='TestSchool_data1',
48     size=3mb,
49     filegrowth=10%,
50     maxsize=100mb,
51     filename='E:\aa\TestSchool_data1.ndf'    --次数据文件
52 )
53 log on
54 (
55     name='TestSchool_log',
56     size=1mb,
57     filegrowth=10%,
58     --maxsize=100mb,--日志文件一般不限制最大容量
59     filename='d:\mydir\database\TestSchool_log.ldf'
60 ),
61 (
62     name='TestSchool_log1',
63     size=1mb,
64     filegrowth=10%,
65     --maxsize=100mb,--日志文件一般不限制最大容量
66     filename='d:\mydir\database\TestSchool_log1.ldf'    --日志文件可以多个
67 )
68
69 --创建数据表
70 /*语法:
71 create table 表名
72 (
73     字段名称 字段类型 字段特征(是否非空 标识列 默认值 主键 唯一键 check 约束 ),
74     字段名称 字段类型 字段特征(是否非空 标识列 默认值 主键 唯一键 check 约束 )
75 )*/
76 ----创建老师表Teacher(Id、Name、Gender、Age、Salary、Birthday)
77 use TestSchool
78 if exists(select * from sysobjects where name='Teacher') --sysobjects不仅有
79 drop table Teacher
80 go
81 create table Teacher
82 (
83     Id int identity(1,1) primary key,--设置标识列 identity(标识种子,标识增量)
84     Name nvarchar(50) not null, --not null标记字段不能为null值.字符类型如果没有指
85     Gender bit not null,
86     ClassId int ,
87     Age int check(age>0 and age<100)
88     Salary money, -- 当一个字段可以为null的时候可以不写也可以写null
89     Birthday datetime not null -- default('2009-9-9')
90 )

```

## 数据完整性

- 1 --数据完整性:
- 2 /\*实体完整性:表的每一行数据就称为一个实体.实体完整性是指每一行记录是唯一的,不重复的
- 3 - 标识列:系统自动生成,永远不会重复
- 4 - 主键:唯一 非空. 一个表的主键只有一个

```

5  - 唯一键：唯一 但是可以为null,只能空一次。一个表的唯一键可以有多个 右键--索引/键--添加--修改名称，修改类型，确定字段
6
7  域完整性：域就是指字段，域完整性就是为了保证字段的值是合理和准确的
8  - 非空，类型，check约束，默认值，关系(主外键约束)
9
10 自定义完整性：用户自己定义的约束规则：
11  - check约束 存储过程 触发器
12
13  - 引用完整性：一个表的某个字段的值引用自另外一个表的某一个字段。被引用的表就称为主表，引用表就是称为从表或者外键表
14 1.选择外键表去创建主外键关系
15 2.建立主外键关系的字段类型和意义必须一致
16 3.建立关系的字段 主表中必须是主键或者唯一键，因为是1对1的关系
17 4.添加数据的时候先添加主表数据,再添加外键表
18 5.删除数据的时候先删除外键表数据再删除主表 数据
19
20 关系建立的表的级联操作：
21 1.不执行任何操作：该报错就报错,能删除就删除
22 2.级联：删除主表记录,对应的从表记录也将被删除
23 3.set null:删除主表，从表对应记录的字段值=null.前提是这个字段可以设置为null
24 4.set default:删除主表，从表对应记录的字段值=设置的默认值.前提是这个字段已经设置了默认值了
25
26 使用代码创建约束：
27 - 种类：主键约束(primary key PK) 唯一键约束(unique UQ) 检查约束(check CK)
   默认值约束(default DF) 外键约束(foreign key FK)
28 - 创建约束的语法：
29 --alter table 表名 add constraint 约束的名称(以简写做为前缀) 约束的类型 约束的说明
   (字段 表达式 值)*/
30
31 --1.将id设置为主键：
32 alter table Teacher add constraint PK_Teacher_id primary key(Id)
33 alter table tablename add primary key(col) --添加主键
34 alter table tablename drop primary key(col) --删除主键
35 --2设置name为唯一键
36 if exists(select * from sysobjects where name='UQ_Teacher_Name')
37 alter table teacher drop constraint UQ_Teacher_Name
38 alter table teacher add constraint UQ_Teacher_Name unique(name)
39 --3.设置年龄0~100之间
40 alter table teacher add constraint CK_Teacher_Age check(age>0 and age
   <=100)
41 --4.为birthday添加默认值约束
42 alter table teacher add constraint DF_Teacher_Birthday default('1990-9-9')
   for birthday --for是说明为那一个字段添加默认值
43 --5.为classid添加外键约束
44 if exists(select * from sysobjects where name='FK_Teacher_classid')
45 alter table teacher drop constraint FK_Teacher_classid
46 alter table teacher --从表的某一个字段引用主表的某一个字段
   with nocheck --不检查现有数据
47 add constraint FK_Teacher_classid foreign key(classId) references
   classes(cid)
48 on delete set null
49 on update set default
50
51
52 --列增加后将不能删除。DB2中列加上后数据类型也不能改变，唯一能改变的是增加varchar类型的长度。
53 ALTER TABLE table_name ADD column_name datatype

```



## 数据插入

```

1  --数据插入
2  /*语法：  方法调用（一 一 对应 顺序对应，数量对应，类型对应）
3  insert [into] 自动编号列不需要手动插入。表名（字段列表） values(值列表)
4  - 自动编号列不需要手动插入。
5  - 说明：标识列值不管什么时候都不可能插入值,同时插入的值需要满足表的所有完整性约束*/
6  --1.为表的所有字段添加值--如果不指定字段列表，那么就默认需要为所有字段添加值
7  insert into Teacher values('张感动1',1,1,20,5000,'1990-8-15')
8  --2.值不能违反表的约束
9  --3.1.也可以指定为那一些列插入值  -列名或所提供值的数目与表定义不匹配。
10 --3.2 INSERT 语句中列的数目大于 VALUES 子句中指定的值的数目。VALUES 子句中值的数目必须与 INSERT 语句中指定的列的数目匹配
11 --4.非空字段一定需要插入值，除非它有默认值
12 --如果一个字段可以为null或者有默认值，那么在插入的时候也可以:可以为空字段赋值null,默认值
   字段赋值default
13 insert into Teacher values('张6',1,6,null,null,'1990-8-15')
14 insert into Teacher values('张7',1,6,null,null,default)
15 --所有值都可以使用''包含，如果字段的类型是数值，那么系统会自动的类型转换
16 insert into Teacher values(N'张8','1','6','30','3000','1990-8-15')
17 --如果字符类型的字段值没有使用'',就：1.如果是非数值字符==报错，如果纯数字字符串--OK
18 insert into Teacher values(8,'1','6','30','3000','1990-8-15')
19 --如果是日期值没有使用''包含，那么就会得系统默认日期
20 insert into Teacher values('张9','1','6','30','3000',1990-8-15)

```

## 数据更新

```

1  --数据更新:更新后的数据不能违反表的约束
2  /*语法:
3  --update 表名 set 字段=值（表达式），字段=值 where 条件(一般能够做条件的是主键，唯一
   键，标识列)*/
4
5  --修改张8的班级为7
6  update Teacher set ClassId=6 where Name='张8'
7  --修改张8 性别 修改为女，同时将年龄修改25 将工资加1000蚊
8  update Teacher set Gender=0,Age=250,Salary+=1000 where Name='张8'
9  --判断多条件 not and or 修改性别是男同时是7班，将工资+500
10 update Teacher set Salary+=500 where Gender=1 and ClassId=7
11 -- where中可以使用的其他逻辑运算符: or、and、not、<、>、>=、<=、<>（不等，
   或!=），between, like等

```

## 数据删除

```

1  --数据删除 不能删除某一列，因为删除是对记录而言
2  /*1.删除是一条一条删除，每一次删除都会将操作写入到日志文件--效率低
3  2.标识列的值不会从种子重新计算
4  3.可以触发触发器

```



```

5  语法:
6  delete [from] 表 where 条件*/
7  --删除姓名为8的人
8  delete from Teacher where Name='8'
9  --多条件删除
10 delete from Teacher where ClassId=6 and Gender=0 and Age>20
11 --删除所有数据
12 delete from Teacher
13
14 /*truncate
15 truncate语句非常高效。由于truncate操作采用按最小方式来记录日志，所以效率非常高。对于数百万条数据使用truncate删除只要几秒钟，而使用delete则可能耗费几小时。
16 1. 一次性删除所有记录，日志文件以最小化的方式写入。效率更高
17 2. 标识列从种子值重新计算
18 3. 不可以触发触发器
19 语法:
20 truncate table 表名 （不能添加条件，因为它不是一条一条删除，而是一次性删除所有记录，不关注删除的条数）*/
21 truncate table teacher

```

## 数据检索

```

1  --数据检索
2  /*语法:
3  select 字段列表 from 表列表 where 条件*/
4  --查询所有信息
5  select * from Teacher,Classes where Teacher.ClassId=Classes.Cid
6  --查询指定的列
7  select Id,Name,Salary from Teacher
8  --指定查询的条件 查询女老师
9  select * from Teacher where Gender=0 and Age<30
10 --between...and... 是大于等于前面的值 小于等于后面的值。not between...and...
11 select * from Student where Sex='女' and BornDate between '1999-1-1' and '1995-1-1'
12 --查询1, 2, 3, 4班的学员信息
13 --in:可以指定一个具体的范围,它可以取其中的任意值. 要求值的类型是一致(值可以相互转换)
14 select * from Student where ClassId in (1,2,'3','4')
15 --为列指定中文别名 --只是结果集的显示。不会修改原始的表结构
16 select Id as 工号,Name 姓名,工资=Salary,公司='传智' from Teacher where Gender=0 and Age<30
17 -- as可以省略,列名=值
18
19 --select 可以输出,只不过输出是结果集--以表的形式显示
20 select 1+1,2,3,4,5
21 select getdate()
22 select top 0 * into newStu from Student
23
24 --Top、Distinct
25 --也可以限百分比
26 select top 10 percent * from Student --不是四舍五入,而是取Ceiling
27 --去除重复值。它的作用与原始的数据表的记录无关,只与当前结果集有关系:处理查询得到的结果集
28 select Sex,Address from Student
29 select distinct Sex,Address from Student
30
31 --聚合函数

```

```

32  --MAX（最大值）、MIN（最小值）、AVG（平均值）、SUM（和）、COUNT（数量：记录的条数。）
33  --查询年龄最小的学员信息
34  select MAX(BornDate) from Student
35  --查询年龄最大的学员信息
36  select min(BornDate) from Student
37  --如果是字符串，那么就按字符串的拼音进行排序，得到最大和最小值
38  select min(StudentName) from Student
39  select MAX(StudentName) from Student
40
41  --sum/avg只能对数值进行计算，对null值不计算
42  --查询学号是10的学号的考试总成绩和平均分
43  select SUM(StudentResult) from Result where StudentNo=10
44  select avg(StudentResult) from Result where StudentNo=10
45
46  --count：得到满足条件的记录数
47  --得到总人数，自动过滤空值
48  select COUNT(*) from Student where Sex='女'
49
50  --带条件的查询-模糊查询
51  --通配符：
52  --%：代码任意个任意字符
53  --_：代表任意一个字符
54  --[]：代表一个具体的范围，能够匹配其中一个字符
55  --^代表取反值，只有一[]中才有意义
56  --=号代表严格的字符串匹配，所以%只是一个字符串，如果需要它是通配符必须使用 like
57  select * from Student where StudentName like '林%'
58  select * from Student where StudentName like '林__'
59  select * from Student where StudentName not like '林%'
60  --查询学号在11~15号的学员信息
61  select * from Student where StudentNo like '1[12345]'
62  select * from Student where StudentNo like '1[1-5]'
63  select * from Student where StudentNo like '[11-15]' --单个字符是1 1-1 5 还可以是0-9 a-z A-Z
64  select * from Student where StudentNo like '1[^1-5]'
65
66  --1.查询六期班所有姓 陈 的学员 int num=GetNum();
67  select classid from grade where classname='六期班'
68  select * from Student where StudentName like '周%' and ClassId=(select classid from grade where classname='六期班')
69  --2.查询所有科目中包含c 字符的科目信息
70  select * from Subject where SubjectName like '%c%'
71  --3.查询office最近一次考试时间
72  select MAX(ExamDate) from Result where SubjectId=(select SubjectId from Subject where SubjectName='office')
73
74  ---空值处理
75  --查询没有电子邮箱的学员信息
76  select * from Students where Email is null
77  select * from Student where Email is not null
78  update Student set Email=null where StudentNo=9
79  --ISNULL如果发现对应的值是null值，则以指定的字符串文本进行替换，仅改变结果值，不改变表数据
80  select StudentNo, StudentName, ISNULL(Email, '没有填写') from Student
81
82  --select 字段列表 from 表列表 where 条件 order by (排序字段列表)对结果集做记录重排
83  select * from Student where sex='女' order by BornDate desc, StudentNo asc

```

## 数据分组-统计信息

- 在使用 `select` 查询的时候,有时需要对数据进行分组汇总(即:将现有的数据按照某列来汇总统计),这时就需要用到 `group by` 语句。`select` 语句中可以使用 `group by` 子句将行划分成较小的组,然后,使用聚合函数返回每一个组的汇总信息。**分组一般都和聚合函数连用。**
- `GROUP BY` 子句必须放到 `WHERE` 语句的之后,`Group By` 与 `Order By` 都是对筛选后的数据进行处理,而 `where` 是用来筛选数据的。  
没有出现在 `GROUP BY` 子句中的列是不能放到 `SELECT` 语句后的列名列表中的(聚合函数中除外)
- 错误: `select sClassId,count(sName),sAge from student group by sClassId`
- 正确: `select sClassId,count(sName),avg(sAge) from student group by sClassId`

### Having 语句-对分好的组做条件筛选

- 注意 `Having` 中不能使用未参与分组的列,`Having` 不能替代 `where`。作用不一样,`Having` 是对组进行过滤。
- `Having` 是 `Group By` 的条件对分组后的数据进行筛选(与 `where` 类似,都是筛选,只不过 `having` 是用来筛选分组后的组的。)
- 在 `where` 中不能使用聚合函数,必须使用 `Having`,`Having` 要位于 `Group By` 之后。
- `Having` 的使用几乎是与 `where` 一样的,也可以用 `in`。
  - `Having count(*) in (5,8,10)`

### SQL语句的执行顺序

5>...`select` 5-1>选择列,5-2> `distinct`,5-3> `top` (确定列)

1>...`from` 表 (确定表)

2>...`where` 条件 (确定行)

3>...`group by` 列 (分组)

4>...`having` 对组来做筛选条件 (对组做筛选)

6>...`order by` 列 (排序,在已经存在的结果集上进行记录的重排)

```
1  --数据分组-统计信息
2  /*select top/distinct字段列表 from 表列表 where 源数据筛选条件 group by 分组字段列表
   having 对分组得到的结果集做筛选 order by 排序字段列表*/
3  --查询男女生的人数
4  select sex, COUNT(*) from Student group by sex
5
6  --查询每一个班级的总人数
7  --与聚合函数一起出现在select后面进行查询的列,只有两种可能性:被聚合 被分组
8  select ClassId,COUNT(*) from Student group by ClassId
9  --查询每一个班级男女生的人数,分组分两次
10 select ClassId,sex,COUNT(*) from Student group by ClassId,sex order by
   ClassId
11 --查询每一个班级男女生的人数,同时只需要显示人数数量超过3人的记录
12 /*
13 1.where里面不能出现聚合函数做为条件--!语法,这点很重要
14 order by必须是select里出现的结果集,因为order by是最后一步
15 2.先执行了where,再执行了select,所以别名where不能使用 --这点也很重要
16 3.having:是对分组统计得到的结果集做筛选的。也不能包含select里出现的别名
```

```

17 */
18 select top 1 ClassId,sex,COUNT(*) as cnt from Student where Email is not
    null group by ClassId,sex having COUNT(*)>3 order by cnt
19
20 --什么时候需要分组--每一个 不同 各自 分别
21 --1.查询每个班级的总学时数，并按照升序排列
22 select classid, SUM(ClassHour) as 总学时 from Subject group by ClassId order
    by SUM(ClassHour)
23 --2.查询每个参加考试的学员的平均分
24 select StudentNo, AVG(StudentResult) from Result group by StudentNo
25 --3.查询每门课程的平均分，并按照降序排列
26 select (select subjectname from subject where subjectid=result.SubjectId),
    AVG(StudentResult) from Result group by SubjectId

```

## 类型转换函数

```

1 --类型转换函数--
2 --cast(expression as data_type)
3 print '我的分数是: '+cast(100 as char(3))
4 --convert(data_type, expression,[style]) --格式是对日期值而言
5 print '我的分数是: '+convert(char(3),100)
6 --输出生日
7 print '我的生日是: '+getdate()
8 print '我的生日是: '+convert(char(20),getdate(),112)
9
10 SELECT FIdNumber,
11 CAST(RIGHT(SNo,3) AS INTEGER) as 后三位的整数形式,
12 CAST(RIGHT(SNo,3) AS INTEGER)+1 as 后三位加1,
13 CONVERT(INTEGER,RIGHT(SNo,3))/2 as 后三位除以2
14 FROM student
15
16 --对日期的转换。转换成各种国家格式的日期。
17 select convert(varchar(20),getdate(),104)
18 --Style的格式，查sql帮助。（输入convert函数查询）
19 --将日期转换为指定格式的字符串。日期→字符串

```

## 一次性插入条记录

- 集合运算符是对两个集合操作的，两个集合必须具有相同的列数，列具有相同的数据类型（至少能隐式转换的），最终输出的集合的列名由第一个集合的列名来确定。（可以用来连接多个结果）
- 联合(union)与连接(join)不一样
- 联合：将多个结果集合并成一个结果集。union(去除重复，相当于默认应用了distinct)、union all
- Union因为要进行重复值扫描，所以效率低，因此如果不是确定要合并重复行，那么就用UNION ALL

```

1 ---查询男女生的总人数
2 select 100, COUNT(*) from Student where Sex='男'
3 union
4 select 'a200',COUNT(*) from Student where Sex='女'
5
6 --union:是用来联合多个结果集的

```

```

7  --1.要求联合的多个结果集有相同数量的字段
8  --2.要求联合的多个结果集对应的列的类型需要一致：所谓的类型一致是指他们可以互相转换
9  select 100, COUNT(*) from Student where Sex='男'
10 union all
11 select 100, COUNT(*) from Student where Sex='男'
12 --union默认是去除重复值的,效率低, 是因为需要为你做是否重复的判断
13 --union all就是不去除重复
14
15 --使用union一次插入多条记录
16 --union还是可以去重复记录, 只有全部都使用union all才不考虑重复值
17 insert into Admin
18 select 'fasdf','fasdf' union all
19 select 'fasdf','fasdf' union all
20 select 'adsfasdf','fsdfasdf'
21
22 --- select into from:可以将from数据源表中的select指定的列的数据into到新表中, 新表是
    系统自行生成的, 不能先人为创建, 也就不能先存在. 新表中列的属性只保留标识列, 其余都消失
23 select * into newtable from Student
24 SELECT * INTO Persons IN 'Backup.mdb' FROM Persons
25 delete from newtable -- 可以通过truncate使标识列从1开始
26 --insert into 表 select from :可以将select查询语句中获取的数据into到指定的表中。表
    需要先存在
27 insert into newtable select
    LoginPwd,StudentName,Sex,ClassId,Phone,Address,BornDate,Email,isDel from
    Student

```