**SQL语句大全**

**--语 句 功 能**

--数据操作

SELECT --从数据库表中检索数据行和列

INSERT --向数据库表添加新数据行

DELETE --从数据库表中删除数据行

UPDATE --更新数据库表中的数据

--数据定义

CREATE TABLE --创建一个数据库表

DROP TABLE --从数据库中删除表

ALTER TABLE --修改数据库表结构

CREATE VIEW --创建一个视图

DROP VIEW --从数据库中删除视图

CREATE INDEX --为数据库表创建一个索引

DROP INDEX --从数据库中删除索引

CREATE PROCEDURE --创建一个存储过程

DROP PROCEDURE --从数据库中删除存储过程

CREATE TRIGGER --创建一个触发器

DROP TRIGGER --从数据库中删除触发器

CREATE SCHEMA --向数据库添加一个新模式

DROP SCHEMA --从数据库中删除一个模式

CREATE DOMAIN --创建一个数据值域

ALTER DOMAIN --改变域定义

DROP DOMAIN --从数据库中删除一个域

--数据控制

GRANT --授予用户访问权限

DENY --拒绝用户访问

REVOKE --解除用户访问权限

--事务控制

COMMIT --结束当前事务

ROLLBACK --中止当前事务

SET TRANSACTION --定义当前事务数据访问特征

--程序化SQL

DECLARE --为查询设定游标

EXPLAN --为查询描述数据访问计划

OPEN --检索查询结果打开一个游标

FETCH --检索一行查询结果

CLOSE --关闭游标

PREPARE --为动态执行准备SQL 语句

EXECUTE --动态地执行SQL 语句

DESCRIBE --描述准备好的查询

---局部变量

declare @id char(10)

--set @id = '10010001'

select @id = '10010001'

---全局变量

---必须以@@开头

--IF ELSE

declare @x int @y int @z int

select @x = 1 @y = 2 @z=3

if @x > @y

print 'x > y' --打印字符串'x > y'

else if @y > @z

print 'y > z'

else print 'z > y'

--CASE

use pangu

update employee

set e\_wage =

case

when job\_level = ’1’ then e\_wage\*1.08

when job\_level = ’2’ then e\_wage\*1.07

when job\_level = ’3’ then e\_wage\*1.06

else e\_wage\*1.05

end

--WHILE CONTINUE BREAK

declare @x int @y int @c int

select @x = 1 @y=1

while @x < 3

begin

print @x --打印变量x 的值

while @y < 3

begin

select @c = 100\*@x + @y

print @c --打印变量c 的值

select @y = @y + 1

end

select @x = @x + 1

select @y = 1

end

--WAITFOR

--例 等待1 小时2 分零3 秒后才执行SELECT 语句

waitfor delay ’01:02:03’

select \* from employee

--例 等到晚上11 点零8 分后才执行SELECT 语句

waitfor time ’23:08:00’

select \* from employee

**\*\*\*SELECT\*\*\***

select \*(列名) from table\_name(表名) where column\_name operator value

ex:(宿主)

select \* from stock\_information where stockid = str(nid)

stockname = 'str\_name'

stockname like '% find this %'

stockname like '[a-zA-Z]%' --------- ([]指定值的范围)

stockname like '[^F-M]%' --------- (^排除指定范围)

--------- 只能在使用like关键字的where子句中使用通配符)

or stockpath = 'stock\_path'

or stocknumber < 1000

and stockindex = 24

not stock\*\*\* = 'man'

stocknumber between 20 and 100

stocknumber in(10,20,30)

order by stockid desc(asc) --------- 排序，desc-降序，asc-升序

order by 1,2 --------- by列号

stockname = (select stockname from stock\_information where stockid = 4)

--------- 子查询

--------- 除非能确保内层select只返回一个行的值，

--------- 否则应在外层where子句中用一个in限定符

select distinct column\_name form table\_name --------- distinct指定检索独有的列值，不重复

select stocknumber ,"stocknumber + 10" = stocknumber + 10 from table\_name

select stockname , "stocknumber" = count(\*) from table\_name group by stockname

--------- group by 将表按行分组,指定列中有相同的值

having count(\*) = 2 --------- having选定指定的组

select \*

from table1, table2

where table1.id \*= table2.id -------- 左外部连接，table1中有的而table2中没有得以null表示

table1.id =\* table2.id -------- 右外部连接

select stockname from table1

union [all] ----- union合并查询结果集，all-保留重复行

select stockname from table2

**\*\*\*insert\*\*\***

insert into table\_name (Stock\_name,Stock\_number) value ("xxx","xxxx")

value (select Stockname , Stocknumber from Stock\_table2)---value为select语句

**\*\*\*update\*\*\***

update table\_name set Stockname = "xxx" [where Stockid = 3]

Stockname = default

Stockname = null

Stocknumber = Stockname + 4

**\*\*\*delete\*\*\***

delete from table\_name where Stockid = 3

truncate table\_name ----------- 删除表中所有行，仍保持表的完整性

drop table table\_name --------------- 完全删除表

**\*\*\*alter table\*\*\* --- 修改数据库表结构**

alter table database.owner.table\_name add column\_name char(2) null .....

sp\_help table\_name ---- 显示表已有特征

create table table\_name (name char(20), age smallint, lname varchar(30))

insert into table\_name select ......... ----- 实现删除列的方法（创建新表）

alter table table\_name drop constraint Stockname\_default ---- 删除Stockname的default约束

**\*\*\*function(/\*常用函数\*/)\*\*\***

----统计函数----

AVG --求平均值

COUNT --统计数目

MAX --求最大值

MIN --求最小值

SUM --求和

--AVG

use pangu

select avg(e\_wage) as dept\_avgWage

from employee

group by dept\_id

--MAX

--求工资最高的员工姓名

use pangu

select e\_name

from employee

where e\_wage =

(select max(e\_wage)

from employee)

--STDEV()

--STDEV()函数返回表达式中所有数据的标准差

--STDEVP()

--STDEVP()函数返回总体标准差

--VAR()

--VAR()函数返回表达式中所有值的统计变异数

--VARP()

--VARP()函数返回总体变异数

**----算术函数----**

/\*\*\*三角函数\*\*\*/

SIN(float\_expression) --返回以弧度表示的角的正弦

COS(float\_expression) --返回以弧度表示的角的余弦

TAN(float\_expression) --返回以弧度表示的角的正切

COT(float\_expression) --返回以弧度表示的角的余切

/\*\*\*反三角函数\*\*\*/

ASIN(float\_expression) --返回正弦是FLOAT 值的以弧度表示的角

ACOS(float\_expression) --返回余弦是FLOAT 值的以弧度表示的角

ATAN(float\_expression) --返回正切是FLOAT 值的以弧度表示的角

ATAN2(float\_expression1,float\_expression2)

--返回正切是float\_expression1 /float\_expres-sion2的以弧度表示的角

DEGREES(numeric\_expression)

--把弧度转换为角度返回与表达式相同的数据类型可为

--INTEGER/MONEY/REAL/FLOAT 类型

RADIANS(numeric\_expression) --把角度转换为弧度返回与表达式相同的数据类型可为

--INTEGER/MONEY/REAL/FLOAT 类型

EXP(float\_expression) --返回表达式的指数值

LOG(float\_expression) --返回表达式的自然对数值

LOG10(float\_expression)--返回表达式的以10 为底的对数值

SQRT(float\_expression) --返回表达式的平方根

/\*\*\*取近似值函数\*\*\*/

CEILING(numeric\_expression) --返回>=表达式的最小整数返回的数据类型与表达式相同可为

--INTEGER/MONEY/REAL/FLOAT 类型

FLOOR(numeric\_expression) --返回<=表达式的最小整数返回的数据类型与表达式相同可为

--INTEGER/MONEY/REAL/FLOAT 类型

ROUND(numeric\_expression) --返回以integer\_expression 为精度的四舍五入值返回的数据

--类型与表达式相同可为INTEGER/MONEY/REAL/FLOAT 类型

ABS(numeric\_expression) --返回表达式的绝对值返回的数据类型与表达式相同可为

--INTEGER/MONEY/REAL/FLOAT 类型

SIGN(numeric\_expression) --测试参数的正负号返回0 零值1 正数或-1 负数返回的数据类型

--与表达式相同可为INTEGER/MONEY/REAL/FLOAT 类型

PI() --返回值为π 即3.1415926535897936

RAND([integer\_expression]) --用任选的[integer\_expression]做种子值得出0-1 间的随机浮点数

**----字符串函数----**

ASCII() --函数返回字符表达式最左端字符的ASCII 码值

CHAR() --函数用于将ASCII 码转换为字符

--如果没有输入0 ~ 255 之间的ASCII 码值CHAR 函数会返回一个NULL 值

LOWER() --函数把字符串全部转换为小写

UPPER() --函数把字符串全部转换为大写

STR() --函数把数值型数据转换为字符型数据

LTRIM() --函数把字符串头部的空格去掉

RTRIM() --函数把字符串尾部的空格去掉

LEFT(),RIGHT(),SUBSTRING() --函数返回部分字符串

CHARINDEX(),PATINDEX() --函数返回字符串中某个指定的子串出现的开始位置

SOUNDEX() --函数返回一个四位字符码

--SOUNDEX函数可用来查找声音相似的字符串但SOUNDEX函数对数字和汉字均只返回0 值

DIFFERENCE() --函数返回由SOUNDEX 函数返回的两个字符表达式的值的差异

--0 两个SOUNDEX 函数返回值的第一个字符不同

--1 两个SOUNDEX 函数返回值的第一个字符相同

--2 两个SOUNDEX 函数返回值的第一二个字符相同

--3 两个SOUNDEX 函数返回值的第一二三个字符相同

--4 两个SOUNDEX 函数返回值完全相同

QUOTENAME() --函数返回被特定字符括起来的字符串

/\*select quotename('abc', '{') quotename('abc')

运行结果如下

----------------------------------{

{abc} [abc]\*/

REPLICATE() --函数返回一个重复character\_expression 指定次数的字符串

/\*select replicate('abc', 3) replicate( 'abc', -2)

运行结果如下

----------- -----------

abcabcabc NULL\*/

REVERSE() --函数将指定的字符串的字符排列顺序颠倒

REPLACE() --函数返回被替换了指定子串的字符串

/\*select replace('abc123g', '123', 'def')

运行结果如下

----------- -----------

abcdefg\*/

SPACE() --函数返回一个有指定长度的空白字符串

STUFF() --函数用另一子串替换字符串指定位置长度的子串

----数据类型转换函数----

CAST() 函数语法如下

CAST() (<expression> AS <data\_ type>[ length ])

CONVERT() 函数语法如下

CONVERT() (<data\_ type>[ length ], <expression> [, style])

select cast(100+99 as char) convert(varchar(12), getdate())

运行结果如下

------------------------------ ------------

199 Jan 15 2000

**----日期函数----**

DAY() --函数返回date\_expression 中的日期值

MONTH() --函数返回date\_expression 中的月份值

YEAR() --函数返回date\_expression 中的年份值

DATEADD(<datepart> ,<number> ,<date>)

--函数返回指定日期date 加上指定的额外日期间隔number 产生的新日期

DATEDIFF(<datepart> ,<number> ,<date>)

--函数返回两个指定日期在datepart 方面的不同之处

DATENAME(<datepart> , <date>) --函数以字符串的形式返回日期的指定部分

DATEPART(<datepart> , <date>) --函数以整数值的形式返回日期的指定部分

GETDATE() --函数以DATETIME 的缺省格式返回系统当前的日期和时间

**----系统函数----**

APP\_NAME() --函数返回当前执行的应用程序的名称

COALESCE() --函数返回众多表达式中第一个非NULL 表达式的值

COL\_LENGTH(<'table\_name'>, <'column\_name'>) --函数返回表中指定字段的长度值

COL\_NAME(<table\_id>, <column\_id>) --函数返回表中指定字段的名称即列名

DATALENGTH() --函数返回数据表达式的数据的实际长度

DB\_ID(['database\_name']) --函数返回数据库的编号

DB\_NAME(database\_id) --函数返回数据库的名称

HOST\_ID() --函数返回服务器端计算机的名称

HOST\_NAME() --函数返回服务器端计算机的名称

IDENTITY(<data\_type>[, seed increment]) [AS column\_name])

--IDENTITY() 函数只在SELECT INTO 语句中使用用于插入一个identity column列到新表中

/\*select identity(int, 1, 1) as column\_name

into newtable

from oldtable\*/

ISDATE() --函数判断所给定的表达式是否为合理日期

ISNULL(<check\_expression>, <replacement\_value>) --函数将表达式中的NULL 值用指定值替换

ISNUMERIC() --函数判断所给定的表达式是否为合理的数值

NEWID() --函数返回一个UNIQUEIDENTIFIER 类型的数值

NULLIF(<expression1>, <expression2>)

--NULLIF 函数在expression1 与expression2 相等时返回NULL 值若不相等时则返回expression1 的值

**sql中的保留字**

action add aggregate all

alter after and as

asc avg avg\_row\_length auto\_increment

between bigint bit binary

blob bool both by

cascade case char character

change check checksum column

columns comment constraint create

cross current\_date current\_time current\_timestamp

data database databases date

datetime day day\_hour day\_minute

day\_second dayofmonth dayofweek dayofyear

dec decimal default delayed

delay\_key\_write delete desc describe

distinct distinctrow double drop

end else escape escaped

enclosed enum explain exists

fields file first float

float4 float8 flush foreign

from for full function

global grant grants group

having heap high\_priority hour

hour\_minute hour\_second hosts identified

ignore in index infile

inner insert insert\_id int

integer interval int1 int2

int3 int4 int8 into

if is isam join

key keys kill last\_insert\_id

leading left length like

lines limit load local

lock logs long longblob

longtext low\_priority max max\_rows

match mediumblob mediumtext mediumint

middleint min\_rows minute minute\_second

modify month monthname myisam

natural numeric no not

null on optimize option

optionally or order outer

outfile pack\_keys partial password

precision primary procedure process

processlist privileges read real

references reload regexp rename

replace restrict returns revoke

rlike row rows second

select set show shutdown

smallint soname sql\_big\_tables sql\_big\_selects

sql\_low\_priority\_updates sql\_log\_off sql\_log\_update sql\_select\_limit

sql\_small\_result sql\_big\_result sql\_warnings straight\_join

starting status string table

tables temporary terminated text

then time timestamp tinyblob

tinytext tinyint trailing to

type use using unique

unlock unsigned update usage

values varchar variables varying

varbinary with write when

where year year\_month zerofill

**常用SQL命令和ASP编程**

在进行数据库操作时，无非就是添加、删除、修改，这得设计到一些常用的SQL语句，如下：

SQL常用命令使用方法：

**(1) 数据记录筛选：**

sql="select \* from 数据表 where 字段名=字段值 order by 字段名 [desc]"

sql="select \* from 数据表 where 字段名 like %字段值% order by 字段名 [desc]"

sql="select top 10 \* from 数据表 where 字段名 order by 字段名 [desc]"

sql="select \* from 数据表 where 字段名 in (值1,值2,值3)"

sql="select \* from 数据表 where 字段名 between 值1 and 值2"

**(2) 更新数据记录：**

sql="update 数据表 set 字段名=字段值 where 条件表达式"

sql="update 数据表 set 字段1=值1,字段2=值2 …… 字段n=值n where 条件表达式"

**(3) 删除数据记录：**

sql="delete from 数据表 where 条件表达式"

sql="delete from 数据表" (将数据表所有记录删除)

**(4) 添加数据记录：**

sql="insert into 数据表 (字段1,字段2,字段3 …) valuess (值1,值2,值3 …)"

sql="insert into 目标数据表 select \* from 源数据表" (把源数据表的记录添加到目标数据表)

**(5) 数据记录统计函数：**

AVG(字段名) 得出一个表格栏平均值

COUNT(\*|字段名) 对数据行数的统计或对某一栏有值的数据行数统计

MAX(字段名) 取得一个表格栏最大的值

MIN(字段名) 取得一个表格栏最小的值

SUM(字段名) 把数据栏的值相加

引用以上函数的方法：

sql="select sum(字段名) as 别名 from 数据表 where 条件表达式"

set rs=conn.excute(sql)

用 rs("别名") 获取统的计值，其它函数运用同上。

**(6) 数据表的建立和删除：**

CREATE TABLE 数据表名称(字段1 类型1(长度),字段2 类型2(长度) …… )

例：CREATE TABLE tab01(name varchar(50),datetime default now())

DROP TABLE 数据表名称 (永久性删除一个数据表)

**在ASP编程时，下面这些语句是必须知道的：**

**1.连接数据库**

a. ASP与Access数据库连接：

<%@ language=VBs cript%>

<%

dim conn,mdbfile

mdbfile=server.mappath("数据库名称.mdb")

set conn=server.createobject("adodb.connection")

conn.open "driver={microsoft access driver (\*.mdb)};uid=admin;pwd=数据库密码;dbq="&mdbfile

%>

b. ASP与SQL数据库连接：

<%@ language=VBs cript%>

<%

dim conn

set conn=server.createobject("ADODB.connection")

con.open "PROVIDER=SQLOLEDB;DATA SOURCE=SQL服务器名称或IP地址;UID=sa;PWD=数据库密码;DATABASE=数据库名称

%>

建立记录集对象：

set rs=server.createobject("adodb.recordset")

rs.open SQL语句,conn,3,2

**2. 记录集对象的方法：**

rs.movenext 将记录指针从当前的位置向下移一行

rs.moveprevious 将记录指针从当前的位置向上移一行

rs.movefirst 将记录指针移到数据表第一行

rs.movelast 将记录指针移到数据表最后一行

rs.absoluteposition=N 将记录指针移到数据表第N行

rs.absolutepage=N 将记录指针移到第N页的第一行

rs.pagesize=N 设置每页为N条记录

rs.pagecount 根据 pagesize 的设置返回总页数

rs.recordcount 返回记录总数

rs.bof 返回记录指针是否超出数据表首端，true表示是，false为否

rs.eof 返回记录指针是否超出数据表末端，true表示是，false为否

rs.delete 删除当前记录，但记录指针不会向下移动

rs.addnew 添加记录到数据表末端

rs.update 更新数据表记录

**附：**

**常数 常数值 说明**

－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－－

adLockReadOnly 1 缺省值，Recordset对象以只读方式启动，无法运行AddNew、Update及Delete等方法

adLockPrssimistic 2 当数据源正在更新时，系统会暂时锁住其他用户的动作，以保持数据一致性。

adLockOptimistic 3 当数据源正在更新时，系统并不会锁住其他用户的动作，其他用户可以对数据进行增、删、改的操作。

adLockBatchOptimistic 4 当数据源正在更新时，其他用户必须将CursorLocation属性改为adUdeClientBatch才能对数据进行增、删、改的操作。

**MSSQL经典语句**

**1.按姓氏笔画排序:**Select \* From TableName Order By CustomerName Collate Chinese\_PRC\_Stroke\_ci\_as

**2.数据库加密:**select encrypt('原始密码')

select pwdencrypt('原始密码')

select pwdcompare('原始密码','加密后密码') = 1--相同；否则不相同 encrypt('原始密码')

select pwdencrypt('原始密码')

select pwdcompare('原始密码','加密后密码') = 1--相同；否则不相同

**3.取回表中字段:**declare @list varchar(1000),@sql nvarchar(1000)

select @list=@list+','+b.name from sysobjects a,syscolumns b where a.id=b.id and a.name='表A'

set @sql='select '+right(@list,len(@list)-1)+' from 表A'

exec (@sql)

**4.查看硬盘分区:**EXEC master..xp\_fixeddrives

**5.比较A,B表是否相等:**if (select checksum\_agg(binary\_checksum(\*)) from A)

=

(select checksum\_agg(binary\_checksum(\*)) from B)

print '相等'

else

print '不相等'

**6.杀掉所有的事件探察器进程:**DECLARE hcforeach CURSOR GLOBAL FOR SELECT 'kill '+RTRIM(spid) FROM master.dbo.sysprocesses

WHERE program\_name IN('SQL profiler',N'SQL 事件探查器')

EXEC sp\_msforeach\_worker '?'

**7.记录搜索:开头到N条记录**

Select Top N \* From 表

-------------------------------

**N到M条记录(要有主索引ID)**Select Top M-N \* From 表 Where ID in (Select Top M ID From 表) Order by ID Desc

----------------------------------

**N到结尾记录**

Select Top N \* From 表 Order by ID Desc

**8.如何修改数据库的名称:**sp\_renamedb 'old\_name', 'new\_name'

9：获取当前数据库中的所有用户表select Name from sysobjects where xtype='u' and status>=0

10：获取某一个表的所有字段select name from syscolumns where id=object\_id('表名')

11：查看与某一个表相关的视图、存储过程、函数select a.\* from sysobjects a, syscomments b where a.id = b.id and b.text like '%表名%'

12：查看当前数据库中所有存储过程select name as 存储过程名称 from sysobjects where xtype='P'

13：查询用户创建的所有数据库select \* from master..sysdatabases D where sid not in(select sid from master..syslogins where name='sa')

或者

select dbid, name AS DB\_NAME from master..sysdatabases where sid <> 0x01

14：查询某一个表的字段和数据类型select column\_name,data\_type from information\_schema.columns

where table\_name = '表名'

**[n].[标题]:**Select \* From TableName Order By CustomerName

**[n].[标题]:**Select \* From TableName Order By CustomerName

from：http://tb.blog.csdn.net/TrackBack.aspx?PostId=585515

**触发器-MSSQL常用操作**

发表人:kendy517 | 发表时间: 2007年二月09日, 09:18

不再新开文章.

这里只打算讲解四部分了，也就最简单、最常用的四部分。

1、触发器。

定义： 何为触发器？在SQL Server里面也就是对某一个表的一定的操作，触发某种条件，从而执行的一段程序。触发器是一个特殊的存储过程。

常见的触发器有三种：分别应用于Insert , Update , Delete 事件。(SQL Server 2000定义了新的触发器，这里不提）

我为什么要使用触发器？比如，这么两个表：

Create Table Student( --学生表

StudentID int primary key, --学号

....

)

Create Table BorrowRecord( --学生借书记录表

BorrowRecord int identity(1,1), --流水号

StudentID int , --学号

BorrowDate datetime, --借出时间

ReturnDAte Datetime, --归还时间

...

)

用到的功能有:

1.如果我更改了学生的学号,我希望他的借书记录仍然与这个学生相关(也就是同时更改借书记录表的学号);

2.如果该学生已经毕业，我希望删除他的学号的同时，也删除它的借书记录。

等等。

这时候可以用到触发器。对于1，创建一个Update触发器：

Create Trigger truStudent

On Student

for Update

As

if Update(StudentID)

begin

Update BorrowRecord

Set StudentID=i.StudentID

From BorrowRecord br , Deleted d ,Inserted i

Where br.StudentID=d.StudentID

end

理解触发器里面的两个临时的表：Deleted , Inserted 。注意Deleted 与Inserted分别表示触发事件的表“旧的一条记录”和“新的一条记录”。

一个Update 的过程可以看作为：生成新的记录到Inserted表，复制旧的记录到Deleted表，然后删除Student记录并写入新纪录。

对于2，创建一个Delete触发器

Create trigger trdStudent

On Student

for Delete

As

Delete BorrowRecord

From BorrowRecord br , Delted d

Where br.StudentID=d.StudentID

从这两个例子我们可以看到了触发器的关键：A.2个临时的表；B.触发机制。

这里我们只讲解最简单的触发器。复杂的容后说明。

事实上，我不鼓励使用触发器。触发器的初始设计思想，已经被“级联”所替代.

来自:<http://www.innerv.com/blogview.asp?logID=512>

**MSSQL数据库导出和导入**

**SQL Server导出导入数据方法**

**一、导出导入SQL Server里某个数据库**

1.在SQL Server企业管理器里选中要转移的数据库，按鼠标右键，选所有任务->备份数据库。 2.备份 选数据库-完全， 目的 备份到 按添加按钮 文件名 在SQL Server服务器硬盘下输入一个自定义的备份数据库文件名(后缀一般是bak) 重写 选重写现有媒体 最后按确定按钮。 如果生成的备份数据库文件大于1M，要用压缩工具压缩后再到Internet上传输。 3.通过FTP或者remote desktop或者pcanywhere等方法 把第二步生成的备份数据库文件或者其压缩后的文件传到目的SQL Server数据库，如果有压缩要解压。 4.目的SQL Server数据库如果还没有此数据库，先创建一个新的数据库； 然后选中这个新创建的数据库，按鼠标右键，选所有任务->还原数据库 还原->从设备->选择设备->磁盘->添加（找到要导入的备份数据库文件名）->确定 还原备份集->数据库-完全 最后按确定按钮。完全的数据库导入成功了。 (如果在已经存在的SQL Server数据库上还原数据库可能遇到有还有其它人正在使用它而恢复操做失败， 可以去看 ->管理->当前活动->锁/对象->找到数据库下锁的进程号->到查询分析器里用kill 进程号杀掉这些锁， 然后再做还原) 注意：如果在原有的目的SQL Server数据库上从备份文件(\*.bak)还原数据库 会把已经存在的表、存储过程等数据库对象全部替换成最近这次导入的备份数据库里的内容。 如果一定要还原备份文件(\*.bak)里部分数据,需要另外建一个新数据库, 其逻辑名称和数量同备份文件(\*.bak)里数据库的逻辑名称和数量一致； 新数据库的物理文件名称取得一定要和备份文件(\*.bak)里数据库的物理文件不一样才行。

**二、导出导入SQL Server里某个表**

1.没有防火墙，同一个局域网里或不在同一个局域网里,但通过Internet可以互相访问 在SQL Server企业管理器里选中目的数据库 ，按鼠标右键，选所有任务->导入数据-> 弹出数据转换服务导入/导出向导窗口->下一步-> 选数据源-> 数据源（用于SQL Server的Microfost OLE DB提供程序）-> 服务器(可选择局域网内能访问到的所有SQL Server服务器,或者直接输入IP地址)-> 选择使用windows身份验证还是使用SQL Serve身份验证(输入数据库的用户名和密码)-> 数据库(可选择上面选中SQL Server服务器上所有权限范围内的数据库)->下一步-> 选择目的->目的（用于SQL Server的Microfost OLE DB提供程序）-> 服务器(默认为上一步里选中的导出服务器，也可以选其它局域网内能访问到的所有SQL Server服务器,或者直接输入IP地址)-> 目的数据库(可选择上面选中SQL Server服务器上所有权限范围内的数据库)->下一步-> 制定表复制或查询->选从源数据库复制表和视图(也可以选择用一条查询指定要传输的数据)->下一步-> 选择源表和视图->在要导入的表和视图前面选中源->目的出现同样的表名(可以手工修改成别的表名)-> 转换->列映射和转换里面可以修改源表和目的表之间字段的对应关系，修改目的表字段的类型和长度等， 并可以选择创建目的表，在目的表中增加行，除去并重新创建目的表，启用标志插入等选项->确定->下一步-> 保存、调度和复制包->时间->立即运行（如果要实现隔一段时间自动导出导入数据，选调度DTS包以便以后执行）-> 保存(可以不选)->[ 保存DTS包(如果以后还要转移这批相同的数据，可以把本次导出导入的内容和步骤保存起来， 存到SQL Server即可，保存的时候要输入DTS的包名及详细描述)->下一步-> ]->完成 正在执行包->图形界面显示创建表及插入记录的步骤和状态->完成 2.经过防火墙，不在同一个局域网里 ①、导出表里的数据到文本文件： 在SQL Server企业管理器里选中目的数据库，按鼠标右键，选所有任务-> 导入数据->弹出数据转换服务导入/导出向导窗口->下一步-> 选数据源-> 数据源（用于SQL Server的Microfost OLE DB提供程序）-> 服务器(可选择局域网内能访问到的所有SQL Server服务器)-> 选择使用windows身份验证还是使用SQL Serve身份验证(输入数据库的用户名和密码)-> 数据库(可选择上面选中SQL Server服务器上所有权限范围内的数据库)->下一步-> 选择目的->目的(文本文件)-> 文件名(在自己的电脑硬盘中生成一个自定义的文本文件) ->下一步-> 制定表复制或查询->选从源数据库复制表和视图(也可以选择用一条查询指定要传输的数据)->下一步-> 选择目的文件格式->源(选择要导出的表)->用默认的带分隔符->选第一行包含有列名称选项->下一步-> 保存、调度和复制包->时间->立即运行（如果要实现隔一段时间自动导出到文本文件，选调度DTS包以便以后执行）-> 保存(可以不选)-> [保存DTS包(保存的时候要输入DTS的包名及详细描述)->下一步->]->完成 正在执行包->图形界面显示表到文本文件的步骤和状态->完成 如果生成的文本文件大于1M，要用压缩工具压缩后再到Internet上传输。 ②、通过FTP或者remote desktop或者pcanywhere等方法把 第①步生成的文本文件或者其压缩后的文件传到目的SQL Server数据库，如果有压缩要解压。 ③、把文本文件导入目的SQL Server数据库 直接把文本文件导入目的SQL Server数据库里跟文本文件同名的新表名时，默认的会把所有字段类型都变成字符串。 所以我们要这样做： 在源SQL Server数据库上先生成创建表的sql语句 在SQL Server查询分析器里->选中源数据库里表名->按右键->在新窗口中编写对象脚本->创建-> 复制下新窗口内创建表名的sql语句 到目标SQL Server数据库上查询分析器里执行创建表名的sql语句，生成空表结构。 (如果已经存在这样的表名，修改建表的sql语句，在表名后面加上导入时间的年月信息，例如table\_0113) 调用导入/导出工具->弹出数据转换服务导入/导出向导窗口->下一步-> 选数据源-> 数据源（文本文件）-> 文件名(已传到目的SQL Server数据库下要导入的文本文件，后缀可以不是\*.txt， 但是常规文本编辑器能打开的文件，文件类型选全部)->下一步-> 选择文件格式->用默认的带分隔符->选第一行包含有列名称选项->下一步-> 制定列分割符->逗号->下一步-> 选择目的->目的(用于SQL Server的Microfost OLE DB提供程序)-> 服务器(可选择目标局域网内能访问到的所有SQL Server服务器)-> 选择使用windows身份验证还是使用SQL Serve身份验证(输入数据库的用户名和密码)-> 数据库(可选择上面选中SQL Server服务器上所有权限范围内的数据库)->下一步-> 选择源表和视图->修改目的表名为刚才创建的表名->转换(在目的表中追加行) ->下一步-> 保存、调度和复制包-> 时间->立即运行（如果要实现隔一段时间自动把文本文件导入，选调度DTS包以便以后执行）-> 保存(可以不选)-> [保存DTS包(保存的时候要输入DTS的包名及详细描述)->下一步->]->完成 正在执行包->图形界面显示文本文件到表的步骤和状态->完成 如果要更改导入时间的年月信息的表名，例如table\_0113到原来的表名， 在企业管理器里把原来的表名改成table\_old\_0113，table\_0113改名成table。 这会对应用程序里频繁访问的表照成一定的中断。 注意：源表上的建的索引和主键约束不能用上面介绍的1和2方法转移过来，还需要手工来建索引和主键。 标志种子和not null的约束可以继承过来。 导入视图时会把源视图里所有的真实数据导入成一个新表，而不是视图。

**三、SQL Server存储过程或用户定义的函数导出导入**

1、导出存储过程或用户定义的函数成\*.sql文件 在SQL Server企业管理器里选中源数据库， 存储过程->单选或者多选中要转移的存储过程-> 用户定义的函数->单选或者多选中要转移的函数-> 按鼠标右键，选所有任务->生成SQL脚本->确定->在自己的电脑硬盘中生成一个自定义的\*.sql文件-> 保存->正在生成SQL脚本->成功 2、如果目的数据库经过防火墙，不在同一个局域网里， 要通过FTP或者remote desktop或者pcanywhere等方法把第1步生成的\*.sql文件传到目的SQL Server数据库服务器上。 3、用查询分析器进入SQL Server目的数据库， 从菜单里选文件->打开->打开查询文件->选中第1步生成的\*.sql文件->点执行查询的绿色倒三角型快捷键-> 查询窗口里会出现执行后的消息（有时候可能因为存储过程和用户定义的函数之间有一定的依赖关系，会报一些错。 最好先执行用户定义的函数的\*.sql文件，再执行存储过程的\*.sql文件）

**四、ORACLE数据库里表导入SQL Server数据库**

1、在目的SQL Server数据库服务器上安装ORACLE Client软件或者ORACLE ODBC Driver. 在$ORACLE\_HOMEnetworkadmintnsnames.ora里配置ORACLE数据库的别名(service name)。 具体配置方法可以参考本站文章:[客户端连服务器的注意事项](http://kendy517.itpub.net/client.htm) 2、在WIN2000或者win2003服务器->管理工具->数据源(ODBC)-> 系统DSN（本机器上NT域用户都可以用）->添加->ORACLE ODBC Driver->完成-> data source name 可以自定义，我一般填ORACLE数据库的sid标志， description里可以填ORACLE数据库详细描述，也可以不填-> data source service name 填第1步定义的ORACLE数据库别名->OK。 （用户DSN和文件DSN也可以类似配置，但使用的时候有一些限制） 3、SQL Server的导入和导出数据工具里->选数据源-> 数据源（其它（ODBC数据源））-> 选第2步在ODBC里定义的系统DSN source name，用户名密码处填写ORACLE系统的用户名和密码-> 下一步->选择目的，选SQL Server数据库（跟上面第二点讲的一致，就不重复了）。 注意：在ORACLE表和SQL Server表之间'转换'那步很重要， 可以改变默认的字段数据类型，如image->text,decimal->int

**五、SQL Server数据库里表导入ORACLE数据库**

方法一.导出目的选通过ODBC数据源里定义的ORACLE数据库, 注意ORACLE里表名都是大写的. 我一般在ORACLE这边先生成好表结构,再选择SQL SERVER源表往ORACLE目的表里追加数据. 数据传输速度比方法二慢. 方法二.从SQL Server数据库导入数据到ORACLE数据库可以选择用Windows下ORACLE9i企业或者个人版数据库做中转。

( 注意：ORACLE通过访问SQL Server的数据库链接时，用select \* 的时候字段名是用双引号引起来的。)

**MySQL数据库导出和导入**

1）.MySQLimport的语法介绍：

　　MySQLimport位于MySQL/bin目录中，是MySQL的一个载入（或者说导入）数据的一个非常有效的工具。这是一个命令行工具。有两个参数以及大量的选项可供选择。这个工具把一个文本文件（text file）导入到你指定的数据库和表中。比方说我们要从文件Customers.txt中把数据导入到数据库Meet\_A\_Geek中的表Custermers中：

　　MySQLimport Meet\_A\_Geek Customers.txt

　　注意：这里Customers.txt是我们要导入数据的文本文件，而Meet\_A\_Geek是我们要操作的数据库，数据库中的表名是Customers，这里文本文件的数据格式必须与Customers表中的记录格式一致，否则MySQLimport命令将会出错。

　　其中表的名字是导入文件的第一个句号（.）前面文件字符串，另外一个例子：MySQLimport Meet\_A\_Geek Cus.to.mers.txt

　　那么我们将把文件中的内容导入到数据库Meet\_A\_Geek 中的Cus表中。

　　上面的例子中，都只用到两个参数，并没有用到更多的选项，下面介绍MySQLimport的选项

　　2）.MySQLimport的常用选项介绍:

　　选项 功能

　　-d or --delete 新数据导入数据表中之前删除数据数据表中的所有信息

　　-f or --force 不管是否遇到错误，MySQLimport将强制继续插入数据

　　-i or --ignore MySQLimport跳过或者忽略那些有相同唯一关键字的行， 导入文件中的数据将被忽略。

　　-l or -lock-tables 数据被插入之前锁住表，这样就防止了，你在更新数据库时，用户的查询和更新受到影响。

　　-r or -replace 这个选项与－i选项的作用相反；此选项将替代表中有相同唯一关键字的记录。

　　--fields-enclosed- by= char 指定文本文件中数据的记录时以什么括起的， 很多情况下数据以双引号括起。 默认的情况下数据是没有被字符括起的。

　　--fields-terminated- by=char 指定各个数据的值之间的分隔符，在句号分隔的文件中，分隔符是句号。您可以用此选项指定数据之间的分隔符。

　　默认的分隔符是跳格符（Tab）

　　--lines-terminated- by=str 此选项指定文本文件中行与行之间数据的分隔字符串或者字符。 默认的情况下MySQLimport以newline为行分隔符。

　　您可以选择用一个字符串来替代一个单个的字符：

　　一个新行或者一个回车。

　　MySQLimport命令常用的选项还有-v 显示版本（version），-p 提示输入密码（password）等。

　　3）.例子：导入一个以逗号为分隔符的文件

　　文件中行的记录格式是这样的：

　　"1", "ORD89876", "1 Dozen Roses", "19991226"

　　我们的任务是要把这个文件里面的数据导入到数据库Meet\_A\_Geek中的表格Orders中，我们使用这个命令:

　　bin/MySQLimport –prl –fields-enclosed-by=" –fields-terminated-by=, Meet\_A\_Geek Orders.txt

　　这个命令可能看起来很不爽，不过当你熟悉了之后，这是非常简单的。第一部分，bin/MySQLimport ,告诉操作系统你要运行的命令是MySQL/bin目录下的MySQLimport，选项p是要求输入密码，这样就要求你在改动数据库之前输入密码，操作起来会更安全。 我们用了r选项是因为我们想要把表中的唯一关键字与文件记录中有重复唯一关键字的记录替换成文件中的数据。我们表单中的数据不是最新的，需要用文件中的数据去更新，因而就用r这个选项，替代数据库中已经有的记录。l选项的作用是在我们插入数据的时候锁住表，这样就阻止了用户在我们更新表的时候对表进行查询或者更改的操作。

　　批处理是一种非交互式运行MySQL程序的方法，如同您在MySQL中使用的命令一样，你仍然将使用这些命令。

　　为了实现批处理，您重定向一个文件到MySQL程序中，首先我们需要一个文本文件，这个文本文件包含有与我们在MySQL中输入的命令相同的文本。

　　比如我们要插入一些数据，使用包含下面文本的文件(文件名为New\_Data.sql,当然我们也可以取名为New\_Data.txt及任何其他的合法名字，并不一定要以后缀sql结尾)：

　　USE Meet\_A\_Geek;

　　INSERT INTO Customers (Customer\_ID, Last\_Name) VALUES(NULL, "Block");

　　INSERT INTO Customers (Customer\_ID, Last\_Name) VALUES(NULL, "Newton");

　　INSERT INTO Customers (Customer\_ID, Last\_Name) VALUES(NULL, "Simmons");

　　注意上面的这些句子的语法都必须是正确的，并且每个句子以分号结束。

　　上面的USE命令选择数据库，INSERT命令插入数据。

　　下面我们要把上面的文件导入到数据库中，导入之前要确认数据库已经在运行，即是MySQLd进程（或者说服务，Windows NT下面称为”服务“，unix下面为”进程“）已经在运行。

　　然后运行下面的命令：

　　bin/MySQL –p < /home/mark/New\_Data.sql

　　接着按提示输入密码，如果上面的文件中的语句没有错误，那么这些数据就被导入到了数据库中。

　　命令行中使用LOAD DATA INFILE 从文件中导入数据到数据库:

　　现在您可能会问自己，"究竟为什么我要输入所有的这些SQL语句到文件中，然后通过程序运行它们呢？”

这样看起来好像需要大量的工作。很好，你这样想很可能就对了。但是假如你有从所有这些命令中产生的log记录呢？现在这样就很棒，嗯，大多数数据库都会自动产生数据库中的事件记录的log。而大部分log都包含有用过的原始的SQL命令。因此，如果您不能从您现在的数据库中导出数据到新的MySQL数据库中使用，那么您可以使用log和MySQL的批处理特性，来快速且方便地导入您地数据。当然，这样就省去了打字的麻烦。

　　LOAD DATA INFILE

　　这是我们要介绍的最后一个导入数据到MySQL数据库中的方法。这个命令与MySQLimport非常相似，但这个方法可以在MySQL命令行中使用。也就是说您可以在所有使用API的程序中使用这个命令。使用这种方法，您就可以在应用程序中导入您想要导入的数据。

　　使用这个命令之前，MySQLd进程（服务）必须已经在运行。

　　启动MySQL命令行：

　　bin/MySQL –p

　　按提示输入密码，成功进入MySQL命令行之后，输入下面的命令：

　　USE Meet\_A\_Geek;

　　LOAD DATA INFILE "/home/mark/data.sql" INTO TABLE Orders;

　　简单的讲，这样将会把文件data.sql中的内容导入到表Orders中，如MySQLimport工具一样，这个命令也有一些可以选择的参数。比如您需要把自己的电脑上的数据导入到远程的数据库服务器中，您可以使用下面的命令：

　　LOAD DATA LOCAL INFILE "C:MyDocsSQL.txt" INTO TABLE Orders;

　　上面的LOCAL参数表示文件是本地的文件，服务器是您所登陆的服务器。

　　这样就省去了使用ftp来上传文件到服务器，MySQL替你完成了.

　　您也可以设置插入语句的优先级，如果您要把它标记为低优先级（LOW\_PRIORITY），那么MySQL将会等到没有其他人读这个表的时候，才把插入数据。可以使用如下的命令：

　　 LOAD DATA LOW\_PRIORITY INFILE "/home/mark/data.sql" INTO TABLE Orders;

　　您也可以指定是否在插入数据的时候，取代或者忽略文件与数据表中重复的键值。替代重复的键值的语法：

　　LOAD DATA LOW\_PRIORITY INFILE "/home/mark/data.sql" REPLACE INTO TABLE Orders;

　　上面的句子看起来有点笨拙，但却把关键字放在了让您的剖析器可以理解的地方。

　　下面的一对选项描述了文件的记录格式，这些选项也是在MySQLimport工具中可以用的。他们在这里看起来有点不同。首先，要用到FIELDS关键字，如果用到这个关键字，MySQL剖析器希望看到至少有下面的一个选项：

　　TERMINATED BY character

　　 ENCLOSED BY character

　　 ESCAPED BY character

　　这些关键字与它们的参数跟MySQLimport中的用法是一样的. The

　　 TERMINATED BY 描述字段的分隔符，默认情况下是tab字符（t）

　　 ENCLOSED BY描述的是字段的括起字符。比方以引号括起每一个字段。

　　 ESCAPED BY 描述的转义字符。默认的是反些杠（backslash： ）.

　　下面仍然使用前面的MySQLimport命令的例子，用LOAD DATA INFILE语句把同样的文件导入到数据库中:

　　LOAD DATA INFILE "/home/mark/Orders.txt" REPLACE INTO TABLE Orders FIELDS TERMINATED BY ',' 　　ENCLOSED BY '"';

　　LOAD DATA INFILE语句中有一个MySQLimport工具中没有特点：

　　LOAD DATA INFILE 可以按指定的列把文件导入到数据库中。

　　当我们要把数据的一部分内容导入的时候，这个特点就很重要。比方说，我们要从Access数据库升级到MySQL数据库的时候，需要加入一些栏目（列/字段/field）到MySQL数据库中，以适应一些额外的需要。

　　这个时候，我们的Access数据库中的数据仍然是可用的，但是因为这些数据的栏目(field)与MySQL中的不再匹配，因此而无法再使用MySQLimport工具。尽管如此，我们仍然可以使用LOAD DATA INFILE，下面的例子显示了如何向指定的栏目(field)中导入数据：

　　LOAD DATA INFILE "/home/Order.txt" INTO TABLE Orders(Order\_Number, Order\_Date, Customer\_ID);

　　如您所见，我们可以指定需要的栏目（fields）。这些指定的字段依然是以括号括起，由逗号分隔的，如果您遗漏了其中任何一个，MySQL将会提醒您^\_^

　　Importing Data from Microsoft Access （从Access中导入数据，略）

　　您可以看到MySQL有很多可以导入数据的方法，然而这些只是数据传输中的一半。另外的一般是从MySQL数据库中导出数据。有许多的原因我们需要导出数据。一个重要的原因是用于备份数据库。数据的造价常常是昂贵的，需要谨慎处理它们。经常地备份可以帮助防止宝贵数据地丢失；另外一个原因是，也许您希望导出数据来共享。 在这个信息技术不断成长的世界中，共享数据变得越来越常见。

　　比方说Macmillan USA维护护着一个将要出版的书籍的大型数据库。这个数据库在许多书店之间共享，这样他们就知道哪些书将会很快出版。医院越来越走向采用无纸病历记录，这样这些病历可以随时跟着你。世界变得越来越小，信息也被共享得越来越多。有很多中导出数据得方法，它们都跟导入数据很相似。因为，毕竟，这些都只是一种透视得方式。从数据库导出的数据就是从另一端导入的数据。这里我们并不讨论其他的数据库各种各样的导出数据的方法，您将学会如何用MySQL来实现数据导出。

　　使用MySQLdump:

　　MySQLdump命令位于MySQL/bin/目录中

　　MySQLdump工具很多方面类似相反作用的工具MySQLimport。它们有一些同样的选项。但MySQLdump能够做更多的事情。它可以把整个数据库装载到一个单独的文本文件中。这个文件包含有所有重建您的数据库所需要的SQL命令。这个命令取得所有的模式（Schema，后面有解释）并且将其转换成DDL语法（CREATE语句，即数据库定义语句），取得所有的数据，并且从这些数据中创建INSERT语句。这个工具将您的数据库中所有的设计倒转。因为所有的东西都被包含到了一个文本文件中。这个文本文件可以用一个简单的批处理和一个合适SQL语句导回到MySQL中。这个工具令人难以置信地简单而快速。决不会有半点让人头疼地地方。

　　因此，如果您像装载整个数据库Meet\_A\_Geek的内容到一个文件中，可以使用下面的命令：

　　 bin/MySQLdump –p Meet\_A\_Geek > MeetAGeek\_Dump\_File.txt

　　这个语句也允许您指定一个表进行dump(备份/导出/装载?)。如果您只是希望把数据库Meet\_A\_Geek中的表Orders中的整个内容导出到一个文件，可以使用下面的命令：

　　 bin/MySQLdump –p Meet\_A\_Geek Orders >MeetAGeek\_Orders.txt

　　这个非常的灵活，您甚至可以使用WHERE从句来选择您需要的记录导出到文件中。要达到这样的目的，可以使用类似于下面的命令：

　　 bin/MySQLdump –p –where="Order\_ID > 2000" Meet\_A\_Geek Orders > Special\_Dump.txt

　　MySQLdump工具有大量的选项，部分选项如下表：

　　选项/Option 作用/Action Performed

　　--add-drop-table

　　 这个选项将会在每一个表的前面加上DROP TABLE IF EXISTS语句，这样可以保证导回MySQL数据库的时候不会出错，因为每次导回的时候，都会首先检查表是否存在，存在就删除

　　--add-locks

　　 这个选项会在INSERT语句中捆上一个LOCK TABLE和UNLOCK TABLE语句。这就防止在这些记录被再次导入数据库时其他用户对表进行的操作

　　-c or - complete\_insert

　　 这个选项使得MySQLdump命令给每一个产生INSERT语句加上列（field）的名字。当把数据导出导另外一个数据库时这个选项很有用。

　　--delayed-insert 在INSERT命令中加入DELAY选项

　　 -F or -flush-logs 使用这个选项，在执行导出之前将会刷新MySQL服务器的log.

　　 -f or -force 使用这个选项，即使有错误发生，仍然继续导出

　　 --full 这个选项把附加信息也加到CREATE TABLE的语句中

　　 -l or -lock-tables 使用这个选项，导出表的时候服务器将会给表加锁。

　　 -t or -no-create- info

　　 这个选项使的MySQLdump命令不创建CREATE TABLE语句，这个选项在您只需要数据而不需要DDL（数据库定义语句）时很方便。

　　-d or -no-data 这个选项使的MySQLdump命令不创建INSERT语句。

　　 在您只需要DDL语句时，可以使用这个选项。

　　 --opt 此选项将打开所有会提高文件导出速度和创造一个可以更快导入的文件的选项。

　　 -q or -quick 这个选项使得MySQL不会把整个导出的内容读入内存再执行导出，而是在读到的时候就写入导文件中。

　　 -T path or -tab = path 这个选项将会创建两个文件，一个文件包含DDL语句或者表创建语句，另一个文件包含数据。DDL文件被命名为table\_name.sql,数据文件被命名为table\_name.txt.路径名是存放这两个文件的目录。目录必须已经存在，并且命令的使用者有对文件的特权。

　　-w "WHERE Clause" or -where = "Where clause "

　　 如前面所讲的，您可以使用这一选项来过筛选将要放到导出文件的数据。

　　假定您需要为一个表单中要用到的帐号建立一个文件，经理要看今年（2004年）所有的订单（Orders），它们并不对DDL感兴趣，并且需要文件有逗号分隔，因为这样就很容易导入到Excel中。 为了完成这个人物，您可以使用下面的句子：

　　bin/MySQLdump –p –where "Order\_Date >='2000-01-01'"

　　 –tab = /home/mark –no-create-info –fields-terminated-by=, Meet\_A\_Geek Orders

　　 这将会得到您想要的结果。

　　schema：模式The set of statements, expressed in data definition language, that completely describe the structure of a data base.

　　 一组以数据定义语言来表达的语句集,该语句集完整地描述了数据库的结构。

　　SELECT INTO OUTFILE ：

　　如果您觉得MySQLdump工具不够酷，就使用SELECT INTO OUTFILE吧, MySQL同样提供一个跟LOAD DATA INFILE命令有相反作用的命令，这就是SELECT INTO OUTFILE 命令，这两个命令有很多的相似之处。首先，它们有所有的选项几乎相同。现在您需要完成前面用MySQLdump完成的功能，可以依照下面的步骤进行操作：

　　1. 确保MySQLd进程（服务）已经在运行

　　 2. cd /usr/local/MySQL

　　 3. bin/MySQLadmin ping ；// 如果这个句子通不过，可以用这个：MySQLadmin -u root -p ping

MySQLadmin ping用于检测MySQLd的状态，is alive说明正在运行，出错则可能需要用户名和密码。

　　 4. 启动MySQL 监听程序.

　　 5. bin/MySQL –p Meet\_A\_Geek；// 进入MySQL命令行，并且打开数据库Meet\_A\_Geek,需要输入密码

　　 6. 在命令行中，输入一下命令:

　　SELECT \* INTO OUTFILE '/home/mark/Orders.txt'

　　 FIELDS

　　 TERMINATED BY = ','

　　 FROM Orders

　　 WHERE Order\_Date >= '2000-01-01'

　　在你按了Return（回车）之后，文件就创建了。这个句子就像一个规则的SELECT语句，只是把想屏幕的输出重定向到了文件中。这意味这您可以使用JOIN来实现多表的高级查询。这个特点也可以被用作一个报表产生器。

　　比方说，您可以组合这一章中讨论的方法来产生一个非常有趣的查询，试试这个：

　　在MySQL目录建立一个名为Report\_G.rpt 的文本文件，加入下面的行：

　　USE Meet\_A\_Geek;

　　 INSERT INTO Customers (Customer\_ID, Last\_Name, First\_Name)

　　 VALUES (NULL, "Kinnard", "Vicky");

　　 INSERT INTO Customers (Customer\_ID, Last\_Name, First\_Name)

　　 VALUES (NULL, "Kinnard", "Steven");

　　 INSERT INTO Customers (Customer\_ID, Last\_Name, First\_Name)

　　 VALUES (NULL, "Brown", "Sam");

　　 SELECT Last\_Name INTO OUTFILE '/home/mark/Report.rpt'

　　 FROM Customers WHERE Customer\_ID > 1;

　　然后确认 MySQL进程在运行，并且您在MySQL目录中， 输入下面的命令：

　　bin/MySQL < Report\_G.rpt检查您命名作为输出的文件，这个文件将会包含所有您在Customers表中输入的顾客的姓。 如您所见，您可以使用今天学到的导入/导出（import/export）的方法来帮助得到报表。 [查看全文](http://kendy517.itpub.net/post/28302/262254)

**关于MSSQL数据库的字段**

数据类型是数据的一种属性，表示数据所表示信息的类型。任何一种计算机语言都定义了自己的数据类型。当然，不同的程序语言都具有不同的特点，所定义的数据类型的各类和名称都或多或少有些不同。SQLServer提供了25种数据类型：

　　·Binary[(n)]

　　·Varbinary[(n)]

　　·Char[(n)]

　　·Varchar[(n)]

　　·Nchar[(n)]

　　·Nvarchar[(n)]

　　·Datetime

　　·Smalldatetime

　　·Decimal[(p[,s])]

　　·Numeric[(p[,s])]

　　·Float[(n)]

　　·Real

　　·Int

　　·Smallint

　　·Tinyint

　　·Money

　　·Smallmoney

　　·Bit

　　·Cursor

　　·Sysname

　　·Timestamp

　　·Uniqueidentifier

　　·Text

　　·Image

　　·Ntext

(1)二进制数据类型

　　二进制数据包括Binary、Varbinary和Image

　　Binary数据类型既可以是固定长度的(Binary),也可以是变长度的。

　　Binary[(n)]是n位固定的二进制数据。其中，n的取值范围是从1到8000。其存储窨的大小是n+4个字节。

　　Varbinary[(n)]是n位变长度的二进制数据。其中，n的取值范围是从1到8000。其存储窨的大小是n+4个字节，不是n个字节。

　　在Image数据类型中存储的数据是以位字符串存储的，不是由SQLServer解释的，必须由应用程序来解释。例如，应用程序可以使用BMP、TIEF、GIF和JPEG格式把数据存储在Image数据类型中。

(2)字符数据类型

　　字符数据的类型包括Char，Varchar和Text

　　字符数据是由任何字母、符号和数字任意组合而成的数据。

　　Varchar是变长字符数据，其长度不超过8KB。Char是定长字符数据，其长度最多为8KB。超过8KB的ASCII数据可以使用Text数据类型存储。例如，因为Html文档全部都是ASCII字符，并且在一般情况下长度超过8KB，所以这些文档可以Text数据类型存储在SQLServer中。

(3)Unicode数据类型

　　Unicode数据类型包括Nchar,Nvarchar和Ntext

　　在MicrosoftSQLServer中，传统的非Unicode数据类型允许使用由特定字符集定义的字符。在SQLServer安装过程中，允许选择一种字符集。使用Unicode数据类型，列中可以存储任何由Unicode标准定义的字符。在Unicode标准中，包括了以各种字符集定义的全部字符。使用Unicode数据类型，所战胜的窨是使用非Unicode数据类型所占用的窨大小的两倍。

　　在SQLServer中，Unicode数据以Nchar、Nvarchar和Ntext数据类型存储。使用这种字符类型存储的列可以存储多个字符集中的字符。当列的长度变化时，应该使用Nvarchar字符类型，这时最多可以存储4000个字符。当列的长度固定不变时，应该使用Nchar字符类型，同样，这时最多可以存储4000个字符。当使用Ntext数据类型时，该列可以存储多于4000个字符。

(4)日期和时间数据类型

　　日期和时间数据类型包括Datetime和Smalldatetime两种类型

　　日期和时间数据类型由有效的日期和时间组成。例如，有效的日期和时间数据包括“4/01/9812:15:00:00:00PM”和“1:28:29:15:01AM8/17/98”。前一个数据类型是日期在前，时间在后一个数据类型是霎时间在前，日期在后。在MicrosoftSQLServer中，日期和时间数据类型包括Datetime和Smalldatetime两种类型时，所存储的日期范围是从1753年1月1日开始，到9999年12月31日结束(每一个值要求8个存储字节)。使用Smalldatetime数据类型时，所存储的日期范围是1900年1月1日开始，到2079年12月31日结束(每一个值要求4个存储字节)。

　　日期的格式可以设定。设置日期格式的命令如下：

[SetDateFormat{format|@format\_var](mailto:SetDateFormat%7Bformat%7C@format_var)|

　　其中，[format|@format\_var](mailto:format%7C@format_var)是日期的顺序。有效的参数包括MDY、DMY、YMD、YDM、MYD和DYM。在默认情况下，日期格式为MDY。

　　例如，当执行SetDateFormatYMD之后，日期的格式为年月日形式；当执行SetDateFormatDMY之后，日期的格式为日月有年形式

（5）数字数据类型

　　数字数据只包含数字。数字数据类型包括正数和负数、小数（浮点数）和整数

　　整数由正整数和负整数组成，例如39、25、0-2和33967。在MicrsoftSQLServer中，整数存储的数据类型是　　Int，Smallint和Tinyint。Int数据类型存储数据的范围大于Smallint数据类型存储数据的范围，而Smallint据类型存储数据的范围大于Tinyint数据类型存储数据的范围。使用Int数据狗昔存储数据的范围是从-2147483648到2147483647（每一个值要求4个字节存储空间）。使用Smallint数据类型时，存储数据的范围从-32768到32767（每一个值要求2个字节存储空间）。使用Tinyint数据类型时，存储数据的范围是从0到255（每一个值要求1个字节存储空间）。

　　精确小娄数据在SQLServer中的数据类型是Decimal和Numeric。这种数据所占的存储空间根据该数据的位数后的位数来确定。

　　在SQLServer中，近似小数数据的数据类型是Float和Real。例如，三分之一这个分数记作。3333333，当使用近似数据类型时能准确表示。因此，从系统中检索到的数据可能与存储在该列中数据不完全一样。

（6）货币数据表示正的或者负的货币数量。

　　在MicrosoftSQLServer中，货币数据的数据类型是Money和Smallmoney

　　Money数据类型要求8个存储字节，Smallmoney数据类型要求4个存储字节。

（7）特殊数据类型

　　特殊数据类型包括前面没有提过的数据类型。特殊的数据类型有3种，即　　Timestamp、Bit和Uniqueidentifier。

　　Timestamp用于表示SQLServer活动的先后顺序，以二进投影的格式表示。Timestamp数据与插入数据或者日期和时间没有关系。

　　Bit由1或者0组成。当表示真或者假、ON或者OFF时，使用Bit数据类型。例如，询问是否是每一次访问的客户机请求可以存储在这种数据类型的列中。

　　Uniqueidentifier由16字节的十六进制数字组成，表示一个全局唯一的。当表的记录行要求唯一时，GUID是非常有用。例如，在客户标识号列使用这种数据类型可以区别不同的客户。

2.用户定义的数据类型

　　用户定义的数据类型基于在MicrosoftSQLServer中提供的数据类型。当几个表中必须存储同一种数据类型时，并且为保证这些列有相同的数据类型、长度和可空性时，可以使用用户定义的数据类型。例如，可定义一种称为　　postal\_code的数据类型，它基于Char数据类型。

　　当创建用户定义的数据类型时，必须提供三个数：数据类型的名称、所基于的系统数据类型和数据类型的可空性。

（1）创建用户定义的数据类型

　　创建用户定义的数据类型可以使用Transact-SQL语句。系统存储过程sp\_addtype可以来创建用户定义的数据类型。其语法形式如下：

　　sp\_addtype{type},[,system\_data\_bype][,'null\_type']

　　其中，type是用户定义的数据类型的名称。system\_data\_type是系统提供的数据类型，例如Decimal、Int、Char　　等等。null\_type表示该数据类型是如何处理空值的，必须使用单引号引起来，例如'NULL'、'NOTNULL'或者'NONULL'。

　　例子：

　　Usecust

　　Execsp\_addtypessn,'Varchar(11)',"NotNull'

　　创建一个用户定义的数据类型ssn，其基于的系统数据类型是变长为11的字符，不允许空。

　　例子：

　　Usecust

　　Execsp\_addtypebirthday,datetime,'Null'

　　创建一个用户定义的数据类型birthday，其基于的系统数据类型是DateTime，允许空。

　　例子：

　　Usemaster

　　Execsp\_addtypetelephone,'varchar(24),'NotNull'

　　Eexcsp\_addtypefax,'varchar(24)','Null'

　　创建两个数据类型，即telephone和fax

（2）删除用户定义的数据类型

　　当用户定义的数据类型不需要时，可删除。删除用户定义的数据类型的命令是sp\_droptype{'type'}。

　　例子：

　　Usemaster

　　Execsp\_droptype'ssn'

　　注意：当表中的列还正在使用用户定义的数据类型时，或者在其上面还绑定有默认或者规则时，这种用户定义的数据类型不能删除。

SQLSERVER的字段类型说明

以下为SQL SERVER7.0以上版本的字段类型说明。SQL SERVER6.5的字段类型说明请参考SQL SERVER提供的说明。

|  |  |
| --- | --- |
| 字段类型 | 描述 |
| bit | 0或1的整型数字 |
| int | 从-2^31(-2,147,483,648)到2^31(2,147,483,647)的整型数字 |
| smallint | 从-2^15(-32,768)到2^15(32,767)的整型数字 |
| tinyint | 从0到255的整型数字 |
|  |  |
| decimal | 从-10^38到10^38-1的定精度与有效位数的数字 |
| numeric | decimal的同义词 |
|  |  |
| money | 从-2^63(-922,337,203,685,477.5808)到2^63-1(922,337,203,685,477.5807)的货币数据，最小货币单位千分之十 |
| smallmoney | 从-214,748.3648到214,748.3647的货币数据，最小货币单位千分之十 |
|  |  |
| float | 从-1.79E+308到1.79E+308可变精度的数字 |
| real | 从-3.04E+38到3.04E+38可变精度的数字 |
|  |  |
| datetime | 从1753年1月1日到9999年12日31的日期和时间数据，最小时间单位为百分之三秒或3.33毫秒 |
| smalldatetime | 从1900年1月1日到2079年6月6日的日期和时间数据，最小时间单位为分钟 |
|  |  |
| timestamp | 时间戳，一个数据库宽度的唯一数字 |
| uniqueidentifier | 全球唯一标识符GUID |
|  |  |
| char | 定长非Unicode的字符型数据，最大长度为8000 |
| varchar | 变长非Unicode的字符型数据，最大长度为8000 |
| text | 变长非Unicode的字符型数据，最大长度为2^31-1(2G) |
|  |  |
| nchar | 定长Unicode的字符型数据，最大长度为8000 |
| nvarchar | 变长Unicode的字符型数据，最大长度为8000 |
| ntext | 变长Unicode的字符型数据，最大长度为2^31-1(2G) |
|  |  |
| binary | 定长二进制数据，最大长度为8000 |
| varbinary | 变长二进制数据，最大长度为8000 |
| image | 变长二进制数据，最大长度为2^31-1(2G) |

[down=upload/images/618fmall2.rar]DOWN LOAD THE FILE[/down]

天气:大雨,ccdot发表于2006-6-20 14:44:36,阅读了655次,共有个2回复.

--语 句　　　　　　　　　　　　　　功 能

--数据操作

SELECT　　　--从数据库表中检索数据行和列

INSERT　　　--向数据库表添加新数据行

DELETE　　　--从数据库表中删除数据行

UPDATE　　　--更新数据库表中的数据

--数据定义

CREATE TABLE　　--创建一个数据库表

DROP TABLE　　 --从数据库中删除表

ALTER TABLE　　 --修改数据库表结构

CREATE VIEW　　 --创建一个视图

DROP VIEW　　 --从数据库中删除视图

CREATE INDEX　　--为数据库表创建一个索引

DROP INDEX　　 --从数据库中删除索引

CREATE PROCEDURE　 --创建一个存储过程

DROP PROCEDURE　　--从数据库中删除存储过程

CREATE TRIGGER　　--创建一个触发器

DROP TRIGGER　　--从数据库中删除触发器

CREATE SCHEMA　　--向数据库添加一个新模式

DROP SCHEMA　　 --从数据库中删除一个模式

CREATE DOMAIN　　--创建一个数据值域

ALTER DOMAIN　　--改变域定义

DROP DOMAIN　　 --从数据库中删除一个域

--数据控制

GRANT　　　--授予用户访问权限

DENY　　　--拒绝用户访问

REVOKE　　　--解除用户访问权限

--事务控制

COMMIT　　　--结束当前事务

ROLLBACK　　 --中止当前事务

SET TRANSACTION　　--定义当前事务数据访问特征

--程序化SQL

DECLARE　　　--为查询设定游标

EXPLAN　　　--为查询描述数据访问计划

OPEN　　　--检索查询结果打开一个游标

FETCH　　　--检索一行查询结果

CLOSE　　　--关闭游标

PREPARE　　　--为动态执行准备SQL 语句

EXECUTE　　　--动态地执行SQL 语句

DESCRIBE　　 --描述准备好的查询

---局部变量

declare @id char(10)

--set @id = ’10010001’

select @id = ’10010001’

---全局变量

---必须以@@开头

**MSSQL常用汇总**

下列语句部分是Mssql语句，不可以在access中使用。

SQL分类：

DDL—数据定义语言(CREATE，ALTER，DROP，DECLARE)

DML—数据操纵语言(SELECT，DELETE，UPDATE，INSERT)

DCL—数据控制语言(GRANT，REVOKE，COMMIT，ROLLBACK)

首先,简要介绍基础语句：

1、说明：创建数据库

CREATE DATABASE database-name

2、说明：删除数据库

drop database dbname

3、说明：备份sql server

--- 创建 备份数据的 device

USE master

EXEC sp\_addumpdevice 'disk', 'testBack', 'c:mssql7backupMyNwind\_1.dat'

--- 开始 备份

BACKUP DATABASE pubs TO testBack

4、说明：创建新表

create table tabname(col1 type1 [not null] [primary key],col2 type2 [not null],..)

根据已有的表创建新表：

A：create table tab\_new like tab\_old (使用旧表创建新表)

B：create table tab\_new as select col1,col2… from tab\_old definition only

5、说明：删除新表drop table tabname

6、说明：增加一个列

Alter table tabname add column col type

注：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。

7、说明：添加主键： Alter table tabname add primary key(col)

说明：删除主键： Alter table tabname drop primary key(col)

8、说明：创建索引：create [unique] index idxname on tabname(col….)

删除索引：drop index idxname

注：索引是不可更改的，想更改必须删除重新建。

9、说明：创建视图：create view viewname as select statement

删除视图：drop view viewname

10、说明：几个简单的基本的sql语句

选择：select \* from table1 where 范围

插入：insert into table1(field1,field2) values(value1,value2)

删除：delete from table1 where 范围

更新：update table1 set field1=value1 where 范围

查找：select \* from table1 where field1 like ’%value1%’ ---like的语法很精妙，查资料!

排序：select \* from table1 order by field1,field2 [desc]

总数：select count(\*) as totalcount from table1

求和：select sum(field1) as sumvalue from table1

平均：select avg(field1) as avgvalue from table1

最大：select max(field1) as maxvalue from table1

最小：select min(field1) as minvalue from table1

11、说明：几个高级查询运算词

A： UNION 运算符

UNION 运算符通过组合其他两个结果表（例如 TABLE1 和 TABLE2）并消去表中任何重复行而派生出一个结果表。当 ALL 随 UNION 一起使用时（即 UNION ALL），不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。

B： EXCEPT 运算符

EXCEPT 运算符通过包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。当 ALL 随 EXCEPT 一起使用时 (EXCEPT ALL)，不消除重复行。

C： INTERSECT 运算符

INTERSECT 运算符通过只包括 TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。当 ALL 随 INTERSECT 一起使用时 (INTERSECT ALL)，不消除重复行。

注：使用运算词的几个查询结果行必须是一致的。

12、说明：使用外连接

A、left outer join：

左外连接（左连接）：结果集几包括连接表的匹配行，也包括左连接表的所有行。

SQL: select a.a, a.b, a.c, b.c, b.d, b.f from a LEFT OUT JOIN b ON a.a = b.c

B：right outer join:

右外连接(右连接)：结果集既包括连接表的匹配连接行，也包括右连接表的所有行。

C：full outer join：

全外连接：不仅包括符号连接表的匹配行，还包括两个连接表中的所有记录。

其次，大家来看一些不错的sql语句

1、说明：复制表(只复制结构,源表名：a 新表名：b) (Access可用)

法一：select \* into b from a where 1<>1

法二：select top 0 \* into b from a

2、说明：拷贝表(拷贝数据,源表名：a 目标表名：b) (Access可用)

insert into b(a, b, c) select d,e,f from b;

3、说明：跨数据库之间表的拷贝(具体数据使用绝对路径) (Access可用)

insert into b(a, b, c) select d,e,f from b in ‘具体数据库’ where 条件

例子：..from b in '"&Server.MapPath(".")&"data.mdb" &"' where..

4、说明：子查询(表名1：a 表名2：b)

select a,b,c from a where a IN (select d from b ) 或者: select a,b,c from a where a IN (1,2,3)

5、说明：显示文章、提交人和最后回复时间

select a.title,a.username,b.adddate from table a,(select max(adddate) adddate from table where table.title=a.title) b

6、说明：外连接查询(表名1：a 表名2：b)

select a.a, a.b, a.c, b.c, b.d, b.f from a LEFT OUT JOIN b ON a.a = b.c

7、说明：在线视图查询(表名1：a )

select \* from (SELECT a,b,c FROM a) T where t.a > 1;

8、说明：between的用法,between限制查询数据范围时包括了边界值,not between不包括

select \* from table1 where time between time1 and time2

select a,b,c, from table1 where a not between 数值1 and 数值2

9、说明：in 的使用方法

select \* from table1 where a [not] in (‘值1’,’值2’,’值4’,’值6’)

10、说明：两张关联表，删除主表中已经在副表中没有的信息

delete from table1 where not exists ( select \* from table2 where table1.field1=table2.field1 )

11、说明：四表联查问题：

select \* from a left inner join b on a.a=b.b right inner join c on a.a=c.c inner join d on a.a=d.d where .....

12、说明：日程安排提前五分钟提醒

SQL: select \* from 日程安排 where datediff('minute',f开始时间,getdate())>5

13、说明：一条sql 语句搞定数据库分页

select top 10 b.\* from (select top 20 主键字段,排序字段 from 表名 order by 排序字段 desc) a,表名 b where b.主键字段 = a.主键字段 order by a.排序字段

14、说明：前10条记录

select top 10 \* form table1 where 范围

15、说明：选择在每一组b值相同的数据中对应的a最大的记录的所有信息(类似这样的用法可以用于论坛每月排行榜,每月热销产品分析,按科目成绩排名,等等.)

select a,b,c from tablename ta where a=(select max(a) from tablename tb where tb.b=ta.b)

16、说明：包括所有在 TableA 中但不在 TableB和TableC 中的行并消除所有重复行而派生出一个结果表

(select a from tableA ) except (select a from tableB) except (select a from tableC)

17、说明：随机取出10条数据

select top 10 \* from tablename order by newid()

18、说明：随机选择记录

select newid()

19、说明：删除重复记录

Delete from tablename where id not in (select max(id) from tablename group by col1,col2,...)

20、说明：列出数据库里所有的表名

select name from sysobjects where type='U'

21、说明：列出表里的所有的

select name from syscolumns where id=object\_id('TableName')

22、说明：列示type、vender、pcs字段，以type字段排列，case可以方便地实现多重选择，类似select 中的case。

select type,sum(case vender