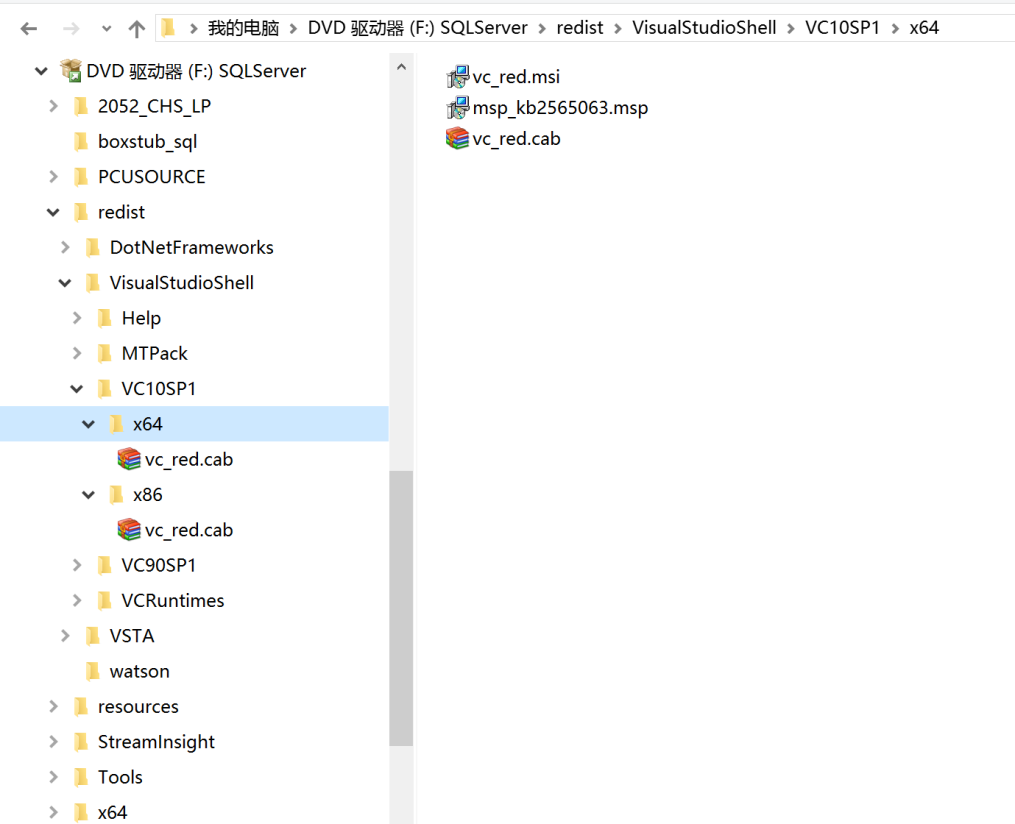
# 安装

## sql server 2008 r2 与sql server 2008是一样的吗

1、不一样。  
2、官方提供的MSDN安装版中无论是SQL 2008还是SQL 2008 R2都是三版本合一，三版本为32位/64位/IA64（安腾多CPU专用版）。  
3、SQL 2008 R2是SQL 2008的后继版本，一般认为它是SQL2008的改进版。  
4、最新的微软SQL Server为SQL Server 2012版。

## SQL 2012安装问题

**1、VC 2010**



2、.NetFramework3.5.1

安装方法：

先把下载的名为NetFx3.cab的离线安装包放到Win10系统盘C:\Windows文件夹里。

然后以管理员身份运行命令提示符，输入并回车运行以下命令：

dism /online /Enable-Feature /FeatureName:NetFx3 /Source:"%windir%" /LimitAccess

等待部署进度100%即可。

## 解决安装时的挂起错误

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager

中找到PendingFileRenameOperations，删除即可。

## WIN7 开通1433端口

Telnet 192.168.2.1 1433

控制面板--系统和安全---windows防火墙---高级设置--入站规则--新建规则

选择：端口-特定1433-允许连接-全部-名称

## sql2000安全

将有安全问题的SQL过程删除.比较全面.一切为了安全!

删除了调用shell，注册表，COM组件的破坏权限

use master

EXEC sp\_dropextendedproc 'xp\_cmdshell'

EXEC sp\_dropextendedproc 'Sp\_OACreate'

EXEC sp\_dropextendedproc 'Sp\_OADestroy'

EXEC sp\_dropextendedproc 'Sp\_OAGetErrorInfo'

EXEC sp\_dropextendedproc 'Sp\_OAGetProperty'

EXEC sp\_dropextendedproc 'Sp\_OAMethod'

EXEC sp\_dropextendedproc 'Sp\_OASetProperty'

EXEC sp\_dropextendedproc 'Sp\_OAStop'

EXEC sp\_dropextendedproc 'Xp\_regaddmultistring'

EXEC sp\_dropextendedproc 'Xp\_regdeletekey'

EXEC sp\_dropextendedproc 'Xp\_regdeletevalue'

EXEC sp\_dropextendedproc 'Xp\_regenumvalues'

EXEC sp\_dropextendedproc 'Xp\_regread'

EXEC sp\_dropextendedproc 'Xp\_regremovemultistring'

EXEC sp\_dropextendedproc 'Xp\_regwrite'

drop procedure sp\_makewebtask

全部复制到"SQL查询分析器"

点击菜单上的--"查询"--"执行"，就会将有安全问题的SQL过程删除(以上是7i24的正版用户的技术支持)

更改默认SA空密码.数据库链接不要使用SA帐户.单数据库单独设使用帐户.只给public和db\_owner权限.

数据库不要放在默认的位置.

SQL不要安装在PROGRAM FILE目录下面.

# 系统管理

## 如何清除SQL Server Management Studio的最近服务器列表

**对于 SQL Server 2005 Management Studio，可以删除以下文件清空该列表：**

Win7: C:\Users\<user>\AppData\Roaming\Microsoft\Microsoft SQL Server\90\Tools\Shell\mru.dat

**对于 SQL Server 2008 Management Studio，可以删除以下文件清空该列表：**

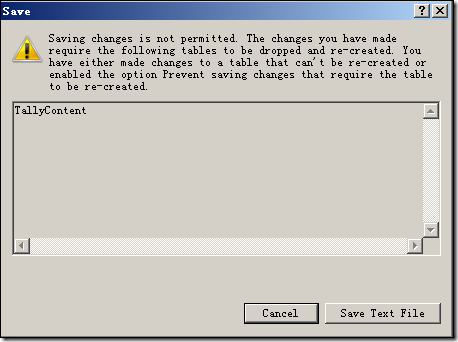
Win7: C:\Users\<user>\AppData\Roaming\Microsoft\Microsoft SQL Server\100\Tools\Shell\SqlStudio.bin

## SQL 2008 无法修改表结构

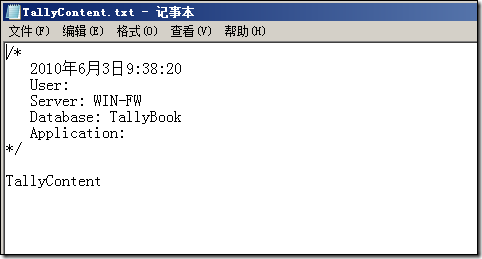
最近使用SqlServer2008，发现在修改完表字段名或是类型后点击保存时会弹出一个对话框，对话框内容大致如下

Saving changes is not permitted. The changes you have made require the following tables to be dropped and re-created. You have either made changes to a table that can't be re-created or enabled the option Prevent saving changes that require the table to be re-created

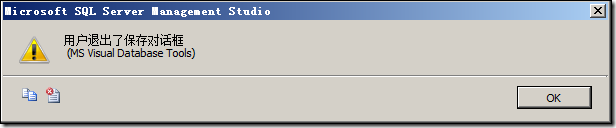
如下图：

[](http://images.cnblogs.com/cnblogs_com/oec2003/WindowsLiveWriter/SqlServer2008savechangesisnotpermitted_892E/2010-06-03_094326_2.png)

如果点击 Save Text File ，会保存一个文本文件，感觉没什么作用，内容如下图：

[](http://images.cnblogs.com/cnblogs_com/oec2003/WindowsLiveWriter/SqlServer2008savechangesisnotpermitted_892E/2010-06-03_095158_2.png)

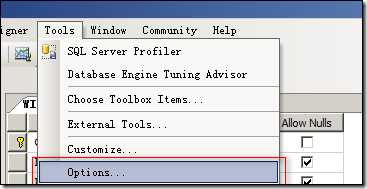
点击 Cancel 后会弹出另一个对话框，如下图：

[](http://images.cnblogs.com/cnblogs_com/oec2003/WindowsLiveWriter/SqlServer2008savechangesisnotpermitted_892E/2010-06-03_095243_2.png)

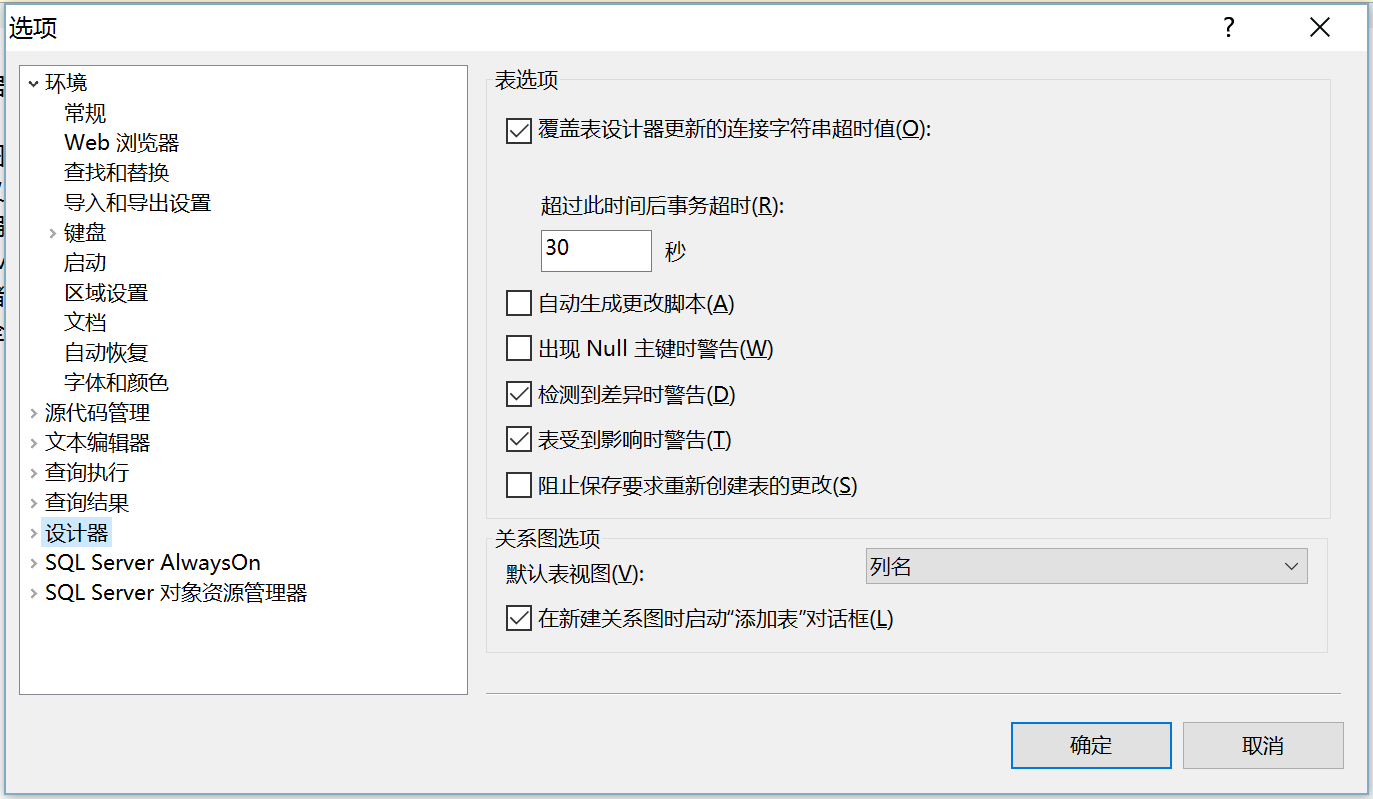
点击OK就关闭了对话框，当然我们的修改肯定也没有保存上。

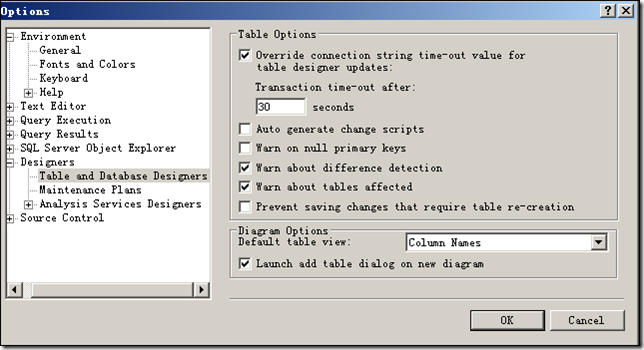
解决方法

打开工具-选项，如下图：

[](http://images.cnblogs.com/cnblogs_com/oec2003/WindowsLiveWriter/SqlServer2008savechangesisnotpermitted_892E/2010-06-03_100942_2.png)

在选项对话框中选择：Designers—Table and DataBase Designers ，将右边的Prevent saving changes that require table re-creation 前的勾选去掉，如下图：



[](http://images.cnblogs.com/cnblogs_com/oec2003/WindowsLiveWriter/SqlServer2008savechangesisnotpermitted_892E/2010-06-03_101349_2.png)

点击OK后，表的结构就可以随意修改保存了

## 查看SQL Server版本号

SELECT SERVERPROPERTY('Productversion') AS '产品版本',

SERVERPROPERTY('ProductLevel') AS '产品级别',

SERVERPROPERTY('edition') AS '版本';

--详细信息查询

EXEC xp\_msver

**SQL Server的版本号:**

* 8.0 -> sql server 2000
* 9.0 -> sql server 2005
* 10.0 -> sql server 2008
* 10.5 -> sql server 2008 R2
* 11.0 -> sql server 2012
* 12.0 -> sql server 2014
* 13.0 -> sql server 2016

## 配置sql server 2008 R2使它能向下兼容sql server 2008 ?

打开sql2008 R2,在数据库上点右键->属性->选项-兼容级别,降低级别后再备份还原到2008



## 命令行osql和sqlcmd

SQL Server 2000用osql

SQL Server 2008用osql，sqlcmd两个都可以

--信任联接，无需用户名、密码

osql -E -S

--执行文本文件中的语句

osql –S (Local) –U sa -P -i D:\Setup\zBillWork\Data\CreateUser.SQL

--附加数据库

osql –S (Local) –U sa -P -Q "exec sp\_attach\_db @dbname='zTimer',

@filename1='D:\SqlDrv \zTimer\_Data.MDF'

## 通过脚本创建数据库

USE master;

GO

CREATE DATABASE Test

ON

(

NAME = Test\_dat,

FILENAME = 'D:\Database\Testdat.mdf',

SIZE = 10,

MAXSIZE = UNLIMITED,

FILEGROWTH = 5 )

LOG ON

(

NAME = Test\_log,

FILENAME = 'D:\Database\Testlog.ldf',

SIZE = 5MB,

MAXSIZE = UNLIMITED,

FILEGROWTH = 5MB )

COLLATE Latin1\_General\_CI\_AS --指定数据库排序规则

GO

--设置数据库日志为简单模式减少日志（根据实际情况决定是否执行）

ALTER DATABASE Test SET RECOVERY SIMPLE;

GO

## 通过脚本附加数据库

1、附加数据库

EXEC sp\_attach\_db @dbname = N'DATA',

@filename1 = N'C:\XXDATA\DATA\_data.mdf',

@filename2 = N'C:\XXDATA\DATA\_log.ldf';

## 创建表和索引

USE Test;

GO

CREATE TABLE dbo.Dev

(

DevId INT IDENTITY(1, 1) NOT NULL,

DevName NVARCHAR(200) NOT NULL,

StockQty DECIMAL(9, 2) NULL,

Flag BIT NOT NULL DEFAULT( 1 ),

DevType SMALLINT NOT NULL DEFAULT( 1 ),

CreateUser NVARCHAR(10) NOT NULL,

CreateDate DATETIME NOT NULL DEFAULT( GETDATE()),

ModifyUser NVARCHAR(10) NULL,

ModifyDate DATETIME NULL,

CONSTRAINT PK\_Dev\_DevId PRIMARY KEY CLUSTERED( DevId ASC )

);

GO

USE Test;

GO

CREATE TABLE dbo.Dev

(

DevId INT IDENTITY(1, 1) NOT NULL,

DevName NVARCHAR(200) NOT NULL,

StockQty DECIMAL(9, 2) NULL,

Flag BIT NOT NULL DEFAULT( 1 ),

DevType SMALLINT NOT NULL DEFAULT( 1 ),

CreateUser NVARCHAR(10) NOT NULL,

CreateDate DATETIME NOT NULL DEFAULT( GETDATE()),

ModifyUser NVARCHAR(10) NULL,

ModifyDate DATETIME NULL,

);

GO

--创建主键

ALTER TABLE dbo.Dev

ADD CONSTRAINT PK\_Dev\_DevID PRIMARY KEY CLUSTERED( DevId );

GO

--删除主键

ALTER TABLE dbo.Dev

DROP CONSTRAINT PK\_Dev\_DevID;

## 修改表，添加索引和约束

USE Test;

GO

--添加字段

ALTER TABLE dbo.Dev

ADD remark NVARCHAR(500) NULL;

--修改字段名

EXECUTE sp\_rename 'dbo.Dev.remark','dbo.Dev.memo';

--修改字段类型

ALTER TABLE dbo.Dev

ALTER COLUMN memo NVARCHAR(300);

--添加有默认值的字段

ALTER TABLE dbo.Dev

ADD delflag INT NOT NULL DEFAULT 0;

--删除字段

ALTER TABLE dbo.Dev

DROP COLUMN delflag;

--删除约束

ALTER TABLE dbo.Dev

DROP CONSTRAINT [DF\_\_Dev\_\_delflag\_\_276EDEB3]

--查询约束

EXECUTE sp\_helpconstraint @objname='dbo.Dev'

--查询约束

SELECT \* FROM sys.objects WHERE parent\_object\_id=OBJECT\_ID('dbo.Dev') --所有

SELECT \* FROM sys.default\_constraints WHERE parent\_object\_id=OBJECT\_ID('dbo.Dev') --默认

SELECT \* FROM sys.check\_constraints WHERE parent\_object\_id=OBJECT\_ID('dbo.Dev') --检查

SELECT \* FROM sys.key\_constraints WHERE parent\_object\_id=OBJECT\_ID('dbo.Dev') --键

--添加默认约束

ALTER TABLE dbo.Dev

ADD CONSTRAINT DF\_Dev\_CreateUser DEFAULT( 'Admin' )FOR CreateUser;

--添加检查约束

ALTER TABLE dbo.Dev

ADD CONSTRAINT CK\_Dev\_StockQtyCheck CHECK( StockQty BETWEEN 0 AND 10000 );

--添加唯一约束

ALTER TABLE dbo.Dev

ADD CONSTRAINT UQ\_Dev\_DevName UNIQUE(DevName);

USE Test;

GO

--查询索引

SELECT i.object\_id, i.name index\_name, i.index\_id, i.type\_desc, i.is\_unique, ic.column\_id, c.name column\_name,ic.is\_descending\_key

FROM sys.indexes i

INNER JOIN sys.index\_columns ic ON ic.object\_id = i.object\_id AND ic.index\_id = i.index\_id

INNER JOIN sys.columns c ON c.object\_id = i.object\_id AND c.column\_id = ic.column\_id

WHERE i.object\_id = OBJECT\_ID('dbo.Dev');

--创建非聚集索引

CREATE NONCLUSTERED INDEX Index\_Dev\_DevName

ON dbo.Dev (DevName)

WITH(DROP\_EXISTING=ON)

--删除索引

DROP INDEX Index\_Dev\_DevName ON dbo.Dev;

--创建唯一非聚集索引

CREATE UNIQUE INDEX Index\_Dev\_DevName

ON dbo.Dev (DevName)

WITH(DROP\_EXISTING=ON)

## 排序规则查看

--查看服务器排序规则(安装时指定的排序规则)

SELECT SERVERPROPERTY('COLLATION') AS ServerCollation,

DATABASEPROPERTYEX('tempdb', 'COLLATION') AS TempdbCollation,

DATABASEPROPERTYEX(DB\_NAME(), 'COLLATION') AS CurrentDBCollation;

--查看数据库排序规则

USE master;

GO

SELECT name, collation\_name

FROM sys.databases;

--当前数据库是否大小写敏感

SELECT CASE WHEN N'A'=N'a' THEN N'不敏感' ELSE N'敏感' END

## 修改服务器的排序规则

1） 获取服务器排序规则

SELECT SERVERPROPERTY('COLLATION');

2） 查看当前服务器数据库安装版本

SELECT @@VERSION

3） 确认当前数据库默认安装版本及默认安装文件目录,请根据自己版本记录数据库默认安装目录。

(SQLServer Service) 默认安装目录：

SQL Server 2008 C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Release

SQL Server 2012 C:\Program Files\Microsoft SQL Server\110\Setup Bootstrap\SQLServer2012

SQL Server 2014 C:\Program Files\Microsoft SQL Server\120\Setup Bootstrap\SQLServer2014

SQL Server 2016 C:\Program Files\Microsoft SQL Server\140\Setup Bootstrap\SQLServer2016

4） 使用管理员权限打开命令行窗口，并进入默认安装目录

5） 停止SQLServer服务

net stop mssqlserver

6） 运行语句修改数据库排序规则

Setup /QUIET /ACTION=REBUILDDATABASE /instancename=MSSQLSERVER /SQLSYSADMINACCOUNTS=操作系统管理员账号 /sapwd=密码 /sqlcollation=Latin1\_General\_CI\_AS

格式说明:

Setup /QUIET

/ACTION=REBUILDDATABASE

/INSTANCENAME=InstanceName

/SQLSYSADMINACCOUNTS=accounts

/ [ SAPWD= StrongPassword ]

/SQLCOLLATION=CollationName

注意: accounts 用户使用电脑管理员用户，CollationName 为需要变更的排序格式

7） 启动SQLServer服务

net start mssqlserver

## 修改数据库排序规则

USE [master]

GO

ALTER DATABASE DBName SET SINGLE\_USER WITH ROLLBACK IMMEDIATE

GO

ALTER DATABASE DBName COLLATE Latin1\_General\_CI\_AS

GO

ALTER DATABASE DBName SET multi\_user WITH ROLLBACK IMMEDIATE

GO

## 批量修改表排序规则的脚本（不完善有的改不过来，待验证）

USE [master];

GO

ALTER DATABASE DBName SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

GO

DECLARE @table NVARCHAR(128); --循环Item表名

DECLARE @column NVARCHAR(128); --循环Item字段名

DECLARE @type NVARCHAR(128); --对应字段的类型，char、nchar、varchar、nvarchar等

DECLARE @typeLenght NVARCHAR(128); --对应类型的长度，nchar、nvarchar需要将数值除于2

DECLARE @sql NVARCHAR(MAX); --要拼接执行的sql语句

SET ROWCOUNT 0;

SELECT NULL mykey, c.name, t.name AS [Table], c.name AS [Column], c.collation\_name AS [Collation],

TYPE\_NAME(c.system\_type\_id) AS [TypeName], c.max\_length AS [TypeLength]

INTO #temp

FROM sys.columns c

RIGHT JOIN sys.tables t ON c.object\_id = t.object\_id

WHERE c.collation\_name IS NOT NULL;

-- AND t.name = 'SysLog'

-- AND TYPE\_NAME(c.system\_type\_id) = 'nvarchar';

SET ROWCOUNT 1;

UPDATE #temp

SET mykey = 1;

WHILE @@ROWCOUNT > 0

BEGIN

SET ROWCOUNT 0;

--每次查询第一条记录并赋值到对应变量中

SELECT @table = [Table], @column = [Column], @type = TypeName, @typeLenght = TypeLength

FROM #temp

WHERE mykey = 1;

--nchar、nvarchar需要将数值除于2

IF CONVERT(INT, @typeLenght) > 0

AND ( @type = 'nvarchar'

OR @type = 'nchar' )

BEGIN

SET @typeLenght = CONVERT(NVARCHAR(128), CONVERT(INT, @typeLenght) / 2);

END;

IF @typeLenght = '-1'

BEGIN

SET @typeLenght = N'max';

END;

--拼接sql，注意表名、字段名要带[]，避免Group等关键字

SET @sql = N' ALTER TABLE [' + @table + N'] ALTER COLUMN [' + @column + N'] ' + @type + N'(' + @typeLenght

+ N') COLLATE Latin1\_General\_CI\_AS';

--Try执行

BEGIN TRY

EXEC( @sql );

END TRY

--Catch查询异常结果

BEGIN CATCH

SELECT @sql AS [ASL], ERROR\_MESSAGE() AS msg;

END CATCH;

DELETE #temp

WHERE mykey = 1;

SET ROWCOUNT 1;

UPDATE #temp

SET mykey = 1;

END;

SET ROWCOUNT 0;

DROP TABLE #temp;

USE [master];

GO

ALTER DATABASE DBName SET MULTI\_USER WITH ROLLBACK IMMEDIATE;

GO

## 修改默认排序规则

Alter Table [Craft] Alter Column [Number] [nvarchar](20) COLLATE Latin1\_General\_CI\_AS NOT NULL

------------批量生成修改SQL

Select 'Alter Table ' + quotename(TABLE\_NAME) +

' Alter Column ' + quotename(COLUMN\_NAME) + ' ' + quotename(DATA\_TYPE) +

CASE WHEN CHARACTER\_MAXIMUM\_LENGTH = -1 then '(max)'

WHEN DATA\_TYPE in ('text','ntext') then ''

WHEN CHARACTER\_MAXIMUM\_LENGTH IS NOT NULL

THEN '('+(CONVERT(VARCHAR,CHARACTER\_MAXIMUM\_LENGTH)+')' )

ELSE isnull(CONVERT(VARCHAR,CHARACTER\_MAXIMUM\_LENGTH),' ')

END +

' COLLATE Latin1\_General\_CI\_AS ' +

CASE IS\_NULLABLE

WHEN 'YES' THEN 'NULL'

WHEN 'No' THEN 'NOT NULL'

END +

CHAR(13) + 'if @@error<>0 begin rollback return end'

From INFORMATION\_SCHEMA.COLUMNS

Where 1=1

And collation\_name<>'Latin1\_General\_CI\_AS'

-- And collation\_name is not null

Order By quotename(TABLE\_NAME)

-----修改数据库的排序规则：【注：如果表结构中定义了约束，则无法更改】

修改数据库为单用户访问，可以锁定数据库。

alter database yourdatabase set single\_user with rollback immediate ;

go

alter database yourdatabase collate Chinese\_PRC\_CI\_AS ;

go

alter database yourdataabse set multi\_user;

这样排序规则就被修改过来了。

sqlServer2008 更改服务器默认排序规则

运行cmd，打开命令提示符窗口  
进入 C:\Program Files\Microsoft SQL Server\100\Setup Bootstrap\Release 目录  
输入以下命令  
Setup /Q /ACTION=REBUILDDATABASE /INSTANCENAME=（InstanceName） /SQLSYSADMINACCOUNTS=（accounts） /[ SAPWD= （StrongPassword） ] [ /SQLCOLLATION = （CollationName）]  
其中（）中的内容替换为你自己的实际内容  
sqlServer2008中文版默认排序规则是Chinese\_PRC\_CI\_AS，建议安装的时候不要改，个别数据库需要特殊排序时，对单独数据库指定排序规则，否则会很麻烦

## 计算表占用的空间大小

USE BaDemoDB;

GO

--查看数据库文件占用空间

EXEC sp\_spaceused;

--查询所有表占用的存贮空间

IF EXISTS ( SELECT name, \*

FROM sysobjects

WHERE name = 'p\_SysSpaceUsed'

AND type = 'P' )

DROP PROCEDURE p\_SysSpaceUsed;

GO

CREATE PROCEDURE p\_SysSpaceUsed

AS

-- 20170113

CREATE TABLE #T\_Data

(

TableName VARCHAR(100),

TableRow VARCHAR(100),

Reserved VARCHAR(100),

DataSize VARCHAR(100),

IndexSize VARCHAR(100),

UnUsed VARCHAR(100),

RowInt INT,

DataInt INT

);

DECLARE @T\_Name VARCHAR(100);

DECLARE Rep\_Cursor CURSOR FOR

SELECT name

FROM sysobjects

WHERE xtype = 'u'

ORDER BY name;

OPEN Rep\_Cursor;

FETCH NEXT FROM Rep\_Cursor

INTO @T\_Name;

WHILE @@FETCH\_STATUS = 0

BEGIN

INSERT INTO #T\_Data( TableName, TableRow, Reserved, DataSize, IndexSize, UnUsed )

EXEC sp\_spaceused @T\_Name;

-- Print @T\_Name

FETCH NEXT FROM Rep\_Cursor

INTO @T\_Name;

END;

CLOSE Rep\_Cursor;

DEALLOCATE Rep\_Cursor;

UPDATE #T\_Data

SET DataInt = CONVERT(INT, REPLACE(DataSize, 'KB', '')), RowInt = CONVERT(INT, TableRow);

SELECT TableName, TableRow, Reserved, DataSize, IndexSize, UnUsed

FROM #T\_Data

ORDER BY RowInt DESC;

DROP TABLE #T\_Data;

GO

--Exec p\_SysSpaceUsed

## 链接数据库

EXEC master.dbo.sp\_addlinkedserver @server=N'LinkDB',@srvproduct=N'',@provider='SQLOLEDB',@datasrc='BESTTESTING.IMWORK.NET'

EXEC master.dbo.sp\_addlinkedsrvlogin @rmtsrvname=N'LinkDB',@useself=N'False',@locallogin=NULL,@rmtuser=N'sa',@rmtpassword='siemens'

Select \* From Linkdb.DsEamDb.dbo.Dev

## 分析死锁的方法

常看到死锁的问题,一般都是KILL进程,但如果不查出引起死锁的原因,死锁会时常发生

可以通过查找引起死锁的的操作,就可以方便的解决死锁,现将日常解决问题的方法总结,也许对大家有帮助

1\死锁发生时,通过如下语法,查询出引起死锁的操作

use master

go

declare @spid int,@bl int

DECLARE s\_cur CURSOR FOR

select 0 ,blocked

from (select \* from sysprocesses where blocked>0 ) a

where not exists(select \* from (select \* from sysprocesses where blocked>0 ) b

where a.blocked=spid)

union select spid,blocked from sysprocesses where blocked>0

OPEN s\_cur

FETCH NEXT FROM s\_cur INTO @spid,@bl

WHILE @@FETCH\_STATUS = 0

begin

if @spid =0

select '引起数据库死锁的是: '+ CAST(@bl AS VARCHAR(10)) + '进程号,其执行的SQL语法如下'

else

select '进程号SPID：'+ CAST(@spid AS VARCHAR(10))+ '被' + '进程号SPID：'+ CAST(@bl AS VARCHAR(10)) +'阻塞,其当前进程执行的SQL语法如下'

DBCC INPUTBUFFER (@bl )

FETCH NEXT FROM s\_cur INTO @spid,@bl

end

CLOSE s\_cur

DEALLOCATE s\_cur

exec sp\_who2

2\查找程序/数据库,此t\_sql语法在什么地方使用

3\分析找到的,并解决问题

EG：

/\*

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

引起数据库死锁的是: 71进程号,其执行的SQL语法如下

EventType Parameters EventInfo

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

Language Event 0

select \* from test

insert test values(1,2)

（所影响的行数为 1 行）

DBCC 执行完毕。如果 DBCC 输出了错误信息，请与系统管理员联系。

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

进程号SPID：64被进程号SPID：71阻塞,其当前进程执行的SQL语法如下

EventType Parameters EventInfo

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

Language Event 0

select \* from test

insert test values(1,2)

（所影响的行数为 1 行）

DBCC 执行完毕。如果 DBCC 输出了错误信息，请与系统管理员联系。

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

进程号SPID：65被进程号SPID：64阻塞,其当前进程执行的SQL语法如下

EventType Parameters EventInfo

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

Language Event 0 begin tran

select \* from test with (holdlock)

waitfor time '12:00'

select \* from test

commit

（所影响的行数为 1 行）

DBCC 执行完毕。如果 DBCC 输出了错误信息，请与系统管理员联系。

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

进程号SPID：73被进程号SPID：64阻塞,其当前进程执行的SQL语法如下

EventType Parameters EventInfo

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

Language Event 0 begin tran

select \* from test with (holdlock)

waitfor time '12:00'

select \* from test

commit

（所影响的行数为 1 行）

DBCC 执行完毕。如果 DBCC 输出了错误信息，请与系统管理员联系。

\*/

## 查询当前锁信息

Use master

Select \* From sysprocesses Where dbid=db\_id('Master')

CREATE PROCEDURE #sp\_who\_lock

AS

BEGIN

DECLARE @spid INT

DECLARE @blk INT

DECLARE @count INT

DECLARE @index INT

DECLARE @lock TINYINT

SET @lock = 0

DECLARE @temp\_who\_lock AS TABLE (

id INT identity(1, 1),

spid INT,

blk INT

)

IF @@error <> 0

RETURN @@error

INSERT INTO @temp\_who\_lock (

spid,

blk

)

SELECT 0,

blocked

FROM (

SELECT \*

FROM master..sysprocesses

WHERE blocked > 0

) a

WHERE NOT EXISTS (

SELECT TOP 1 1

FROM master..sysprocesses

WHERE a.blocked = spid

AND blocked > 0

)

UNION

SELECT spid,

blocked

FROM master..sysprocesses

WHERE blocked > 0

IF @@error <> 0

RETURN @@error

SELECT @count = count(1),

@index = 1

FROM @temp\_who\_lock

IF @@error <> 0

RETURN @@error

IF @count = 0

BEGIN

SELECT N'没有阻塞和死锁信息'

RETURN 0

END

WHILE @index <= @count

BEGIN

IF EXISTS (

SELECT TOP 1 1

FROM @temp\_who\_lock a

WHERE id > @index

AND EXISTS (

SELECT TOP 1 1

FROM @temp\_who\_lock

WHERE id <= @index

AND a.blk = spid

)

)

BEGIN

SET @lock = 1

SELECT @spid = spid,

@blk = blk

FROM @temp\_who\_lock

WHERE id = @index

SELECT N'引起数据库死锁的是:' + CAST(@spid AS NVARCHAR(10)) + N'进程号,其执行的SQL语法如下'

SELECT @spid,

@blk

DBCC INPUTBUFFER (@spid)

DBCC INPUTBUFFER (@blk)

END

SET @index = @index + 1

END

IF @lock = 0

BEGIN

SET @index = 1

WHILE @index <= @count

BEGIN

SELECT @spid = spid,

@blk = blk

FROM @temp\_who\_lock

WHERE id = @index

IF @spid = 0

SELECT N'引起阻塞的是:' + CAST(@blk AS NVARCHAR(10)) + N'进程号,其执行的SQL语法如下'

ELSE

SELECT N'进程号SPID：' + CAST(@spid AS NVARCHAR(10)) + N'被进程号SPID：' + CAST(@blk AS NVARCHAR(10)) + N'阻塞,其当前进程执行的SQL语法如下'

DBCC INPUTBUFFER (@spid)

DBCC INPUTBUFFER (@blk)

SET @index = @index + 1

END

END

RETURN 0

END

GO

EXEC #sp\_who\_lock

Select Distinct a.rsc\_dbid,a.rsc\_objid,isnull(b.name,e.name) As TableName,Rtrim(c.Loginame) Loginame,

case a.rsc\_type when 1 then N'NULL 资源（未使用）'when 2 then N'数据库'when 3 then N'文件'when 4 then N'索引'when 5 then N'表'when 6 then N'页'when 7 then N'键'when 8 then N'扩展盘区'when 9 then N'RID（行 ID)'when 10 then N'应用程序'end As 资源类型,

case a.req\_mode when 1 then N'Sch-S（架构稳定性）'when 2 then N'Sch-M（架构修改）'when 3 then N'S（共享）'when 4 then N'U（更新）'when 5 then N'X（排它）'when 6 then N'IS（意向共享）'when 7 then N'IU（意向更新）'when 8 then N'IX（意向排它）'when 9 then N'SIU（共享意向更新）'when 10 then N'SIX（共享意向排它）'when 11 then N'UIX（更新意向排它）'when 12 then N'BU。由大容量操作使用'when 13 then N'RangeS\_S（共享键范围和共享资源锁）'when 14 then N'RangeS\_U（共享键范围和更新资源锁）'when 15 then N'RangeI\_N（插入键范围和空资源锁）'when 16 then N'RangeI\_S'when 17 then N'RangeI\_U'when 18 then N'RangeI\_X'when 19 then N'RangeX\_S'when 21 then N'RangeX\_X'end as 锁请求模式,

case a.req\_status when 1 then N'已授予'when 2 then N'正在转换'when 3 then N'正在等待'end as 锁请求的状态,

a.req\_spid as 内部进程ID,rtrim(c.hostname) 主机名,c.blocked,

case a.req\_ownertype when 1 then N'事务'when 2 then N'游标'when 3 then N'会话'when 4 then N'ExSession' end as 对象类型,

c.cmd,c.net\_library,c.lastwaittype,c.open\_tran as 事务数,rtrim(c.status) as 状态

From master..syslockinfo a

left join SITMesDB..sysobjects b on b.id=a.rsc\_objid

left join tempdb..sysobjects e on e.id=a.rsc\_objid

left join master..sysprocesses c on c.spid=a.req\_spid

Where rsc\_objid>=100

order by a.req\_spid

## 查询每个表的记录数

SELECT o.name AS "Table Name", i.rowcnt AS "Row Count"

FROM sysobjects o, sysindexes i

WHERE i.id = o.id

AND i.indid IN(0,1)

AND o.xtype = 'u' --只统计用户表

AND o.name <> 'sysdiagrams'

ORDER BY i.rowcnt DESC --按行排降序

## 定时执行Proc

在企业管理器－管理－SQL Server 代理－作业中，设置要执行Proc，设置好时间就ＯＫ。

## 简单数据库备份

压缩备份：

BACKUP DATABASE DBName TO DISK =N'D:\Backup\DBName.bak' WITH COMPRESSION

## 自动备份数据库脚本

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

批量备份数据库

将数据库名写在SQL语句中

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

IF EXISTS ( SELECT name, \*

FROM sysobjects

WHERE name = 'p\_SysDbBackup'

AND type = 'P' )

DROP PROCEDURE p\_SysDbBackup;

GO

CREATE PROCEDURE p\_SysDbBackup

AS

-- 最后修改日期：20120925 By Levept

-- 只需设置备份的数据库和保存目录即可！！！

DECLARE @backupfile VARCHAR(1024);

DECLARE @backdesc VARCHAR(1024);

DECLARE @filename VARCHAR(1024);

DECLARE @path VARCHAR(1024);

DECLARE @dbname VARCHAR(1024);

DECLARE @extension\_name VARCHAR(16);

DECLARE tmp\_Cur CURSOR FOR

SELECT name

FROM sys.databases

WHERE name IN

--【在此输入要备份的数据库名】

('CETCCollectData', 'CETCEamDb', 'CETCQcDb')

ORDER BY name;

--【在此输入备份文件保存的目录】

SET @path = N'C:\zBackup\';

SET @extension\_name = N'Bak';

SET @filename = CONVERT(VARCHAR(1024), GETDATE(), 120);

SET @filename = REPLACE(@filename, ':', '.');

SET @filename = @filename + N'.' + @extension\_name;

OPEN tmp\_Cur;

FETCH NEXT FROM tmp\_Cur

INTO @dbname;

WHILE @@FETCH\_STATUS = 0

BEGIN

-- 得到完整目标文件，数据库将备份到这个文件中

SET @backupfile = @path + @dbname + N'\_' + @filename;

PRINT @backupfile;

SET @backdesc = @dbname + N'-Full Database Backup By Levept';

-- 开始备份, COMPRESSION 参数表示压缩，可节省磁盘空间

BACKUP DATABASE @dbname

TO DISK = @backupfile

WITH NOFORMAT, NOINIT, NAME = @backdesc, SKIP, NOREWIND, NOUNLOAD, STATS = 10, COMPRESSION;

FETCH NEXT FROM tmp\_Cur

INTO @dbname;

END;

CLOSE tmp\_Cur;

DEALLOCATE tmp\_Cur;

-- Exec p\_SysDbBackup

GO

在SQL Server 2005数据库中实现自动备份的具体步骤:

1、打开SQL Server Management Studio

2、启动SQL Server代理

3、点击作业->新建作业

4、"常规"中输入作业的名称

5、新建步骤，类型选T-SQL，在下面的命令中输入下面语句

DECLARE @strPath NVARCHAR(200);

SET @strPath = CONVERT(NVARCHAR(19), GETDATE(), 120);

SET @strPath = REPLACE(@strPath, ':', '.');

SET @strPath = N'D:\bak\\_' + N'databasename' + @strPath + N'.bak';

BACKUP DATABASE [databasename]

TO DISK = @strPath

WITH NOINIT, NOUNLOAD, NOSKIP, STATS = 10, NOFORMAT;

（D:\bak\改为自己的备份路径，databasename修改为想备份的数据库的名称）

6、添加计划，设置频率，时间等。

确定，完成。

## 查看备份文件内容

USE DBName;

RESTORE FILELISTONLY FROM DISK = N'D:\Backup\DBName.bak';

GO

## 简单还原数据库脚本

--设置数据库为单用户模式防止其他连接占用无法还原

USE [master];

GO

ALTER DATABASE DBName SET SINGLE\_USER WITH ROLLBACK IMMEDIATE;

GO

--从磁盘备份还原数据库

RESTORE DATABASE DBName

FROM DISK = N'D:\Backup\DBName' WITH FILE = 1,

MOVE N'DBName' TO N'D:\Database\DBName.mdf',

MOVE N'DBName\_log' TO N'D:\Database\vlog.ldf', NOUNLOAD, REPLACE, STATS = 5

GO

--恢复数据库为多用户模式

USE [master];

GO

ALTER DATABASE DBName SET MULTI\_USER WITH ROLLBACK IMMEDIATE;

GO

## 查看存储过程、视图、触发器、函数内容的脚本

SELECT a.name, b.[definition]

FROM sys.all\_objects a, sys.sql\_modules b

WHERE a.is\_ms\_shipped = 0

AND a.object\_id = b.object\_id

AND a.[type] = 'V' --脚本类型，P存储过程、V视图、TR触发器、AF聚合函数

AND a.is\_ms\_shipped = 0 --是否为系统脚本，0为用户脚本

AND LEFT(name, 3) NOT IN ('SP\_', 'XP\_') --可通过名称过滤，例如过滤SP和XP开头的存储过程

## 查询加密存储过程、函数、视图、触发器的原代码

If Exists (Select Name,\*

From SysObjects

Where Name = 'p\_SysDecrypt'

And Type = 'P')

Drop Procedure p\_SysDecrypt

Go

Create Procedure p\_SysDecrypt

(@objectName varchar(50))

AS

begin

set nocount on

--2006.02.22 By Levept

--破解字节不受限制，适用于SQLSERVER2000+存储过程，函数，视图，触发器

begin tran

declare @objectname1 varchar(100),@orgvarbin varbinary(8000)

declare @sql1 nvarchar(4000),@sql2 varchar(8000),@sql3 nvarchar(4000),@sql4 nvarchar(4000)

DECLARE @OrigSpText1 nvarchar(4000), @OrigSpText2 nvarchar(4000) , @OrigSpText3 nvarchar(4000), @resultsp nvarchar(4000)

declare @i int,@status int,@type varchar(10),@parentid int

declare @colid int,@n int,@q int,@j int,@k int,@encrypted int,@number int

select @type=xtype,@parentid=parent\_obj from sysobjects where id=object\_id(@ObjectName)

create table #temp(number int,colid int,ctext varbinary(8000),encrypted int,status int)

insert #temp SELECT number,colid,ctext,encrypted,status FROM syscomments WHERE id = object\_id(@objectName)

select @number=max(number) from #temp

set @k=0

while @k<=@number

begin

if exists(select 1 from syscomments where id=object\_id(@objectname) and number=@k)

begin

if @type='P'

set @sql1=(case when @number>1 then 'ALTER PROCEDURE '+ @objectName +';'+rtrim(@k)+' WITH ENCRYPTION AS '

else 'ALTER PROCEDURE '+ @objectName+' WITH ENCRYPTION AS '

end)

if @type='TR'

set @sql1='ALTER TRIGGER '+@objectname+' ON '+OBJECT\_NAME(@parentid)+' WITH ENCRYPTION FOR INSERT AS PRINT 1 '

if @type='FN' or @type='TF' or @type='IF'

set @sql1=(case @type when 'TF' then

'ALTER FUNCTION '+ @objectName+'(@a char(1)) returns @b table(a varchar(10)) with encryption as begin insert @b select @a return end '

when 'FN' then

'ALTER FUNCTION '+ @objectName+'(@a char(1)) returns char(1) with encryption as begin return @a end'

when 'IF' then

'ALTER FUNCTION '+ @objectName+'(@a char(1)) returns table with encryption as return select @a as a'

end)

if @type='V'

set @sql1='ALTER VIEW '+@objectname+' WITH ENCRYPTION AS SELECT 1 as f'

set @q=len(@sql1)

set @sql1=@sql1+REPLICATE('-',4000-@q)

select @sql2=REPLICATE('-',8000)

set @sql3='exec(@sql1'

select @colid=max(colid) from #temp where number=@k

set @n=1

while @n<=CEILING(1.0\*(@colid-1)/2) and len(@sQL3)<=3996

begin

set @sql3=@sql3+'+@'

set @n=@n+1

end

set @sql3=@sql3+')'

exec sp\_executesql @sql3,N'@Sql1 nvarchar(4000),@ varchar(8000)',@sql1=@sql1,@=@sql2

end

set @k=@k+1

end

set @k=0

while @k<=@number

begin

if exists(select 1 from syscomments where id=object\_id(@objectname) and number=@k)

begin

select @colid=max(colid) from #temp where number=@k

set @n=1

while @n<=@colid

begin

select @OrigSpText1=ctext,@encrypted=encrypted,@status=status FROM #temp WHERE colid=@n and number=@k

SET @OrigSpText3=(SELECT ctext FROM syscomments WHERE id=object\_id(@objectName) and colid=@n and number=@k)

if @n=1

begin

if @type='P'

SET @OrigSpText2=(case when @number>1 then 'CREATE PROCEDURE '+ @objectName +';'+rtrim(@k)+' WITH ENCRYPTION AS '

else 'CREATE PROCEDURE '+ @objectName +' WITH ENCRYPTION AS '

end)

if @type='FN' or @type='TF' or @type='IF'

SET @OrigSpText2=(case @type when 'TF' then

'CREATE FUNCTION '+ @objectName+'(@a char(1)) returns @b table(a varchar(10)) with encryption as begin insert @b select @a return end '

when 'FN' then

'CREATE FUNCTION '+ @objectName+'(@a char(1)) returns char(1) with encryption as begin return @a end'

when 'IF' then

'CREATE FUNCTION '+ @objectName+'(@a char(1)) returns table with encryption as return select @a as a'

end)

if @type='TR'

set @OrigSpText2='CREATE TRIGGER '+@objectname+' ON '+OBJECT\_NAME(@parentid)+' WITH ENCRYPTION FOR INSERT AS PRINT 1 '

if @type='V'

set @OrigSpText2='CREATE VIEW '+@objectname+' WITH ENCRYPTION AS SELECT 1 as f'

set @q=4000-len(@OrigSpText2)

set @OrigSpText2=@OrigSpText2+REPLICATE('-',@q)

end

else

begin

SET @OrigSpText2=REPLICATE('-', 4000)

end

SET @i=1

SET @resultsp = replicate(N'A', (datalength(@OrigSpText1) / 2))

WHILE @i<=datalength(@OrigSpText1)/2

BEGIN

SET @resultsp = stuff(@resultsp, @i, 1, NCHAR(UNICODE(substring(@OrigSpText1, @i, 1)) ^

(UNICODE(substring(@OrigSpText2, @i, 1)) ^

UNICODE(substring(@OrigSpText3, @i, 1)))))

SET @i=@i+1

END

set @orgvarbin=cast(@OrigSpText1 as varbinary(8000))

set @resultsp=(case when @encrypted=1

then @resultsp

else convert(nvarchar(4000),case when @status&2=2 then uncompress(@orgvarbin) else @orgvarbin end)

end)

print @resultsp

set @n=@n+1

end

end

set @k=@k+1

end

drop table #temp

rollback tran

end

-- Exec p\_SysDecrypt 'v\_GetDate'

Go

## 七种数据转换方式：

1. 通过工具DTS的设计器进行导入或导出   
  
　　DTS的设计器功能强大，支持多任务，也是可视化界面，容易操作，但知道的人一般不多，如果只是进行SQL Server数据库中部分表的移动，用这种方法最好，当然，也可以进行全部表的移动。在SQL Server Enterprise Manager中，展开服务器左边的+，选择数据库，右击，选择All tasks/Import Data...(或All tasks/Export Data...)，进入向导模式，按提示一步一步走就行了，里面分得很细，可以灵活的在不同数据源之间复制数据，很方便的。而且可以另存成DTS包，如果以后还有相同的复制任务，直接运行DTS包就行，省时省力。也可以直接打开DTS设计器，方法是展开服务器名称下面的Data Transformation Services，选Local Packages，在右边的窗口中右击，选New Package，就打开了DTS设计器。值得注意的是：如果源数据库要拷贝的表有外键，注意移动的顺序，有时要分批移动，否则外键主键，索引可能丢失，移动的时候选项旁边的提示说的很明白，或者一次性的复制到目标数据库中，再重新建立外键，主键，索引。   
  
　　其实建立数据库时，建立外键，主键，索引的文件应该和建表文件分开，而且用的数据文件也分开，并分别放在不同的驱动器上，有利于数据库的优化。   
  
　　2. 利用Bcp工具   
  
　　这种工具虽然在SQL Server7的版本中不推荐使用，但许多数据库管理员仍很喜欢用它，尤其是用过SQL Server早期版本的人。Bcp有局限性，首先它的界面不是图形化的，其次它只是在SQL Server的表（视图）与文本文件之间进行复制，但它的优点是性能好，开销小，占用内存少，速度快。有兴趣的朋友可以查参考手册。   
  
　　3. 利用备份和恢复   
  
　　先对源数据库进行完全备份，备份到一个设备（device）上，然后把备份文件复制到目的服务器上（恢复的速度快），进行数据库的恢复操作，在恢复的数据库名中填上源数据库的名字（名字必须相同），选择强制型恢复（可以覆盖以前数据库的选项），在选择从设备中进行恢复，浏览时选中备份的文件就行了。这种方法可以完全恢复数据库，包括外键，主键，索引。   
  
　　4. 直接拷贝数据文件   
  
　　把数据库的数据文件（\*.mdf）和日志文件（\*.ldf）都拷贝到目的服务器，在SQL Server Query Analyzer中用语句进行恢复:   
EXEC sp\_attach\_db @dbname = 'test',   
@filename1 = 'd:\mssql7\data\test\_data.mdf',   
@filename2 = 'd:\mssql7\data\test\_log.ldf'   
这样就把test数据库附加到SQL Server中，可以照常使用。如果不想用原来的日志文件，可以用如下的命令：   
EXEC sp\_detach\_db @dbname = 'test'   
EXEC sp\_attach\_single\_file\_db @dbname = 'test',   
@physname = 'd:\mssql7\data\test\_data.mdf'   
这个语句的作用是仅仅加载数据文件，日志文件可以由SQL Server数据库自动添加，但是原来的日志文件中记录的数据就丢失了。

## 批量到出表结构和数据的脚本

注意：数据表的来源库需要批量替换代码中的“来源数据库名”

--设置目标库

USE Test;

GO

SET NOCOUNT ON;

DECLARE @table\_name sysname; --表名，包含dbo

DECLARE @name sysname; --表名

DECLARE @keep\_collation BIT = 0; --是否保持原表的排序规则，1是、0否

DECLARE @t\_sqlstr NVARCHAR(MAX) = N''; --创建表的脚本

DECLARE @i\_sqlstr NVARCHAR(MAX) = N''; --插入数据的脚本

DECLARE @i\_column NVARCHAR(MAX) = N''; --列名集合，逗号分隔

DECLARE @object\_name sysname;

DECLARE @object\_id INT;

BEGIN TRY

BEGIN TRAN;

--创建游标，从源库获取表名

DECLARE c\_table\_name CURSOR FOR

SELECT name

FROM 来源数据库名.sys.objects

WHERE type = 'U'

AND name NOT IN ('sysdiagrams') --将要排除的表放入此处

ORDER BY name;

OPEN c\_table\_name;

FETCH NEXT FROM c\_table\_name

INTO @name;

SELECT @table\_name = 'dbo.' + @name; --要生成创建脚本的表名

WHILE @@FETCH\_STATUS = 0

BEGIN

SELECT @object\_name = '[' + s.name + '].[' + o.name + ']', @object\_id = o.[object\_id]

FROM 来源数据库名.sys.objects o WITH( NOWAIT )

JOIN 来源数据库名.sys.schemas s WITH( NOWAIT )ON o.[schema\_id] = s.[schema\_id]

WHERE s.name + '.' + o.name = @table\_name

AND o.[type] = 'U'

AND o.is\_ms\_shipped = 0;

WITH

index\_column AS

(

SELECT ic.[object\_id], ic.index\_id, ic.is\_descending\_key, ic.is\_included\_column, c.name

FROM 来源数据库名.sys.index\_columns ic WITH( NOWAIT )

JOIN 来源数据库名.sys.columns c WITH( NOWAIT )ON ic.[object\_id] = c.[object\_id]

AND ic.column\_id = c.column\_id

WHERE ic.[object\_id] = @object\_id

),

fk\_columns AS

(

SELECT k.constraint\_object\_id, cname = c.name, rcname = rc.name

FROM 来源数据库名.sys.foreign\_key\_columns k WITH( NOWAIT )

JOIN 来源数据库名.sys.columns rc WITH( NOWAIT )ON rc.[object\_id] = k.referenced\_object\_id

AND rc.column\_id = k.referenced\_column\_id

JOIN 来源数据库名.sys.columns c WITH( NOWAIT )ON c.[object\_id] = k.parent\_object\_id

AND c.column\_id = k.parent\_column\_id

WHERE k.parent\_object\_id = @object\_id

)

SELECT @t\_sqlstr = N'CREATE TABLE ' + @object\_name + CHAR(13) + N'(' + CHAR(13)

+ STUFF(

( SELECT CHAR(9) + ', [' + c.name + '] '

+ CASE

WHEN c.is\_computed = 1 THEN 'AS ' + cc.[definition]

ELSE

UPPER(tp.name)

+ CASE

WHEN tp.name IN ('varchar', 'char', 'varbinary', 'binary') THEN

'(' + CASE

WHEN c.max\_length = -1 THEN 'MAX'

ELSE CAST(c.max\_length AS VARCHAR(5))

END + ')'

WHEN tp.name IN ('nvarchar', 'nchar', 'ntext') THEN

'(' + CASE

WHEN c.max\_length = -1 THEN 'MAX'

ELSE CAST(c.max\_length / 2 AS VARCHAR(5))

END + ')'

WHEN tp.name IN ('datetime2', 'time2', 'datetimeoffset') THEN

'(' + CAST(c.scale AS VARCHAR(5)) + ')'

WHEN tp.name = 'decimal' THEN

'(' + CAST(c.[precision] AS VARCHAR(5)) + ',' + CAST(c.scale AS VARCHAR(5)) + ')'

ELSE ''

END + CASE

WHEN c.collation\_name IS NOT NULL

AND @keep\_collation = 1 THEN ' COLLATE ' + c.collation\_name

ELSE ''

END + CASE

WHEN c.is\_nullable = 1 THEN ' NULL'

ELSE ' NOT NULL'

END + CASE

WHEN dc.[definition] IS NOT NULL THEN ' DEFAULT' + dc.[definition]

ELSE ''

END

+ CASE

WHEN ic.is\_identity = 1 THEN

' IDENTITY(' + CAST(ISNULL(ic.seed\_value, '0') AS CHAR(1)) + ','

+ CAST(ISNULL(ic.increment\_value, '1') AS CHAR(1)) + ')'

ELSE ''

END

END + CHAR(13)

FROM 来源数据库名.sys.columns c WITH( NOWAIT )

JOIN 来源数据库名.sys.types tp WITH( NOWAIT )ON c.user\_type\_id = tp.user\_type\_id

LEFT JOIN 来源数据库名.sys.computed\_columns cc WITH( NOWAIT )ON c.[object\_id] = cc.[object\_id]

AND c.column\_id = cc.column\_id

LEFT JOIN 来源数据库名.sys.default\_constraints dc WITH( NOWAIT )ON c.default\_object\_id != 0

AND c.[object\_id] = dc.parent\_object\_id

AND c.column\_id = dc.parent\_column\_id

LEFT JOIN 来源数据库名.sys.identity\_columns ic WITH( NOWAIT )ON c.is\_identity = 1

AND c.[object\_id] = ic.[object\_id]

AND c.column\_id = ic.column\_id

WHERE c.[object\_id] = @object\_id

ORDER BY c.column\_id

FOR XML PATH(''), TYPE ).value('.', 'NVARCHAR(MAX)'), 1, 2, CHAR(9) + ' ')

+ ISNULL(

( SELECT CHAR(9) + ', CONSTRAINT [' + k.name + '] PRIMARY KEY ('

+ ( SELECT STUFF(

( SELECT ', [' + c.name + '] ' + CASE

WHEN ic.is\_descending\_key = 1 THEN 'DESC'

ELSE 'ASC'

END

FROM 来源数据库名.sys.index\_columns ic WITH( NOWAIT )

JOIN 来源数据库名.sys.columns c WITH( NOWAIT )ON c.[object\_id] = ic.[object\_id]

AND c.column\_id = ic.column\_id

WHERE ic.is\_included\_column = 0

AND ic.[object\_id] = k.parent\_object\_id

AND ic.index\_id = k.unique\_index\_id

FOR XML PATH(N''), TYPE ).value('.', 'NVARCHAR(MAX)'), 1, 2, '')) + ')' + CHAR(13)

FROM 来源数据库名.sys.key\_constraints k WITH( NOWAIT )

WHERE k.parent\_object\_id = @object\_id

AND k.[type] = 'PK' ), '') + N')' + CHAR(13)

+ ISNULL(

( SELECT ( SELECT CHAR(13) + 'ALTER TABLE ' + @object\_name + ' WITH'

+ CASE

WHEN fk.is\_not\_trusted = 1 THEN ' NOCHECK'

ELSE ' CHECK'

END + ' ADD CONSTRAINT [' + fk.name + '] FOREIGN KEY('

+ STUFF(( SELECT ', [' + k.cname + ']'

FROM fk\_columns k

WHERE k.constraint\_object\_id = fk.[object\_id]

FOR XML PATH(''), TYPE ).value('.', 'NVARCHAR(MAX)'), 1, 2, '')

+ ')' + ' REFERENCES [' + SCHEMA\_NAME(ro.[schema\_id]) + '].[' + ro.name

+ '] ('

+ STUFF(( SELECT ', [' + k.rcname + ']'

FROM fk\_columns k

WHERE k.constraint\_object\_id = fk.[object\_id]

FOR XML PATH(''), TYPE ).value('.', 'NVARCHAR(MAX)'), 1, 2, '')

+ ')'

+ CASE

WHEN fk.delete\_referential\_action = 1 THEN ' ON DELETE CASCADE'

WHEN fk.delete\_referential\_action = 2 THEN ' ON DELETE SET NULL'

WHEN fk.delete\_referential\_action = 3 THEN ' ON DELETE SET DEFAULT'

ELSE ''

END

+ CASE

WHEN fk.update\_referential\_action = 1 THEN ' ON UPDATE CASCADE'

WHEN fk.update\_referential\_action = 2 THEN ' ON UPDATE SET NULL'

WHEN fk.update\_referential\_action = 3 THEN ' ON UPDATE SET DEFAULT'

ELSE ''

END + CHAR(13) + 'ALTER TABLE ' + @object\_name + ' CHECK CONSTRAINT ['

+ fk.name + ']' + CHAR(13)

FROM 来源数据库名.sys.foreign\_keys fk WITH( NOWAIT )

JOIN 来源数据库名.sys.objects ro WITH( NOWAIT )ON ro.[object\_id] = fk.referenced\_object\_id

WHERE fk.parent\_object\_id = @object\_id

FOR XML PATH(N''), TYPE ).value('.', 'NVARCHAR(MAX)')), '')

+ ISNULL(

(( SELECT CHAR(13) + 'CREATE' + CASE

WHEN i.is\_unique = 1 THEN ' UNIQUE'

ELSE ''

END + ' NONCLUSTERED INDEX [' + i.name + '] ON '

+ @object\_name + ' ('

+ STUFF(

( SELECT ', [' + c.name + ']' + CASE

WHEN c.is\_descending\_key = 1 THEN ' DESC'

ELSE ' ASC'

END

FROM index\_column c

WHERE c.is\_included\_column = 0

AND c.index\_id = i.index\_id

FOR XML PATH(''), TYPE ).value('.', 'NVARCHAR(MAX)'), 1, 2, '') + ')'

+ ISNULL(

CHAR(13) + 'INCLUDE ('

+ STUFF(( SELECT ', [' + c.name + ']'

FROM index\_column c

WHERE c.is\_included\_column = 1

AND c.index\_id = i.index\_id

FOR XML PATH(''), TYPE ).value('.', 'NVARCHAR(MAX)'), 1, 2, '')

+ ')', '') + CHAR(13)

FROM 来源数据库名.sys.indexes i WITH( NOWAIT )

WHERE i.[object\_id] = @object\_id

AND i.is\_primary\_key = 0

AND i.[type] = 2

FOR XML PATH(''), TYPE ).value('.', 'NVARCHAR(MAX)')), '');

EXEC sys.sp\_executesql @t\_sqlstr;

PRINT CONVERT(NVARCHAR(30), GETDATE(), 21) + N': Create Table ' + @table\_name;

SET @i\_sqlstr = N'';

IF EXISTS ( SELECT \*

FROM 来源数据库名.sys.identity\_columns

WHERE object\_id = @object\_id

AND is\_identity = 1 )

BEGIN

SET @i\_sqlstr = @i\_sqlstr + N' SET IDENTITY\_INSERT ' + @table\_name + N' ON ';

END;

SET @i\_sqlstr = @i\_sqlstr + N' INSERT INTO ' + @table\_name + N' (';

SET @i\_column = ( SELECT '' + name + ', '

FROM 来源数据库名.sys.syscolumns

WHERE id = @object\_id

FOR XML PATH(''));

SET @i\_column = LEFT(@i\_column, LEN(@i\_column) - 1);

SET @i\_sqlstr = @i\_sqlstr + @i\_column + N')';

SET @i\_sqlstr = @i\_sqlstr + N' select ' + @i\_column + N' from 来源数据库名.' + @table\_name;

IF EXISTS ( SELECT \*

FROM 来源数据库名.sys.identity\_columns

WHERE object\_id = @object\_id

AND is\_identity = 1 )

BEGIN

SET @i\_sqlstr = @i\_sqlstr + N' SET IDENTITY\_INSERT ' + @table\_name + N' OFF ';

END;

EXEC sys.sp\_executesql @i\_sqlstr;

PRINT CONVERT(NVARCHAR(30), GETDATE(), 21) + N' Insert Data to ' + @table\_name;

FETCH NEXT FROM c\_table\_name

INTO @name;

SELECT @table\_name = 'dbo.' + @name; --要生成创建脚本的表名

END;

CLOSE c\_table\_name;

DEALLOCATE c\_table\_name;

COMMIT TRAN;

END TRY

BEGIN CATCH

ROLLBACK TRAN;

PRINT ERROR\_MESSAGE();

END CATCH;

## 表注释的添加和维护

--添加表注释

EXECUTE sys.sp\_addextendedproperty @name = N'MS\_Description', @value = N'设备主数据', @level0type = N'USER',

@level0name = N'dbo', @level1type = N'TABLE', @level1name = N'Dev',

@level2type = NULL, @level2name = NULL;

--修改表注释

EXECUTE sys.sp\_updateextendedproperty @name = N'MS\_Description', @value = N'设备主数据', @level0type = N'USER',

@level0name = N'dbo', @level1type = N'TABLE', @level1name = N'Dev',

@level2type = NULL, @level2name = NULL;

--删除表注释

EXECUTE sys.sp\_dropextendedproperty @name = N'MS\_Description', @level0type = N'USER', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Dev', @level2type = NULL,

@level2name = NULL;

--添加字段注释

EXECUTE sys.sp\_addextendedproperty @name = N'MS\_Description', @value = N'设备ID', @level0type = N'SCHEMA',

@level0name = N'dbo', @level1type = N'TABLE', @level1name = N'Dev',

@level2type = N'COLUMN', @level2name = N'DevId';

--更新字段注释

EXECUTE sys.sp\_updateextendedproperty @name = N'MS\_Description', @value = N'设备ID', @level0type = N'SCHEMA',

@level0name = N'dbo', @level1type = N'TABLE', @level1name = N'Dev',

@level2type = N'COLUMN', @level2name = N'DevId';

--删除字段注释

EXECUTE sys.sp\_dropextendedproperty @name = 'MS\_Description', @level0type = N'SCHEMA', @level0name = N'dbo',

@level1type = N'TABLE', @level1name = N'Dev', @level2type = N'COLUMN',

@level2name = N'DevId';

## 导出表的所有注释的脚本

DECLARE @sqlstr NVARCHAR(MAX) = N'';

DECLARE @TableName NVARCHAR(255) = N'表名';

DECLARE @TableDesc NVARCHAR(255);

DECLARE @ColumnName NVARCHAR(255);

DECLARE @ColumnDesc NVARCHAR(255);

DECLARE @IsDrop BIT = 1; --是否产生删除注释的脚本

IF EXISTS ( SELECT \*

FROM sys.extended\_properties

WHERE major\_id = OBJECT\_ID(@TableName)

AND minor\_id = 0 )

BEGIN

SELECT @TableDesc = CONVERT(NVARCHAR(255), value)

FROM sys.extended\_properties

WHERE major\_id = OBJECT\_ID(@TableName)

AND minor\_id = 0;

IF( @IsDrop = 1 )

BEGIN

SET @sqlstr = N'EXECUTE sys.sp\_dropextendedproperty @name = N''MS\_Description'', @level0type = N''USER'',';

SET @sqlstr = @sqlstr + N'@level0name = N''dbo'',@level1type = N''TABLE'', @level1name = N''' + @TableName + N''',';

SET @sqlstr = @sqlstr + N'@level2type = NULL, @level2name = NULL';

--删除表注释

PRINT @sqlstr;

END;

SET @sqlstr = N'EXECUTE sys.sp\_addextendedproperty @name = N''MS\_Description'', @value = N''' + @TableDesc

+ N''',@level0name = N''dbo'',';

SET @sqlstr = @sqlstr + N'@level0type = N''USER'',@level1type = N''TABLE'', @level1name = N''' + @TableName + N''',';

SET @sqlstr = @sqlstr + N'@level2type = NULL, @level2name = NULL';

--创建表注释

PRINT @sqlstr;

END;

DECLARE c\_Column CURSOR FOR

SELECT CONVERT(NVARCHAR(255), a.name) ColumnName, CONVERT(NVARCHAR(255), b.value) ColumnDesc

FROM sys.columns a

LEFT OUTER JOIN sys.extended\_properties b ON b.major\_id = a.object\_id

AND b.minor\_id = a.column\_id

WHERE a.object\_id = OBJECT\_ID(@TableName);

OPEN c\_Column;

FETCH NEXT FROM c\_Column

INTO @ColumnName, @ColumnDesc;

WHILE @@FETCH\_STATUS = 0

BEGIN

IF( @ColumnDesc <> N'' )

BEGIN

IF( @IsDrop = 1 )

BEGIN

SET @sqlstr = N'EXECUTE sys.sp\_dropextendedproperty @name = ''MS\_Description'', @level0type = N''SCHEMA'', @level0name = N''dbo'',';

SET @sqlstr = @sqlstr + N'@level1type = N''TABLE'', @level1name = N''' + @TableName

+ N''', @level2type = N''COLUMN'',';

SET @sqlstr = @sqlstr + N'@level2name = N''' + @ColumnName + N'''';

--删除列注释

PRINT @sqlstr;

END;

SET @sqlstr = N'EXECUTE sys.sp\_addextendedproperty @name = N''MS\_Description'', @value = N''' + @ColumnDesc

+ N''', @level0type = N''SCHEMA'',';

SET @sqlstr = @sqlstr + N'@level0name = N''dbo'', @level1type = N''TABLE'', @level1name = N''' + @TableName

+ N''',';

SET @sqlstr = @sqlstr + N'@level2type = N''COLUMN'', @level2name = N''' + @ColumnName + N'''';

--创建列注释

PRINT @sqlstr;

END;

FETCH NEXT FROM c\_Column

INTO @ColumnName, @ColumnDesc;

END;

CLOSE c\_Column;

DEALLOCATE c\_Column;

## 日志清理-适用于2008以上版本

USE [master]

GO

--设置数据库恢复模式设为SIMPLE

ALTER DATABASE DB\_Name SET RECOVERY SIMPLE WITH NO\_WAIT

GO

ALTER DATABASE DB\_Name SET RECOVERY SIMPLE

GO

USE DB\_Name

GO

--收缩日志，需要指定日志的逻辑名，可通过后面的语句查看逻辑名

DBCC SHRINKFILE (N'DB\_Name\_log' , 11, TRUNCATEONLY)

GO

USE [master]

GO

--设置数据库恢复模式为FULL

ALTER DATABASE DB\_Name SET RECOVERY FULL WITH NO\_WAIT

GO

ALTER DATABASE DB\_Name SET RECOVERY FULL

GO

--查看数据库文件的逻辑名

USE DB\_Name

GO

SELECT file\_id,name from sys.database\_files;

## 分区数据库

全新

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

Use Master;

GO

If Exists (Select name From sys.databases

Where name = N'DsEamDB')

Drop Database DsEamDB;

GO

--【新建分区数据库】

Create Database DsEamDB

On Primary

(Name = N'DB\_Primary',FileName = N'C:\zTestData\Primary\EamDB\_Primary.mdf',SIZE=5,FILEGROWTH=1),

FileGroup Archive

(Name = N'DB\_Archive',FileName = N'C:\zTestData\Archive\EamDB\_Archive.ndf',SIZE=5,FILEGROWTH=1),

FileGroup Part2017

(Name = N'DB\_Part2017',FileName = N'C:\zTestData\Archive\EamDB\_Part2017.ndf',SIZE=5,FILEGROWTH=1),

FileGroup Part2018

(Name = N'DB\_Part2018',FileName = N'C:\zTestData\Archive\EamDB\_Part2018.ndf',SIZE=5,FILEGROWTH=1),

FileGroup Part2019

(Name = N'DB\_Part2019',FileName = N'C:\zTestData\Archive\EamDB\_Part2019.ndf',SIZE=5,FILEGROWTH=1)

Log On

(Name = N'DB\_Journal',FileName = N'C:\zTestData\Primary\EamDB\_Journal.ldf',SIZE=1,FILEGROWTH=10%)

COLLATE Latin1\_General\_CI\_AS

GO

Alter Database DsEamDB SET COMPATIBILITY\_LEVEL = 100

GO

Alter Database DsEamDB SET RECOVERY SIMPLE

GO

Use DsEamDB

GO

--查询数据库分区记录

Select file\_id,name,type\_desc,physical\_name From sys.database\_files

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

--新建分区函数，参数类型是bit，即已归档的数据（Range left：小于等于，Range right ：大于等于）

Create Partition Function EamDB\_ArchivePartitionRange\_Bit(bit)

As Range right For Values(1)

--新建分区函数，参数类型是datetime，即已归档的数据（Range left：小于等于，Range right ：大于等于）

Create Partition Function EamDB\_ArchivePartitionRange\_Date(datetime)

As Range right For Values(N'2018-01-01',N'2019-01-01')

--新建一个分区方案，即已经归档的数据保存到Archiving分区文件上

Create Partition Scheme EamDB\_ArchivePatitionScheme\_Bit

As Partition EamDB\_ArchivePartitionRange\_Bit To ([Primary],[Archive]);

--新建一个分区方案，即已经归档的数据保存到Part-N分区文件上

Create Partition Scheme EamDB\_ArchivePatitionScheme\_Date

As Partition EamDB\_ArchivePartitionRange\_Date To (Part2017,Part2018,Part2019);

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

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

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

--【增加新的数据库分区】

Use Master;

GO

Alter Database DsEamDB Add FileGroup Part2020

GO

Alter Database DsEamDB Add File

(Name = N'DB\_Part2020',FileName = N'C:\zTestData\Archive\EamDB\_Part2020.ndf',SIZE=5,FILEGROWTH=1)

To FileGroup Part2020

GO

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

--增加一个新的分区函数

Use DsEamDB

Alter Partition Scheme EamDB\_ArchivePatitionScheme\_Date Next Used Part2020

Alter Partition Function EamDB\_ArchivePartitionRange\_Date() Split Range(N'2020-01-01')

GO

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

--【Bit】创建一个测试数据表，绑定一个分区方案

Create Table TestArchiveBit

(Archived Bit NOT NULL, CreateDate DateTime)

--指定到对应的分区中

ON EamDB\_ArchivePatitionScheme\_Bit (Archived)

--插入一些新的数据，已供测试

Insert Into TestArchiveBit (Archived, CreateDate) Values (0,'2016-01-01');

Insert Into TestArchiveBit (Archived, CreateDate) Values (0,'2017-02-01');

Insert Into TestArchiveBit (Archived, CreateDate) Values (0,'2018-03-01');

Insert Into TestArchiveBit (Archived, CreateDate) Values (0,'2019-03-01');

--看看每个分区表存放数据的情况，分区一有3条记录，分区2没有记录，即没有归档数据

Select \* From sys.partitions WHERE OBJECT\_NAME(OBJECT\_ID)='TestArchiveBit';

--好了，我们归档一条记录看看

Update TestArchiveBit Set Archived = 1 where CreateDate >= '2017-01-01';

Select \* From sys.partitions WHERE OBJECT\_NAME(OBJECT\_ID)='TestArchiveBit';

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

--【Date】创建一个测试数据表，绑定一个分区方案

Create Table TestArchiveDate

(Archived Bit NOT NULL, CreateDate DateTime)

--指定到对应的分区中

ON EamDB\_ArchivePatitionScheme\_Date (CreateDate)

--插入一些新的数据，已供测试

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2016-06-01');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2017-07-01');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2018-08-01');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2018-12-31');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2018-12-31 23:59:59');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2019-01-01 00:00:00');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2019-09-01 00:00:00');

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2020-01-01 00:00:00');

--

Insert Into TestArchiveDate (Archived, CreateDate) Values (0,'2020-02-02 00:00:00');

--看看每个分区表存放数据的情况，分区一有2条记录，分区二有3条记录，分区三有4条记录。

Select \* From sys.partitions Where OBJECT\_NAME(OBJECT\_ID)='TestArchiveDate';

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

--查询指定分区中的记录

Select \* From TestArchiveDate Where $PARTITION.EamDB\_ArchivePartitionRange\_Date(CreateDate) = 3

--可以通过三个系统视图来查看我们的分区函数，分区方案，边界值点等。

Select \* From sys.partition\_functions

Select \* From sys.partition\_range\_values

Select \* From sys.partition\_schemes

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

--将原有表单进行分区处理

Begin Transaction

Create Clustered Index [ClusteredIndex\_On\_Tmp]

On TestArchiveDate (CreateDate) --指定表名和字段名

WITH (SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF)

On EamDB\_ArchivePatitionScheme\_Date (CreateDate) --指定字段名

Drop Index [ClusteredIndex\_On\_Tmp] On TestArchiveDate

Commit Transaction

Select \* From sys.partitions Where OBJECT\_NAME(OBJECT\_ID)='TestArchiveDate';

增加

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

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

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

------将EamDb数据增加归档处理 Levept 2018.02.02

----1、增加分区存贮

Use Master

GO

--增加：归档分区

Alter Database DsEamDb Add FileGroup Archive

GO

Alter Database DsEamDb Add File

(Name = N'DB\_Archive',FileName = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DSEamDb\_Archive.ndf',SIZE=5,FILEGROWTH=1)

To FileGroup Archive

GO

--增加：2017年分区

Alter Database DsEamDb Add FileGroup Part2017

GO

Alter Database DsEamDb Add File

(Name = N'DB\_Part2017',FileName = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DSEamDb\_Part2017.ndf',SIZE=5,FILEGROWTH=1)

To FileGroup Part2017

GO

--增加：2018年分区

Alter Database DsEamDb Add FileGroup Part2018

GO

Alter Database DsEamDb Add File

(Name = N'DB\_Part2018',FileName = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DSEamDb\_Part2018.ndf',SIZE=5,FILEGROWTH=1)

To FileGroup Part2018

GO

--增加：2019年分区

Alter Database DsEamDb Add FileGroup Part2019

GO

Alter Database DsEamDb Add File

(Name = N'DB\_Part2019',FileName = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DSEamDb\_Part2019.ndf',SIZE=5,FILEGROWTH=1)

To FileGroup Part2019

GO

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

Use DsEamDb

GO

Select file\_id,name,type\_desc,physical\_name From sys.database\_files

----2、增加分区函数

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

--新建分区函数，参数类型是bit，即已归档的数据（Range left：小于等于，Range right ：大于等于）

Create Partition Function EamDB\_ArchivePartitionRange\_Bit(bit)

As Range right For Values(1)

--新建分区函数，参数类型是datetime，即已归档的数据（Range left：小于等于，Range right ：大于等于）

Create Partition Function EamDB\_ArchivePartitionRange\_Date(datetime)

As Range right For Values(N'2018-01-01',N'2019-01-01')

--新建一个分区方案，即已经归档的数据保存到Archiving分区文件上

Create Partition Scheme EamDB\_ArchivePatitionScheme\_Bit

As Partition EamDB\_ArchivePartitionRange\_Bit To ([Primary],[Archive]);

--新建一个分区方案，即已经归档的数据保存到Part-N分区文件上

Create Partition Scheme EamDB\_ArchivePatitionScheme\_Date

As Partition EamDB\_ArchivePartitionRange\_Date To (Part2017,Part2018,Part2019);

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

----3、将原有表单按年归档处理

Begin Transaction

Create Clustered Index [ClusteredIndex\_On\_Tmp]

On Benchmark (CreateDate) --指定表名和字段名

WITH (SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF)

On EamDB\_ArchivePatitionScheme\_Date (CreateDate) --指定字段名

Drop Index [ClusteredIndex\_On\_Tmp] On Benchmark --指定表名

Commit Transaction

--查询每个分区表存放数据的情况

Select index\_id,Case index\_id when 0 then N'数据' else N'索引' end Index\_Type,partition\_number,rows

From sys.partitions Where OBJECT\_NAME(OBJECT\_ID)='Benchmark'

--查询指定分区中的记录

Select '2017' As Year,Count(1) From Benchmark Where $PARTITION.EamDB\_ArchivePartitionRange\_Date(CreateDate) = 1

Union

Select '2018' As Year,Count(1) From Benchmark Where $PARTITION.EamDB\_ArchivePartitionRange\_Date(CreateDate) = 2

Union

Select '2019' As Year,Count(1) From Benchmark Where $PARTITION.EamDB\_ArchivePartitionRange\_Date(CreateDate) = 3

Union

Select '2020' As Year,Count(1) From Benchmark Where $PARTITION.EamDB\_ArchivePartitionRange\_Date(CreateDate) = 4

Union

Select 'Total' As Year,Count(1) From Benchmark

GO

----4、将原有表单归档处理

--为表单增加归档字段

-- Alter Table BenchmarkDetail Add Archived Bit NOT NULL Default 0

Begin Transaction

Create Clustered Index [ClusteredIndex\_On\_Tmp]

On BenchmarkDetail (Archived) --指定表名和字段名

WITH (SORT\_IN\_TEMPDB = OFF, DROP\_EXISTING = OFF, ONLINE = OFF)

On EamDB\_ArchivePatitionScheme\_Bit (Archived) --指定字段名

Update BenchmarkDetail Set Archived = 1 Where StandardWorkTime > 0 --按需执行归档操作

Drop Index [ClusteredIndex\_On\_Tmp] On BenchmarkDetail --指定表名

Commit Transaction

--查询每个分区表存放数据的情况

Select index\_id,Case index\_id when 0 then N'数据' else N'索引' end Index\_Type,partition\_number,rows

From sys.partitions Where OBJECT\_NAME(OBJECT\_ID)='BenchmarkDetail'

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

--【增加新的数据库分区】

Use Master;

GO

Alter Database DsEamDb Add FileGroup Part2020

GO

Alter Database DsEamDb Add File

(Name = N'DB\_Part2020',FileName = N'C:\Program Files\Microsoft SQL Server\MSSQL11.MSSQLSERVER\MSSQL\DATA\DSEamDb\_Part2020.ndf',SIZE=5,FILEGROWTH=1)

To FileGroup Part2020

GO

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

--增加一个新的分区函数

Use DsEamDb

GO

Alter Partition Scheme EamDB\_ArchivePatitionScheme\_Date Next Used Part2020

Alter Partition Function EamDB\_ArchivePartitionRange\_Date() Split Range(N'2020-01-01')

GO

# 常用脚本

## Cast 类型转换

-- 计算百分比

Cast(Cast(M2.InvoiceComplete As Decimal(18,2))/Cast(M2.InvoiceCount As Decimal(18,2))\*100 As Decimal(18,2)) As Rate

-- 计算百分比

Select

Cast(

(Select COUNT(1) From sys.objects Where type = 'U')

As Float)

/

(Select COUNT(1) From sys.objects)

\*100

## Case When

Case具有两种格式。简单Case函数和Case搜索函数。

--简单Case函数

CASE sex

         WHEN '1' THEN '男'

         WHEN '2' THEN '女'

ELSE '其他' END

--Case搜索函数

CASE WHEN sex = '1' THEN '男'

     WHEN sex = '2' THEN '女'

ELSE '其他' END

这两种方式，可以实现相同的功能。简单Case函数的写法相对比较简洁，但是和Case搜索函数相比，功能方面会有些限制，比如写判断式。

Case When Rtrim(dbo.f\_GetSigningContent(:3))<>'' Then 2 Else 1 End

## Begin Tran

Begin Tran

if @@error<>0 begin rollback return end

if @@error<>0 rollback else Commit Tran

**例子1：**

Begin Tran

Create Table Test(sID Int Primary Key)

if @@error<>0 begin rollback return end

Insert Into Test Values (1)

if @@error<>0 begin rollback return end

Select sID From Test

Insert Into Test Values (2)

if @@error<>0 begin rollback return end

Select sID From Test

Insert Into Test Values (3)

if @@error<>0 begin rollback return end

Select sID From Test

Insert Into Test Values (4)

if @@error<>0 begin rollback return end

--测试时，先整体执行上面的语句，再整体执行下面的词句，测试报错后事务回滚。

Select sID From Test

Insert Into Test Values (4)

-- Select sID From Test

if @@error<>0 begin rollback return end

Select sID From Test

Insert Into Test Values (5)

if @@error<>0 rollback else Commit Tran

-- Select sID From Test

-- Drop Table Test

**例子2：**

--建立锁定表SysLock

IF Object\_ID('SysLock') IS NOT NULL

Drop Table SysLock

Go

Create Table SysLock(

LockID int Primary Key,

LockValues int NOT NULL

)

Go

Insert Into SysLock Values(1,1)

Go

--测试锁定表SysLock

Begin Tran

Update SysLock Set LockValues = 1 Where LockID = 1

if @@error<>0 begin rollback return end

Insert Into SysLock Values(3,3)

if @@error<>0 begin rollback return end

Insert Into SysLock Values(2,2)

if @@error<>0 begin rollback return end

Update SysLock Set LockValues = 22 Where LockID = 1

if @@error<>0 begin rollback return end

Select \* From SysLock

--测试时先执行上面的所有语句

Insert Into SysLock Values(1,1)

if @@error<>0 begin rollback return end

Select \* From SysLock

Insert Into SysLock Values(4,4)

if @@error<>0 rollback else Commit Tran

-----删除测试表

IF Object\_ID('SysLock') IS NOT NULL

Drop Table SysLock

Go

## Select Into 生成唯一标识

Identity(Int,1,1) As RANK

Select Identity(Int,1,1) As RANK into XXXX from XXXX

## 手工SQL生成行号

select 行号=(select count(\*) from 表 where 主键 <=a.主键),\* from 表 a

Select Row\_Number() Over(Order By name) As RowID,\* From sys.objects

## 相同条件查询最大值

**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

--Delete SCPGLL

Select \* From SCPGLL l

Inner Join

(

Select ZDRQ,CHBM,ZYXH,PCSN,max(SN) maxSN From SCPGLL

Group by ZDRQ,CHBM,ZYXH,PCSN

Having count(\*)>1

) m On l.SN=m.maxSN

## 两张关联表，删除主表中已经在副表中没有的信息

　　delete from info where not exists ( select \* from infobz where info.infid=infobz.infid )

## 产生随机数

Select Newid() As NewID,Rand() As Rand

Select CHECKSUM(NEWID()) As CheckSumValue, CHECKSUM(NEWID()) As CheckSumValue2

--生成0至99之间任一整数

Select Cast(Floor(Rand()\*100) As int) As Rank

--生成1至100之间任一整数

Select Cast(Ceiling(Rand()\*100) As int) As Rank

## 特殊空格

那个空格的字符是"&nbsp;"

## 查找回车、单引号

--以文本显示结果

Select '第一行' + CHAR(13) + CHAR(10) + '第二行'

Select 'AAAAA' + CHAR(13) + CHAR(10) + 'BBBBB'

必须是CHAR(13)+CHAR(10),

Select \* From CheckItem Where Name Like '%'+CHAR(10)+'%'

Update CheckItem Set Name = Replace(Name,CHAR(10),N'【回车】')

Where Name Like '%'+CHAR(10)+'%'

制表符 CHAR(9)   
换行符 CHAR(10)   
回车 CHAR(13)

--单引号

Select \* From Dev Where DevName Like ('%'+char(39)+'%')

## SQL语句中定义游标

Declare

@v\_Sjbm Char(20),

@v\_Jc Int

Declare rep\_cursor Cursor For

Select SB,LA From XTTMP Where BSM='00971' And LA<>0

Group By SB,LA Order By LA Desc

Open rep\_cursor

Fetch Next From rep\_cursor Into @v\_Sjbm,@v\_Jc

While (@@FETCH\_STATUS=0)

Begin

Update ……………

Fetch Next From rep\_cursor Into @v\_Sjbm,@v\_Jc

End

Close rep\_cursor

Deallocate rep\_cursor

## NewID

NEWID

创建 uniqueidentifier 类型的唯一值。

语法

NEWID ( )

返回类型

uniqueidentifier

示例

A.对变量使用 NEWID 函数

下面的示例使用 NEWID 对声明为 uniqueidentifier 数据类型的变量赋值。在测试该值前，将先打印 uniqueidentifier 数据类型变量的值。

-- Creating a local variable with DECLARE/SET syntax.

DECLARE @myid uniqueidentifier

SET @myid = NEWID()

PRINT 'Value of @myid is: '+ CONVERT(varchar(255), @myid)

下面是结果集：

Value of @myid is: 6F9619FF-8B86-D011-B42D-00C04FC964FF

说明 对于每台计算机，由 NEWID 返回的值不同。所显示的数字仅起解释说明的作用。

B.在 CREATE TABLE 语句中使用 NEWID

下面的示例创建具有 uniqueidentifier 数据类型的 cust 表，并使用 NEWID 将默认值填充到表中。为 NEWID() 赋默认值时，每个新行和现有行均具有 cust\_id 列的唯一值。

-- Creating a table using NEWID for uniqueidentifier data type.

CREATE TABLE cust

(

cust\_id uniqueidentifier NOT NULL DEFAULT newid(),

company varchar(30) NOT NULL,

contact\_name varchar(60) NOT NULL,

address varchar(30) NOT NULL,

city varchar(30) NOT NULL,

state\_province varchar(10) NULL,

postal\_code varchar(10) NOT NULL,

country varchar(20) NOT NULL,

telephone varchar(15) NOT NULL,

fax varchar(15) NULL

)

GO

-- Inserting data into cust table.

INSERT cust

(cust\_id, company, contact\_name, address, city, state\_province,

postal\_code, country, telephone, fax)

VALUES

(newid(), 'Wartian Herkku', 'Pirkko Koskitalo', 'Torikatu 38', 'Oulu', NULL,

'90110', 'Finland', '981-443655', '981-443655')

INSERT cust

(cust\_id, company, contact\_name, address, city, state\_province,

postal\_code, country, telephone, fax)

VALUES

(newid(), 'Wellington Importadora', 'Paula Parente', 'Rua do Mercado, 12', 'Resende', 'SP',

'08737-363', 'Brazil', '(14) 555-8122', '')

INSERT cust

(cust\_id, company, contact\_name, address, city, state\_province,

postal\_code, country, telephone, fax)

VALUES

(newid(), 'Cactus Comidas para Ilevar', 'Patricio Simpson', 'Cerrito 333', 'Buenos Aires', NULL,

'1010', 'Argentina', '(1) 135-5555', '(1) 135-4892')

INSERT cust

(cust\_id, company, contact\_name, address, city, state\_province,

postal\_code, country, telephone, fax)

VALUES

(newid(), 'Ernst Handel', 'Roland Mendel', 'Kirchgasse 6', 'Graz', NULL,

'8010', 'Austria', '7675-3425', '7675-3426')

INSERT cust

(cust\_id, company, contact\_name, address, city, state\_province,

postal\_code, country, telephone, fax)

VALUES

(newid(), 'Maison Dewey', 'Catherine Dewey', 'Rue Joseph-Bens 532', 'Bruxelles', NULL,

'B-1180', 'Belgium', '(02) 201 24 67', '(02) 201 24 68')

GO

C. 使用 uniqueidentifier 和变量赋值

下面的示例声明局部变量 @myid 为 uniqueidentifier 数据类型。然后使用 SET 语句为该变量赋值。

DECLARE @myid uniqueidentifier

SET @myid = 'A972C577-DFB0-064E-1189-0154C99310DAAC12'

GO

## 跨天的一条数据拆成多条

--开始时间

DECLARE @beginDT DATETIME = '2019-05-07 08:58:00.000';

--结束时间

DECLARE @endDT DATETIME = '2019-05-07 09:08:00.000';

SELECT t.d,

CASE

WHEN d = CONVERT(DATE, @beginDT) THEN CASE

WHEN d = CONVERT(DATE, @endDT) THEN DATEDIFF(MINUTE, @beginDT, @endDT)

ELSE DATEDIFF(MINUTE, @beginDT, DATEADD(DAY, 1, d))

END

WHEN d > CONVERT(DATE, @beginDT)

AND d < CONVERT(DATE, @endDT) THEN 24 \* 60

WHEN d = CONVERT(DATE, @endDT) THEN

DATEDIFF(MINUTE, CONVERT(DATETIME, CONVERT(NVARCHAR(10), d, 120) + ' 00:00:00'), @endDT)

ELSE 0

END h

FROM ( SELECT CONVERT(NVARCHAR(10), DATEADD(DAY, number, @beginDT), 120) d

FROM master..spt\_values

WHERE type = 'p'

AND number <= DATEDIFF(DAY, @beginDT, @endDT)) t;

## 取数据库的当前日期为字符型

If Exists (Select Name

From SysObjects

Where Name = 'v\_GetDate'

And Type = 'v')

Drop View v\_GetDate

Go

Create View v\_GetDate As

-- 康贵明：20101023

Select

d1=Ltrim(Str(DatePart(YY,GetDate()))),

d2=Convert(Char(6),GetDate(),112),

d3=Convert(Char(6),GetDate(),12),

d4=Convert(Char(8),GetDate(),112),

d5=Ltrim(Str(Datepart(yy,Getdate())))

+'-'+

Case When Len(Ltrim(Str(Datepart(mm,Getdate()))))=1 then

'0'+Ltrim(Str(Datepart(mm,Getdate()))) else Ltrim(Str(Datepart(mm,Getdate()))) end

+'-'+

Case When Len(Ltrim(Str(Datepart(dd,Getdate()))))=1 then

'0'+Ltrim(Str(Datepart(dd,Getdate()))) else Ltrim(Str(Datepart(dd,Getdate()))) end,

d6=Convert(Char(20),GetDate(),111),

d7=Convert(Char(20),GetDate(),102),

d8=Convert(Char(20),GetDate(),114),

d9=Convert(Char(20),GetDate(),120),

dA=Replace(Replace(Replace(Convert(varchar,Getdate(),120),'-',''),' ',''),':','')

Go

--当月天数

Select Day(DateAdd(MS,-3,DateAdd(m, DateDiff(m,0,GetDate())+1,0)))

--当月第一天

Select DateAdd(d,-Day(GetDate())+1,GetDate())

--当月最后一天

Select DateAdd(d,-Day(GetDate()),DateAdd(m,1,GetDate()))

--当月第一个星期一

Select DateAdd(wk, DateDiff(wk,'', DateAdd(dd, 6 - Day(GetDate()), GetDate())), '')

## 分拆统计字符串

在数据库表tbl1中有一个字段Keywords，它是nvarchar类型，长度为1000，该字段的内容是所要分析的论文的关键字

id keywords

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

1 kw1;kw2;kw3

2 kw2;kw3

3 kw3;kw1;kw4

问题1。

对于在keywords字段中出现的所有关键字集合(上例中关键字集合为{kw1,kw2,kw3,kw4})中的任意一个关键字，要统计它出现的次数（也就是包含该关键字的纪录的条数），然后写到另一张表中。最后的效果就是

keywords count

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

kw1 2

kw2 2

kw3 3

kw4 1

问题2。

在此基础上，要进行组合查询。也就是说在整个关键字集合中任意抽出两个关键字，统计它们在数据库表纪录中同时出现的次数。对于上题，最后效果要是：

keywords count

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

kw1;kw2 1

kw1;kw3 2

kw1;kw4 1

kw2;kw3 2

kw2;kw4 0

kw3;kw4 1

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

--统计示例

--为统计处理专门做的序数表

select top 1000 id=identity(int,1,1) into 序数表 from syscolumns a,syscolumns b

alter table 序数表 add constraint pk\_id\_序数表 primary key(id)

go

--示例数据

create table tbl1(id int,keywords nvarchar(1000))

insert tbl1 select 1,'kw1;kw2;kw3'

union all select 2,'kw2;kw3'

union all select 3,'kw3;kw1;kw4'

go

--第一种统计(计数)

select keyword=substring(a.keywords,b.id,charindex(';',a.keywords+';',b.id)-b.id)

,[count]=count(distinct a.id)

from tbl1 a,序数表 b

where b.id<=len(a.keywords)

and substring(';'+a.keywords,b.id,1)=';'

group by substring(a.keywords,b.id,charindex(';',a.keywords+';',b.id)-b.id)

go

--第二种统计(组合统计)

select keyword=substring(a.keywords,b.id,charindex(';',a.keywords+';',b.id)-b.id)

,[count]=count(distinct a.id),a.id

into #t

from tbl1 a,序数表 b

where b.id<=len(a.keywords)

and substring(';'+a.keywords,b.id,1)=';'

group by substring(a.keywords,b.id,charindex(';',a.keywords+';',b.id)-b.id),a.id

select keyword=a.keyword+';'+b.keyword,[count]=sum(case a.id when b.id then 1 else 0 end)

from #t a,#t b

where a.keyword<b.keyword

group by a.keyword,b.keyword

order by keyword

drop table #t

go

--删除测试环境

drop table tbl1,序数表

/\*--测试结果

--统计1

keyword count

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

kw1 2

kw2 2

kw3 3

kw4 1

（所影响的行数为 4 行）

--统计2

keyword count

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

kw1;kw2 1

kw1;kw3 2

kw1;kw4 1

kw2;kw3 2

kw2;kw4 0

kw3;kw4 1

（所影响的行数为 6 行）

## 字符串拆分为行

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: Insus.NET

-- Create date: 2012-02-26 00:15:00

-- Description: Split the string from the delimiter

-- =============================================

CREATE FUNCTION [dbo].[udf\_SplitString]

(

@Value NVARCHAR(MAX),

@Delimiter CHAR(1)

)

RETURNS @SplitResult TABLE ([ID] INT IDENTITY(1,1),[WORD] NVARCHAR(MAX))

AS

BEGIN

DECLARE @xml XML = CAST('<insus>' + REPLACE(@Value,@Delimiter,'</insus><insus>') + '</insus>' AS XML)

INSERT INTO @SplitResult([WORD]) SELECT n.value('.','NVARCHAR(50)') AS w

FROM @xml.nodes('/insus') AS E(n)

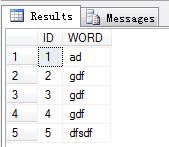
RETURN

END

应用自定义函数：

SELECT [ID],[WORD] FROM [dbo].[udf\_SplitString]('ad;gdf;gdf;gdf;dfsdf',';')

执行结果：



## 字符串分拆查询

有这样的数据

字段1 字段2

2,4,23 3,6,345

23,56,4 3,3,67

取数据的是

查询 字段1中 条件是 4 那么在字段2 在取的是6与 67

结果如下

============

4 6

4 67

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

--处理示例

--测试数据

create table tb(字段1 varchar(10),字段2 varchar(10))

insert tb select '2,4,23' ,'3,6,345'

union all select '23,56,4','3,3,67'

go

--写个自定义函数来处理

create function f\_value(

@a varchar(10),

@b varchar(10),

@c varchar(10)

)returns varchar(10)

as

begin

declare @i int,@pos int

select @a=left(@a,charindex(','+@c+',',','+@a+',')-1)

,@pos=len(@a)-len(replace(@a,',',''))+1

,@i=charindex(',',@b)

while @i>0 and @pos>1

select @b=substring(@b,@i+1,8000)

,@i=charindex(',',@b)

,@pos=@pos-1

return(case @pos when 1

then case when @i>0 then left(@b,@i-1) else @b end

else '' end)

end

go

--查询

declare @a varchar(10)

set @a='23' --查询参数

--查询语句

select A=@a,B=dbo.f\_value(字段1,字段2,@a)

from tb

go

--删除测试

drop table tb

drop function f\_value

/\*--测试结果

A B

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

23 345

23 3

（所影响的行数为 2 行）

## 读取ntext字段

SET TEXTSIZE 64512

TEXTSIZE 的默认设置为 4096 (4 KB)。以下语句将 TEXTSIZE 重置为默认值：

SET TEXTSIZE 0

使用 TEXTPTR 函数可获得传递给 READTEXT 语句的文本指针。

READTEXT 语句用于读取 ntext、text 或 image 数据块。例如，以下查询将返回每个出版商的示例文本数据的前 25 个字符（或第一行）：

USE pubs

DECLARE @textpointer varbinary(16)

SELECT @textpointer = TEXTPTR(pr\_info)

FROM pub\_info

READTEXT pub\_info.pr\_info @textpointer 1 25

## Identity 字段手工插入

-- Create products table.

CREATE TABLE products

(

id INT IDENTITY PRIMARY KEY,

product VARCHAR(40)

);

GO

-- Inserting values into products table.

INSERT INTO products( product )

VALUES( 'screwdriver' );

INSERT INTO products( product )

VALUES( 'hammer' );

INSERT INTO products( product )

VALUES( 'saw' );

INSERT INTO products( product )

VALUES( 'shovel' );

GO

-- Create a gap in the identity values.

DELETE products

WHERE product = 'saw';

GO

SELECT \*

FROM products;

GO

-- Attempt to insert an explicit ID value of 3;

-- should return a warning.

INSERT INTO products( id, product )

VALUES( 3, 'garden shovel' );

GO

-- SET IDENTITY\_INSERT to ON.

SET IDENTITY\_INSERT products ON;

GO

-- Attempt to insert an explicit ID value of 3

INSERT INTO products( id, product )

VALUES( 3, 'garden shovel' );

GO

SELECT \*

FROM products;

GO

SET IDENTITY\_INSERT products OFF;

GO

-- Drop products table.

DROP TABLE products;

GO

## 手工创建与删除用户

/\*

创建SQL Server用户

用户名：UServer

密 码：aaaaaa

如果存在，则删除原用户。

2008-06-12

\*/

Use master

Declare

@username sysname,

@userpassword nvarchar(50)

Set @username=N'UServer' --要创建的登录(用户)名称

Set @userpassword='aaaaaa' --要创建的登录(用户)密码

--删除登录(用户)

If Exists (Select \* From master.dbo.syslogins Where loginname = @username)

Begin

Declare @lcStr nvarchar(4000)

Declare cTmp cursor Local For

Select N'use ['+replace(name,N']',N']]')+N']

If Exists(Select \* From sysusers Where islogin=1 And name=@username)

Exec sp\_revokedbaccess @name\_in\_db = @username'

From master.dbo.sysdatabases

Open cTmp

Fetch cTmp Into @lcStr

While @@fetch\_status=0

Begin

Exec sp\_executesql @lcStr,N'@username sysname',@username

Fetch cTmp Into @lcStr

End

Close cTmp

Deallocate cTmp

Exec sp\_droplogin @loginame = @username

End

--创建登录(用户)

Declare

@logindb nvarchar(132),

@loginlang nvarchar(132)

Select @logindb = N'master', @loginlang = N'简体中文'

If @logindb Is null Or Not Exists (Select \* From master.dbo.sysdatabases Where name = @logindb)

Select @logindb = N'master'

If @loginlang Is null Or (Not Exists (Select \* From master.dbo.syslanguages Where name = @loginlang) And @loginlang <> N'us\_english')

Select @loginlang = @@language

Exec sp\_addlogin @username, @userpassword, @logindb, @loginlang

Exec sp\_addsrvrolemember @username, sysadmin

Exec sp\_addsrvrolemember @username, securityadmin

Exec sp\_addsrvrolemember @username, serveradmin

Exec sp\_addsrvrolemember @username, setupadmin

Exec sp\_addsrvrolemember @username, processadmin

Exec sp\_addsrvrolemember @username, diskadmin

Exec sp\_addsrvrolemember @username, dbcreator

Exec sp\_addsrvrolemember @username, bulkadmin

If Not Exists (Select \* From dbo.sysusers Where name = @username And uid < 16382)

Exec sp\_grantdbaccess @username, @username

Exec sp\_addrolemember N'db\_owner', @username

GO

## 替换函数

Select CHBM,CHMC,Replace(CHMC,'5.1前促销','6.1前促销') From XTCHBMB Where CHMC Like '5.1前促销%'

## 将空格转为‘0’

replace(str(12345,10) , ' ', '0')

## 数值换算为16进制

Create FUNCTION dbo.f\_dec\_hex(@num bigint,@length int)

RETURNS varchar(16)

AS

BEGIN

DECLARE @result varchar(16)

SET @result=''

IF @num<=0 or @length<0

SET @result='0'

ELSE

BEGIN

WHILE @num<>0

SELECT @result=SUBSTRING('0123456789ABCDEF',@num%16+1,1)+@result,@num=@num/16

IF @length>0

SET @result=RIGHT(REPLICATE('0',@length)+@result,@length)

END

RETURN @result

END

## char varchar varchar2 的区别

1．CHAR的长度是固定的，而VARCHAR2的长度是可以变化的， 比如，存储字符串“abc"，对于CHAR (20)，表示你存储的字符将占20个字节(包括17个空字符)，而同样的VARCHAR2 (20)则只占用3个字节的长度，20只是最大值，当你存储的字符小于20时，按实际长度存储。   
2．CHAR的效率比VARCHAR2的效率稍高。   
3．目前VARCHAR是VARCHAR2的同义词。工业标准的VARCHAR类型可以存储空字符串，但是oracle不这样做，尽管它保留以后这样做的权利。Oracle自己开发了一个数据类型VARCHAR2，这个类型不是一个标准的VARCHAR，它将在数据库中varchar列可以存储空字符串的特性改为存储NULL值。如果你想有向后兼容的能力，Oracle建议使用VARCHAR2而不是VARCHAR。   
何时该用CHAR，何时该用varchar2？   
CHAR与VARCHAR2是一对矛盾的统一体，两者是互补的关系.   
VARCHAR2比CHAR节省空间，在效率上比CHAR会稍微差一些，即要想获得效率，就必须牺牲一定的空间，这也就是我们在数据库设计上常说的‘以空间换效率’。   
VARCHAR2虽然比CHAR节省空间，但是如果一个VARCHAR2列经常被修改，而且每次被修改的数据的长度不同，这会引起‘行迁移’(Row Migration)现象，而这造成多余的I/O，是数据库设计和调整中要尽力避免的，在这种情况下用CHAR代替VARCHAR2会更好一些。

## 创建不同排序规则的视图

Create View v\_Dept As

-- 20161223

Select

d.DepartmentCode COLLATE Latin1\_General\_CI\_AS As DeptNumber,

d.DeptName\_EN COLLATE Latin1\_General\_CI\_AS As DeptName

From DWX\_DMS\_DB.dbo.uv\_SITMesDB\_PRM\_Department As d

Go

## 自动生成带表名的SQL

Select 'Insert Into ' + quotename(TABLE\_NAME) +

' Select \* From EAMDb.dbo.' + quotename(TABLE\_NAME)

From INFORMATION\_SCHEMA.TABLES

Where 1=1

Order By quotename(TABLE\_NAME)

## 用SQL语句实现分页的方式

-- 方式一

select top @pageSize \* from company where id not in

(select top @pageSize\*(@pageIndex-1) id from company)

-- 方式三

SELECT \* FROM (

SELECT ROW\_NUMBER() OVER(ORDER BY id asc) AS rownum, id FROM company ) AS D

WHERE rownum BETWEEN (@pageIndex-1)\*@pageSize+1 AND @pageIndex\*@pageSize

ORDER BY id asc

## 删除列的约束（默认值）

【重点】约束的名称是随机产生的，所以要用变量查找

-- 查找约束

Declare @name varchar(50)

Select @name = b.name

From sysobjects b

Inner Join syscolumns a on b.id = a.cdefault

Where a.id = object\_id('S\_User')

And a.name ='CreateDate'

Select @name

-- 删除存在的约束

exec('Alter Table S\_User Drop Constraint ' + @name)

## Row\_Number () Over()

【2005+】

Row\_Number()Over(Order By WorkOrder ASC) As RowNumber

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

Select \*

From (

Select Row\_Number()Over(Order By object\_id) As RankID

,\* From sys.objects s

) m

Where m.RankID between 11 and 20

----以下为手工生成序列号- 从执行效率上来看要低一些

Select \*

From (

Select

(Select COUNT(1) From sys.objects m where m.object\_id<=s.object\_id) As RankID

,\*

From sys.objects s

) m

Where m.RankID between 11 and 20

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

SQL Server2005为我们引入了一个ROW\_NUMBER函数。

 开窗函数指定了分析函数工作的数据窗口大小，这个数据窗口大小可能会随着行的变化而变化，举例如下：  
over后的写法：      
   over（order by salary） 按照salary排序进行累计，order by是个默认的开窗函数  
   over（partition by dept）按照部门分区

   over（partition by dept order by salary）

SELECT A.\*,ROW\_NUMBER() OVER(PARTITION BY A.AREA\_ID ORDER BY A.SER\_ID DESC) RK FROM TEST A;

注意：  
    1.在求第一名成绩的时候，不能用row\_number()，因为如果同班有两个并列第一，row\_number()只返回一个结果;  
select \* from                                                                        
    (                                                                             
    select name,class,s,row\_number()over(partition by class order by s desc) mm from t2  
    )                                                                             
    where mm=1；  
1        95        1  --95有两名但是只显示一个  
2        92        1  
3        99        1 --99有两名但也只显示一个

 2.rank()和dense\_rank()可以将所有的都查找出来：  
如上可以看到采用rank可以将并列第一名的都查找出来；  
     rank()和dense\_rank()区别：  
     --rank()是跳跃排序，有两个第二名时接下来就是第四名；  
select name,class,s,rank()over(partition by class order by s desc) mm from t2  
dss        1        95        1  
ffd        1        95        1  
fda        1        80        3 --直接就跳到了第三  
gds        2        92        1  
cfe        2        74        2  
gf         3        99        1  
ddd        3        99        1  
3dd        3        78        3  
asdf       3        55        4  
adf        3        45        5  
     --dense\_rank()l是连续排序，有两个第二名时仍然跟着第三名  
select name,class,s,dense\_rank()over(partition by class order by s desc) mm from t2  
dss        1        95        1  
ffd        1        95        1  
fda        1        80        2 --连续排序（仍为2）  
gds        2        92        1  
cfe        2        74        2  
gf         3        99        1  
ddd        3        99        1  
3dd        3        78        2  
asdf       3        55        3  
adf        3        45        4

**--sum()over（）的使用**  
select name,class,s, sum(s)over(partition by class order by s desc) mm from t2 --根据班级进行分数求和  
dss        1        95        190  --由于两个95都是第一名，所以累加时是两个第一名的相加  
ffd        1        95        190   
fda        1        80        270  --第一名加上第二名的  
gds        2        92        92  
cfe        2        74        166  
gf         3        99        198  
ddd        3        99        198  
3dd        3        78        276  
asdf       3        55        331  
adf        3        45        376

## CTE-递归查询【2005+】

在sql2005加入了cte实现sql递归，语法如下：

;WITH batchTable(batch) As

(

Select 1 batch

Union ALL

Select batch+1 From batchTable Where batch+1<=100

)

Select \* From batchTable OPTION (MAXRECURSION 1000)

要点一：实现递归查询一定要有递归出口，否则就成了死循环了

要点二：OPTION (MAXRECURSION 1000) 规定最大的递归次数为1000

递归CTE最少包含两个查询(也被称为成员)。第一个查询为定点成员，定点成员只是一个返回有效表的查询，用于递归的基础或定位点。第二个查询被称为递归成员，使该查询称为递归成员的是对CTE名称的递归引用是触发。在逻辑上可以将CTE名称的内部应用理解为前一个查询的结果集。

递归查询没有显式的递归终止条件，只有当第二个递归查询返回空结果集或是超出了递归次数的最大限制时才停止递归。是指递归次数上限的方法是使用MAXRECURION。

 在使用CTE时应注意如下几点：  
1. CTE后面必须直接跟使用CTE的SQL语句（如select、insert、update等），否则，CTE将失效。如下面的SQL语句将无法正常使用CTE：

with cr as  
(  
    select CountryRegionCode from person.CountryRegion where Name like 'C%'  
)  
select \* from person.CountryRegion  -- 应将这条SQL语句去掉  
-- 使用CTE的SQL语句应紧跟在相关的CTE后面 --  
select \* from person.StateProvince where CountryRegionCode in (select \* from cr)

2. CTE后面也可以跟其他的CTE，但只能使用一个with，多个CTE中间用逗号（,）分隔，如下面的SQL语句所示：

with  
cte1 as  
(  
    select \* from table1 where name like 'abc%'  
),  
cte2 as  
(  
    select \* from table2 where id > 20  
),  
cte3 as  
(  
    select \* from table3 where price < 100  
)  
select a.\* from cte1 a, cte2 b, cte3 c where a.id = b.id and a.id = c.id

3. 如果CTE的表达式名称与某个数据表或视图重名，则紧跟在该CTE后面的SQL语句使用的仍然是CTE，当然，后面的SQL语句使用的就是数据表或视图了，如下面的SQL语句所示：

--  table1是一个实际存在的表

with  
table1 as  
(  
    select \* from persons where age < 30  
)  
select \* from table1  --  使用了名为table1的公共表表达式  
select \* from table1  --  使用了名为table1的数据表

4. CTE 可以引用自身，也可以引用在同一 WITH 子句中预先定义的 CTE。不允许前向引用。

5. 不能在 CTE\_query\_definition 中使用以下子句：

（1）COMPUTE 或 COMPUTE BY

（2）ORDER BY（除非指定了 TOP 子句）

（3）INTO

（4）带有查询提示的 OPTION 子句

（5）FOR XML

（6）FOR BROWSE

6. 如果将 CTE 用在属于批处理的一部分的语句中，那么在它之前的语句必须以分号结尾，如下面的SQL所示：

declare @s nvarchar(3)  
set @s = 'C%'  
;  -- 必须加分号  
with  
t\_tree as  
(  
    select CountryRegionCode from person.CountryRegion where Name like @s  
)  
select \* from person.StateProvince where CountryRegionCode in (select \* from t\_tree)

cte可以看作临时表,但是它的生命周期仅存在于访问每一次的TSQL批处理语法中,而一般临时对象的生命周期与连接同在

**一、生命周期**

注意CTE和临时表有个重要的区别,就是生存周期,那么CTE的生存周期到底有多久呢,我们看下面的语句

None--从帖子表中选出前30条放入一个叫CTE\_Temp的临时表  
Nonewith CTE\_Temp AS(  
NoneSelect Top(**30**) \* From Topics  
None)  
None  
None--从CTE\_temp中查出所有记录(第一次),没有问题,返回30条记录  
Noneselect \* from CTE\_Temp  
None  
None--从CTE\_Temp中查询(第二次),报错,提示cte\_temp对象不存在  
Noneselect \* from CTE\_Temp

 紧跟在with语句后面的第一条语句是有效果的,执行第二条前对象就消亡了,也就是说cte的存在周期是with语句的下一条语句,所以,cte不能替代临时表,但是适用于那种只用一次的临时表的场合,在这种情况下,使用cte不会造成日志文件的增大,也不需要手工销毁临时表

/\*

标题：SQL SERVER 2000中查询指定节点及其所有父节点的函数(字符串形式显示)

作者：爱新觉罗·毓华(十八年风雨,守得冰山雪莲花开)

时间：-02-02

地点：新疆乌鲁木齐

\*/

create table tb(id varchar(3) , pid varchar(3) , name varchar(10))

insert into tb values('001' , null , '广东省')

insert into tb values('002' , '001' , '广州市')

insert into tb values('003' , '001' , '深圳市')

insert into tb values('004' , '002' , '天河区')

insert into tb values('005' , '003' , '罗湖区')

insert into tb values('006' , '003' , '福田区')

insert into tb values('007' , '003' , '宝安区')

insert into tb values('008' , '007' , '西乡镇')

insert into tb values('009' , '007' , '龙华镇')

insert into tb values('010' , '007' , '松岗镇')

go

--查询各节点的父路径函数(从父到子)

create function f\_pid1(@id varchar(3)) returns varchar(100)

as

begin

declare @re\_str as varchar(100)

set @re\_str = ''

select @re\_str = name from tb where id = @id

while exists (select 1 from tb where id = @id and pid is not null)

begin

select @id = b.id , @re\_str = b.name + ',' + @re\_str from tb a , tb b where a.id = @id and a.pid = b.id

end

return @re\_str

end

go

--查询各节点的父路径函数(从子到父)

create function f\_pid2(@id varchar(3)) returns varchar(100)

as

begin

declare @re\_str as varchar(100)

set @re\_str = ''

select @re\_str = name from tb where id = @id

while exists (select 1 from tb where id = @id and pid is not null)

begin

select @id = b.id , @re\_str = @re\_str + ',' + b.name from tb a , tb b where a.id = @id and a.pid = b.id

end

return @re\_str

end

go

select \* ,

dbo.f\_pid1(id) [路径(从父到子)] ,

dbo.f\_pid2(id) [路径(从子到父)]

from tb order by id

drop function f\_pid1 , f\_pid2

drop table tb

/\*

id pid name 路径(从父到子) 路径(从子到父)

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

001 NULL 广东省 广东省 广东省

002 001 广州市 广东省,广州市 广州市,广东省

003 001 深圳市 广东省,深圳市 深圳市,广东省

004 002 天河区 广东省,广州市,天河区 天河区,广州市,广东省

005 003 罗湖区 广东省,深圳市,罗湖区 罗湖区,深圳市,广东省

006 003 福田区 广东省,深圳市,福田区 福田区,深圳市,广东省

007 003 宝安区 广东省,深圳市,宝安区 宝安区,深圳市,广东省

008 007 西乡镇 广东省,深圳市,宝安区,西乡镇 西乡镇,宝安区,深圳市,广东省

009 007 龙华镇 广东省,深圳市,宝安区,龙华镇 龙华镇,宝安区,深圳市,广东省

010 007 松岗镇 广东省,深圳市,宝安区,松岗镇 松岗镇,宝安区,深圳市,广东省

（所影响的行数为10 行）

\*/

/\*

标题：SQL SERVER 2005中查询指定节点及其所有父节点的方法(字符串形式显示)

作者：爱新觉罗·毓华(十八年风雨,守得冰山雪莲花开)

时间：-02-02

地点：新疆乌鲁木齐

\*/

create table tb(id varchar(3) , pid varchar(3) , name nvarchar(10))

insert into tb values('001' , null , N'广东省')

insert into tb values('002' , '001' , N'广州市')

insert into tb values('003' , '001' , N'深圳市')

insert into tb values('004' , '002' , N'天河区')

insert into tb values('005' , '003' , N'罗湖区')

insert into tb values('006' , '003' , N'福田区')

insert into tb values('007' , '003' , N'宝安区')

insert into tb values('008' , '007' , N'西乡镇')

insert into tb values('009' , '007' , N'龙华镇')

insert into tb values('010' , '007' , N'松岗镇')

go

;with t as

(

select id , pid = id from tb

union all

select t.id , pid = tb.pid from t inner join tb on t.pid = tb.id

)

select id ,

[路径(从父到子)] = STUFF((SELECT ',' + pid FROM t WHERE id = tb.id order by t.id , t.pid FOR XML PATH('')) , 1 , 1 , ''),

[路径(从子到父)] = STUFF((SELECT ',' + pid FROM t WHERE id = tb.id FOR XML PATH('')) , 1 , 1 , '')

from tb Where id='001'

group by id

order by id

/\*

id 路径(从父到子) 路径(从子到父)

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

001 001 001

002 001,002 002,001

003 001,003 003,001

004 001,002,004 004,002,001

005 001,003,005 005,003,001

006 001,003,006 006,003,001

007 001,003,007 007,003,001

008 001,003,007,008 008,007,003,001

009 001,003,007,009 009,007,003,001

010 001,003,007,010 010,007,003,001

(10 行受影响)

\*/

;with t as

(

select id , name , pid = id , path = cast(name as nvarchar(100)) from tb

union all

select t.id , t.name , pid = tb.pid , path = cast(tb.name as nvarchar(100)) from t join tb on tb.id = t.pid

)

select id ,

name ,

[路径(从父到子)\_1] = pid1,

[路径(从父到子)\_2] = reverse(substring(reverse(path1) , charindex(',' , reverse(path1)) + 1 , len(path1))) ,

[路径(从子到父)\_1] = pid2,

[路径(从子到父)\_2] = substring(path2 , charindex(',' , path2) + 1 , len(path2)) from

(

select id , name ,

pid1 = STUFF((SELECT ',' + pid FROM t WHERE id = tb.id order by t.id , t.pid FOR XML PATH('')) , 1 , 1 , ''),

pid2 = STUFF((SELECT ',' + pid FROM t WHERE id = tb.id FOR XML PATH('')) , 1 , 1 , ''),

path1 = STUFF((SELECT ',' + path FROM t WHERE id = tb.id order by t.id , t.pid FOR XML PATH('')) , 1 , 1 , ''),

path2 = STUFF((SELECT ',' + path FROM t WHERE id = tb.id FOR XML PATH('')) , 1 , 1 , '')

from tb

group by id , name

) m

order by id

/\*

id name 路径(从父到子)\_1 路径(从父到子)\_2 路径(从子到父)\_1 路径(从子到父)\_2

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

001 广东省 001 广东省 001 广东省

002 广州市 001,002 广东省,广州市 002,001 广州市,广东省

003 深圳市 001,003 广东省,深圳市 003,001 深圳市,广东省

004 天河区 001,002,004 广东省,广州市,天河区 004,002,001 天河区,广州市,广东省

005 罗湖区 001,003,005 广东省,深圳市,罗湖区 005,003,001 罗湖区,深圳市,广东省

006 福田区 001,003,006 广东省,深圳市,福田区 006,003,001 福田区,深圳市,广东省

007 宝安区 001,003,007 广东省,深圳市,宝安区 007,003,001 宝安区,深圳市,广东省

008 西乡镇 001,003,007,008 广东省,深圳市,宝安区,西乡镇 008,007,003,001 西乡镇,宝安区,深圳市,广东省

009 龙华镇 001,003,007,009 广东省,深圳市,宝安区,龙华镇 009,007,003,001 龙华镇,宝安区,深圳市,广东省

010 松岗镇 001,003,007,010 广东省,深圳市,宝安区,松岗镇 010,007,003,001 松岗镇,宝安区,深圳市,广东省

(10 行受影响)

\*/

drop table tb

--参考一下实例

--> 生成测试数据表:tb

IF NOT OBJECT\_ID('[tb]') IS NULL

DROP TABLE [tb]

GO

CREATE TABLE [tb](GUID INT IDENTITY,[col1] NVARCHAR(10),[col2] NVARCHAR(20))

INSERT [tb]

SELECT N'A','01' UNION ALL

SELECT N'B','01.01' UNION ALL

SELECT N'C','01.01.01' UNION ALL

SELECT N'F','01.01.01.01' UNION ALL

SELECT N'E','01.01.01.02' UNION ALL

SELECT N'D','01.01.01.03' UNION ALL

SELECT N'O','02' UNION ALL

SELECT N'P','02.01' UNION ALL

SELECT N'Q','02.01.01'

GO

--SELECT \* FROM [tb]

-->SQL查询如下:

---另一种方法

;WITH T AS

(

SELECT \*,PATH=CAST([COL1] AS VARCHAR(1000)) FROM TB A

WHERE NOT EXISTS(

SELECT 1 FROM TB

WHERE A.COL2 LIKE COL2+'%'

AND LEN(A.COL2)>LEN(COL2))

UNION ALL

SELECT A.\*,CAST(PATH+'-->'+A.COL1 AS VARCHAR(1000))

FROM TB A

JOIN T B

ON A.COL2 LIKE B.COL2+'%'

AND LEN(A.COL2)-3=LEN(B.COL2)

)

SELECT \* FROM T ORDER BY LEFT(COL2,2)

/\*

GUID COL1 COL2 PATH

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

1 A 01 A

2 B 01.01 A-->B

3 C 01.01.01 A-->B-->C

4 F 01.01.01.01 A-->B-->C-->F

5 E 01.01.01.02 A-->B-->C-->E

6 D 01.01.01.03 A-->B-->C-->D

7 O 02 O

8 P 02.01 O-->P

9 Q 02.01.01 O-->P-->Q

(9 行受影响)

\*/

;WITH T AS

(

SELECT \*,CAST(COL1 AS VARCHAR(1000)) AS PATH

FROM TB

WHERE COL2 NOT LIKE '%.%'

UNION ALL

SELECT A.\*,CAST(B.PATH+'-->'+A.COL1 AS VARCHAR(1000))

FROM TB A,T B

WHERE A.COL2 LIKE B.COL2+'.[01-99][01-99]'

)

SELECT \* FROM T

ORDER BY LEFT(COL2,2)

/\*

GUID COL1 COL2 PATH

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

1 A 01 A

2 B 01.01 A-->B

3 C 01.01.01 A-->B-->C

4 F 01.01.01.01 A-->B-->C-->F

5 E 01.01.01.02 A-->B-->C-->E

6 D 01.01.01.03 A-->B-->C-->D

7 O 02 O

8 P 02.01 O-->P

9 Q 02.01.01 O-->P-->Q

(9 行受影响)

\*/

**我的例子**

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

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

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

if object\_id('[tb]') is not null drop table [tb]

go

create table [tb]([ID] int,[Name] varchar(10),[ParentID] int,[lft] int,[rgt] int,[Total] int)

insert [tb]

select 2, 'test', 0,1,1,2 union all

select 3, 'test1', 2,8,15,10 union all

select 7, 'test2', 2,16,17,1 union all

select 8, 'test3', 2,18,19,1 union all

select 4, 'test12', 3,9,12,20 union all

select 5, 'test13', 3,13,14,30 union all

select 9, 'test121',4,13,14,30 union all

select 6, 'test122',4,10,11,40 union all

select 10,'test21' ,7,13,14,30 union all

select 11,'test22' ,7,13,14,30 union all

select 12,'test211',10,13,14,30 union all

select 13,'test212',10,13,14,30

select \* from [tb]

--2005的方法：

;With C\_Cte As (

Select a.ID,a.Name,a.ParentID,1 As nlevel,nPath = Cast(a.ID As Varchar(1000))

From tb a Where a.ID=2

Union All

Select b.ID,b.Name,b.ParentID,c.nlevel+1 As nlevel,nPath = Cast(c.nPath + '-->'+ Cast(b.ID As Varchar(100)) As Varchar(1000))

From tb b

Inner Join C\_Cte c on c.ID=b.ParentID

)

Select \* From C\_Cte Order By nPath

------------------------------部门表

Select \* From S\_Depart

--

;With C\_Cte As (

Select Top 1 a.GroupID,a.GroupName,a.ParentGroupID,a.GroupLevel,1 As nlevel,

nPath = Cast(a.GroupID As nvarchar(1000)),nPathName = Cast(a.GroupName As nvarchar(1000))

From S\_Depart a

Where

a.GroupLevel = 1

-- a.GroupID='K0102000'

Union All

Select b.GroupID,b.GroupName,b.ParentGroupID,b.GroupLevel,c.nlevel+1 As nlevel,

nPath = Cast(c.nPath + '-->'+ Cast(b.GroupID As nvarchar(100)) As nvarchar(1000)),

nPathName = Cast(c.nPathName + '-->'+ Cast(b.GroupName As nvarchar(100)) As nvarchar(1000))

From S\_Depart b

Inner Join C\_Cte c on c.GroupID=b.ParentGroupID

--Where b.GroupLevel<=3

) Select \* From C\_Cte d Order By d.nPath

------------------------------工艺树

Select \* From Craft Where LayLevel<=2 Order By Number

--

;With C\_Cte As (

Select a.CraftID,a.Number,a.Name,a.LayLevel,1 As nlevel,

nPath = Cast(a.Number As nvarchar(1000)),nPathName = Cast(a.Name As nvarchar(1000))

From Craft a

Where a.Number='K01'

Union All

Select b.CraftID,b.Number,b.Name,b.LayLevel,c.nlevel+1 As nlevel,

nPath = Cast(c.nPath + '-->'+ Cast(b.Number As nvarchar(100)) As nvarchar(1000)),

nPathName = Cast(c.nPathName + '-->'+ Cast(b.Name As nvarchar(100)) As nvarchar(1000))

From Craft b

Inner Join C\_Cte c on c.CraftID=b.ParentID

Where b.LayLevel<=2

)

Select \* From C\_Cte d Order By d.nPath

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

;with szx as

(

select topid=id,id,name,parentid,lft,rgt,total from tb where parentid=2

union all

select b.topid,a.id,b.name,b.parentid,b.lft,b.rgt,a.total

from tb a join szx b on a.parentid=b.id

)

-- select id=topid,name,parentid,lft,rgt,total=sum(total) from szx group by topid,name,parentid,lft,rgt

Select \* From szx

--2000的方法：

select topid=id,\* into # from tb where parentid=2

insert #

select b.topid,a.id,b.name,b.parentid,b.lft,b.rgt,a.total

from tb a join # b on a.parentid=b.id

where a.id not in (select id from #)

while @@rowcount>0

insert #

select b.topid,a.id,b.name,b.parentid,b.lft,b.rgt,a.total

from tb a join # b on a.parentid=b.id

where a.id not in (select id from #)

select id=topid,name,parentid,lft,rgt,total=sum(total)

from #

group by topid,name,parentid,lft,rgt

drop table #

## Pivot、UnPivot

SELECT ROW\_NUMBER() OVER (order by MaintenanceUser desc)AS Row, MaintenanceUser, UserName

,ISNULL([1],0) [1],ISNULL([2],0) [2],ISNULL([3],0) [3],ISNULL([4],0) [4],ISNULL([5],0) [5],ISNULL([6],0) [6],ISNULL([7],0) [7],ISNULL([8],0) [8],ISNULL([9],0) [9]

,ISNULL([10],0) [10],ISNULL([11],0) [11],ISNULL([12],0) [12] ,ISNULL([13],0) [13],ISNULL([14],0) [14],ISNULL([15],0) [15],ISNULL([16],0) [16],ISNULL([17],0) [17],ISNULL([18],0) [18]

,ISNULL([19],0) [19],ISNULL([20],0) [20],ISNULL([21],0) [21],ISNULL([22],0) [22],ISNULL([23],0) [23],ISNULL([24],0) [24],ISNULL([25],0) [25],ISNULL([26],0) [26],ISNULL([27],0) [27]

,ISNULL([28],0) [28],ISNULL([29],0) [29],ISNULL([30],0) [30],ISNULL([31],0) [31]

from

(

select MaintenanceUser,UserName, DAY(PlanDateStart) CreateDay,

case when AllWorkOrderCount=0 then 0

when AllWorkOrderCount>ISNULL(CompleteWorkOrderCount,0) then 1

when AllWorkOrderCount=ISNULL(CompleteWorkOrderCount,0) then 2

end TaskStatus

from

(

select temp.\*,VU.UserName,isnull(tempcomplete.WorkOrderCount,0) CompleteWorkOrderCount

from (

--本月总工单数量

select WO.MaintenanceUser,Convert(varchar(10), WO.PlanDateStart, 120) PlanDateStart,COUNT(1) as AllWorkOrderCount

from WorkOrder WO

WHERE 1=1 and year(WO.PlanDateStart)=2017 and month(WO.PlanDateStart)=2

group by WO.MaintenanceUser,Convert(varchar(10), WO.PlanDateStart, 120)

) temp

left join

(

--本月完成工单数量

select MaintenanceUser, Convert(varchar(10), WO.PlanDateStart, 120) PlanDateStart,COUNT(1) as WorkOrderCount

from WorkOrder WO

where Status>0 and year(WO.PlanDateStart)=2017 and month(WO.PlanDateStart)=2

group by MaintenanceUser,Convert(varchar(10), WO.PlanDateStart, 120)

) tempcomplete

on temp.MaintenanceUser=tempcomplete.MaintenanceUser and temp.PlanDateStart=tempcomplete.PlanDateStart

left join v\_User VU ON temp.MaintenanceUser=VU.UserNumber

) as T

) a

Pivot (Max(TaskStatus) For CreateDay in([1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14],[15],[16],[17],[18],[19],[20],[21],[22],[23],[24],[25],[26],[27],[28],[29],[30],[31])) As Pi

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

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

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

PIVOT通过将表达式某一列中的唯一值转换为输出中的多个列来旋转表值表达式，并在必要时对最终输出中所需的任何其余列值执行聚合。UNPIVOT与PIVOT执行相反的操作，将表值表达式的列转换为列值。

通俗简单的说：PIVOT就是行转列，UNPIVOT就是列传行

**一、PIVOT实例**

**1. 建表**

建立一个销售情况表，其中，year字段表示年份，quarter字段表示季度，amount字段表示销售额。quarter字段分别用Q1, Q2, Q3, Q4表示一、二、三、四季度。

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif CREATE TABLE SalesByQuarter

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif ( year INT, -- 年份

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif quarter CHAR(**2**), -- 季度

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif amount MONEY -- 总额

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif )

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

**2. 填入表数据**

使用如下程序填入表数据。

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifSET NOCOUNT ON

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif DECLARE @index INT

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif DECLARE @q INT

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif SET @index = **0**

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif DECLARE @year INT

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif while (@index < **30**)

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif BEGIN

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif SET @year = **2005** + (@index % **4**)

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif SET @q = (CAST((RAND() \* **500**) AS INT) % **4**) + **1**

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif INSERT INTO SalesByQuarter VALUES (@year, 'Q' + CAST(@q AS CHAR(**1**)), RAND() \* **10000.00**)

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif SET @index = @index + **1**

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif END

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

**3、如果我们要比较每年中各季度的销售状况，要怎么办呢？有以下两种方法：**

**（1）、使用传统Select的CASE语句查询**  
  
在SQL Server以前的版本里，将行级数据转换为列级数据就要用到一系列CASE语句和聚合查询。虽然这种方式让开发人员具有了对所返回数据进行高度控制的能力，但是编写出这些查询是一件很麻烦的事情。

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif SELECT year as 年份

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif , sum (case when quarter = 'Q1' then amount else **0** end) 一季度

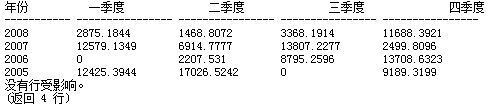
http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif , sum (case when quarter = 'Q2' then amount else **0** end) 二季度

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif , sum (case when quarter = 'Q3' then amount else **0** end) 三季度

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif , sum (case when quarter = 'Q4' then amount else **0** end) 四季度

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif FROM SalesByQuarter GROUP BY year ORDER BY year DESC

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

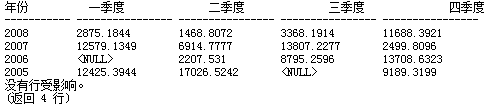
**得到的结果如下：**  


**（2）、使用PIVOT**

由于SQL Server 2005有了新的PIVOT运算符，就不再需要CASE语句和GROUP BY语句了。（每个PIVOT查询都涉及某种类型的聚合，因此你可以忽略GROUP BY语句。）PIVOT运算符让我们能够利用CASE语句查询实现相同的功能，但是你可以用更少的代码就实现，而且看起来更漂亮。

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifSELECT year as 年份, Q1 as 一季度, Q2 as 二季度, Q3 as 三季度, Q4 as 四季度 FROM SalesByQuarter PIVOT (SUM (amount) FOR quarter IN (Q1, Q2, Q3, Q4) ) AS P ORDER BY YEAR DESC

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

**得到的结果如下：**  


**二、通过下面一个实例详细介绍PIVOT的过程**

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifSELECT [星期一],[星期二],[星期三],[星期四],[星期五],[星期六],[星期日]--这里是PIVOT第三步（选择行转列后的结果集的列）这里可以用“\*”表示选择所有列，也可以只选择某些列(也就是某些天)

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifFROM WEEK\_INCOME --这里是PIVOT第二步骤(准备原始的查询结果，因为PIVOT是对一个原始的查询结果集进行转换操作，所以先查询一个结果集出来)这里可以是一个select子查询，但为子查询时候要指定别名，否则语法错误

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gifPIVOT

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif(

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif SUM(INCOME) for [week] in([星期一],[星期二],[星期三],[星期四],[星期五],[星期六],[星期日])--这里是PIVOT第一步骤，也是核心的地方，进行行转列操作。聚合函数SUM表示你需要怎样处理转换后的列的值，是总和(sum)，还是平均(avg)还是min,max等等。例如如果week\_income表中有两条数据并且其week都是“星期一”，其中一条的income是1000,另一条income是500，那么在这里使用sum，行转列后“星期一”这个列的值当然是1500了。后面的for [week] in([星期一],[星期二]...)中 for [week]就是说将week列的值分别转换成一个个列，也就是“以值变列”。但是需要转换成列的值有可能有很多，我们只想取其中几个值转换成列，那么怎样取呢？就是在in里面了，比如我此刻只想看工作日的收入，在in里面就只写“星期一”至“星期五”（注意，in里面是原来week列的值,"以值变列"）。总的来说，SUM(INCOME) for [week] in([星期一],[星期二],[星期三],[星期四],[星期五],[星期六],[星期日])这句的意思如果直译出来，就是说：将列[week]值为"星期一","星期二","星期三","星期四","星期五","星期六","星期日"分别转换成列，这些列的值取income的总和。

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif)TBL--别名一定要写

http://www.studyofnet.com/Codefan-Controls/OutliningIndicators/None.gif

**三.UNPIVOT**

很明显，UN这个前缀表明了，它做的操作是跟PIVOT相反的，即列转行。UNPIVOT操作涉及到以下三个逻辑处理阶段。

1、生成副本

2、提取元素

3、删除带有NULL的行

**UNPIVOT实例**

--UNPIVOT就是列传行

CREATE TABLE pvt (VendorID int, Emp1 int, Emp2 int,

Emp3 int, Emp4 int, Emp5 int);

GO

INSERT INTO pvt VALUES (1,4,3,5,4,4);

INSERT INTO pvt VALUES (2,4,1,5,5,5);

INSERT INTO pvt VALUES (3,4,3,5,4,4);

INSERT INTO pvt VALUES (4,4,2,5,5,4);

INSERT INTO pvt VALUES (5,5,1,5,5,5);

GO

Select \* From pvt

--Unpivot the table.

SELECT VendorID, Employee, Orders FROM

(SELECT VendorID, Emp1, Emp2, Emp3, Emp4, Emp5

FROM pvt) p

UNPIVOT

(Orders FOR Employee IN

(Emp1, Emp2, Emp3, Emp4, Emp5)

)AS unpvt

--

Drop Table pvt

/\*

上面UNPIVOT实例的分析

UNPIVOT的输入是左表表达式P，第一步，先为P中的行生成多个副本，在UNPIVOT中出现的每一列，都会生成一个副本。

因为这里的IN子句有5个列名称，所以要为每个来源行生成5个副本。

结果得到的虚拟表中将新增一个列，用来以字符串格式保存来源列的名称（for和IN之间的，上面例子是 Employee ）。

第二步，根据新增的那一列中的值从来源列中提取出与列名对应的行。

第三步，删除掉结果列值为null的行，完成这个查询。

\*/

## cross apply、outer apply、with rollup、with cube

create table Dept(UNumber nvarchar(10),UName nvarchar(10))

insert into Dept values('001','张三')

insert into Dept values('002','李四')

insert into Dept values('003','王五')

create table ChengJi(UNumber nvarchar(10) , KeMu nvarchar(10) , FenShu int)

insert into ChengJi values('001' , '语文' , 74)

insert into ChengJi values('001' , '数学' , 83)

insert into ChengJi values('001' , '物理' , 93)

insert into ChengJi values('002' , '语文' , 90)

insert into ChengJi values('002' , '数学' , 66)

insert into ChengJi values('002' , '物理' , 88)

insert into ChengJi values('002' , '物理' , 84)

-- cross apply：求有成绩的每个人考试科目中成绩好的前两科

select \* from Dept d

cross apply

(select top 2 \* from ChengJi c where c.UNumber=d.UNumber order by c.FenShu desc) as m

-- outer apply：求全班同学每个人考试科目中成绩好的前两科

select \* from Dept d

outer apply

(select top 2 \* from ChengJi c where c.UNumber=d.UNumber order by c.FenShu desc) as m

-- with rollup：计算每个人每科成绩总数，并计算个人总成绩

Select grouping(c.UNumber) As SumType,

c.UNumber,c.KeMu,SUM(c.FenShu) as SumFenshu,COUNT(1) As CountNum

From ChengJi c

Group By c.UNumber,c.KeMu

with rollup

Order By SumType,c.UNumber,CountNum

-- with cube：计算每个人每科成绩总数，并计算个人总成绩、科目总成绩

Select grouping(c.UNumber) As SumType,

c.UNumber,c.KeMu,SUM(c.FenShu) as SumFenshu,COUNT(1) As CountNum

From ChengJi c

Group By c.UNumber,c.KeMu

with cube

Order By SumType,c.UNumber,CountNum

--删除表单

Drop Table Dept

Drop Table ChengJi

## 储存过程中事务处理范例

--声明错误相关变量

DECLARE @ErrorCode INT; --错误代码

DECLARE @ErrorSeverity INT; --错误级别

DECLARE @ErrorState INT; --错误状态

DECLARE @ErrorText NVARCHAR(500); --错误内容

BEGIN TRY

--开始事务

BEGIN TRAN;

/\*

增、删、改代码块

此处遇到错误会自动跳到CATCH

\*/

--逻辑处理语句，抛出自定义错误

SET @ErrorCode = -3;

SET @ErrorSeverity = 16;

SET @ErrorState = 1;

SET @ErrorText = N'错误内容';

--抛出自定义错误，会被CATCH捕获

RAISERROR(@ErrorText, @ErrorSeverity, @ErrorState);

--提交事务

COMMIT TRAN;

--返回0

RETURN 0;

END TRY

BEGIN CATCH

--回滚事务

ROLLBACK TRAN;

SET @ErrorCode = ERROR\_NUMBER();

SET @ErrorSeverity = ERROR\_SEVERITY();

SET @ErrorState = ERROR\_STATE();

SET @ErrorText = ERROR\_MESSAGE();

--抛出错误

RAISERROR(@ErrorText, @ErrorSeverity, @ErrorState);

RETURN @ErrorCode;

END CATCH;

## 我的存贮过程

----位置信息带路径

If Exists (Select Name

From SysObjects

Where Name = 'v\_Cte\_Craft'

And Type = 'v')

Drop View v\_Cte\_Craft

Go

Create View v\_Cte\_Craft

As

-- 20151205

----位置信息带路径

With C\_Cte As (

Select Top 1

a.CraftID,a.CraftNumber,a.CraftName,a.LayLevel,a.Terminal,a.ParentID,

nPath = Cast(a.CraftNumber As nvarchar(1000)),

nPathDESC = Cast(a.CraftNumber As nvarchar(1000)),

nPathName = Cast(a.CraftName As nvarchar(1000)),

nPathNameDESC = Cast(a.CraftName As nvarchar(1000))

From Craft a

Where a.LayLevel = 1

Union All

Select b.CraftID,b.CraftNumber,b.CraftName,b.LayLevel,b.Terminal,b.ParentID,

nPath = Cast(Cast(c.nPath + '-->' + b.CraftNumber As nvarchar(100)) As nvarchar(1000)),

nPathDESC = Cast(Cast(b.CraftNumber As nvarchar(100)) + '-->' + c.nPathDESC As nvarchar(1000)),

nPathName = Cast(Cast(c.nPathName + '-->' + b.CraftName As nvarchar(100)) As nvarchar(1000)),

nPathNameDESC = Cast(Cast(b.CraftName As nvarchar(100)) + '-->' + c.nPathNameDESC As nvarchar(1000))

From Craft b

Inner Join C\_Cte c on c.CraftID=b.ParentID

Where b.LayLevel > 1

) Select d.CraftID,d.CraftNumber,d.CraftName,d.LayLevel,d.Terminal,d.ParentID,

d.nPath,nPathDESC,d.nPathName,nPathNameDESC From C\_Cte d

go

----当前设备类型下各位置点所有上级至顶级结点List

If Exists (Select Name,\*

From SysObjects

Where Name = 'p\_CraftParentListByCategory'

And Type = 'P')

Drop Procedure p\_CraftParentListByCategory

Go

Create Procedure p\_CraftParentListByCategory

(@v\_CategoryID nvarchar(50),@v\_BenchmarkType int)

As

-- 20151205

----当前设备类型下各位置点所有上级至顶级结点List

Begin Tran

Declare

@v\_CraftID nvarchar(50),

@v\_SQLTxt nvarchar(4000),

@v\_SQLWhere nvarchar(50)

If Exists (Select 1 From DevCategory Where CategoryID=@v\_CategoryID)

Begin

Set @v\_SQLWhere = Case @v\_BenchmarkType

When 1 Then 'And d.IsCheck = 1 '

When 2 Then 'And d.IsOil = 1 '

When 3 Then 'And d.IsKeep = 1 '

Else '' end

Set @v\_SQLTxt = 'Select c.CraftID From Craft c '

Set @v\_SQLTxt = @v\_SQLTxt + 'Inner Join DevInfo i on i.CraftID = c.CraftID And i.Status < 8 '

Set @v\_SQLTxt = @v\_SQLTxt + 'Inner Join Dev d on d.DevID = i.DevID '

Set @v\_SQLTxt = @v\_SQLTxt + @v\_SQLWhere

Set @v\_SQLTxt = @v\_SQLTxt + 'Inner Join DevCategory t on t.CategoryID = d.CategoryID And t.CategoryID = '''

Set @v\_SQLTxt = @v\_SQLTxt + @v\_CategoryID + ''''

Create Table #t\_Tmp(ParentID nvarchar(50))

Exec ('Declare Sql\_Cursor Cursor For ' + @v\_SQLTxt)

Open Sql\_Cursor

Fetch Next From Sql\_Cursor Into @v\_CraftID

While @@FETCH\_STATUS=0

Begin

;With C\_Cte As (

Select Top 1 a.CraftID,a.ParentID From Craft a

Where a.CraftID = @v\_CraftID

Union All

Select b.CraftID,b.ParentID From Craft b

Inner Join C\_Cte c on c.ParentID=b.CraftID

Where b.LayLevel > 0)

Insert Into #t\_Tmp(ParentID) Select d.CraftID From C\_Cte d

Fetch Next From Sql\_Cursor Into @v\_CraftID

End

Close Sql\_Cursor

Deallocate Sql\_Cursor

Select

c.CraftID,c.CraftNumber,c.CraftName,c.LayLevel,c.Terminal,c.ParentID,

d.DevID,d.DevNumber,d.DevName,d.DevModel,i.Status,d.IsCheck,d.IsOil,d.IsKeep

From Craft c

Inner Join (Select ParentID From #t\_Tmp Group By ParentID)t on t.ParentID=c.CraftID

Left Join DevInfo i on i.CraftID=c.CraftID

Left Join Dev d on d.DevID=i.DevID

Order By c.CraftNumber

Drop Table #t\_Tmp

End

if @@error<>0 rollback else Commit Tran

Go

----各位置点所有上级至顶级结点List

If Exists (Select Name,\*

From SysObjects

Where Name = 'p\_CraftParentListByDevInfo'

And Type = 'P')

Drop Procedure p\_CraftParentListByDevInfo

Go

Create Procedure p\_CraftParentListByDevInfo

(@v\_Type int)

As

-- 20151205

----各位置点所有上级至顶级结点List

Begin Tran

Declare

@v\_CraftID nvarchar(50),

@v\_SQLTxt nvarchar(4000),

@v\_SQLWhere nvarchar(50)

If @v\_Type > 0

Begin

if @v\_Type = 1 Set @v\_SQLWhere = 'In ' Else Set @v\_SQLWhere = 'Not In '

Set @v\_SQLTxt = 'Select c.CraftID From Craft c Where c.Terminal = 1 And c.CraftID '

Set @v\_SQLTxt = @v\_SQLTxt + @v\_SQLWhere

Set @v\_SQLTxt = @v\_SQLTxt + '(Select CraftID From DevInfo Where Status < 8)'

Create Table #t\_Tmp(ParentID nvarchar(50))

Exec ('Declare Sql\_Cursor Cursor For ' + @v\_SQLTxt)

Open Sql\_Cursor

Fetch Next From Sql\_Cursor Into @v\_CraftID

While @@FETCH\_STATUS=0

Begin

;With C\_Cte As (

Select Top 1 a.CraftID,a.ParentID From Craft a

Where a.CraftID = @v\_CraftID

Union All

Select b.CraftID,b.ParentID From Craft b

Inner Join C\_Cte c on c.ParentID=b.CraftID

Where b.LayLevel > 0)

Insert Into #t\_Tmp(ParentID) Select d.CraftID From C\_Cte d

Fetch Next From Sql\_Cursor Into @v\_CraftID

End

Close Sql\_Cursor

Deallocate Sql\_Cursor

Select

c.CraftID,c.CraftNumber,c.CraftName,c.LayLevel,c.Terminal,c.ParentID,

d.DevID,d.DevNumber,d.DevName,d.DevModel,i.Status,d.IsCheck,d.IsOil,d.IsKeep

From Craft c

Inner Join (Select ParentID From #t\_Tmp Group By ParentID)t on t.ParentID=c.CraftID

Left Join DevInfo i on i.CraftID=c.CraftID

Left Join Dev d on d.DevID=i.DevID

Order By c.CraftNumber

Drop Table #t\_Tmp

End

If @v\_Type = 0

Begin

Select

c.CraftID,c.CraftNumber,c.CraftName,c.LayLevel,c.Terminal,c.ParentID,

d.DevID,d.DevNumber,d.DevName,d.DevModel,i.Status,d.IsCheck,d.IsOil,d.IsKeep

From Craft c

Left Join DevInfo i on i.CraftID=c.CraftID

Left Join Dev d on d.DevID=i.DevID

Order By c.CraftNumber

End

if @@error<>0 rollback else Commit Tran

Go

## FOR XML PATH【2005+】

  FOR XML PATH 有的人可能知道有的人可能不知道，其实它就是将查询结果集以XML形式展现，有了它我们可以简化我们的查询语句实现一些以前可能需要借助函数活存储过程来完成的工作。那么以一个实例为主.

**一.FOR XML PATH 简单介绍**

那么还是首先来介绍一下FOR XML PATH ，假设现在有一张兴趣爱好表（hobby）用来存放兴趣爱好，表结构如下：

       接下来我们来看应用FOR XML PATH的查询结果语句如下：

SELECT \* FROM @hobby FOR XML PATH

       结果：

[复制代码](javascript:void(0);)

<row>  
  <hobbyID>1</hobbyID>  
  <hName>爬山</hName>  
</row>  
<row>  
  <hobbyID>2</hobbyID>  
  <hName>游泳</hName>  
</row>  
<row>  
  <hobbyID>3</hobbyID>  
  <hName>美食</hName>  
</row>

[复制代码](javascript:void(0);)

      由此可见FOR XML PATH 可以将查询结果根据行输出成XML各式！

      那么，如何改变XML行节点的名称呢？代码如下：

SELECT \* FROM @hobby FOR XML PATH('MyHobby')

      结果一定也可想而知了吧？没错原来的行节点<row> 变成了我们在PATH后面括号()中，自定义的名称<MyHobby>,结果如下：

[复制代码](javascript:void(0);)

<MyHobby>  
  <hobbyID>1</hobbyID>  
  <hName>爬山</hName>  
</MyHobby>  
<MyHobby>  
  <hobbyID>2</hobbyID>  
  <hName>游泳</hName>  
</MyHobby>  
<MyHobby>  
  <hobbyID>3</hobbyID>  
  <hName>美食</hName>  
</MyHobby>

[复制代码](javascript:void(0);)

      这个时候细心的朋友一定又会问那么列节点如何改变呢？还记的给列起别名的关键字AS吗？对了就是用它!代码如下：

SELECT hobbyID as 'MyCode',hName as 'MyName' FROM @hobby FOR XML PATH('MyHobby')

      那么这个时候我们列的节点名称也会编程我们自定义的名称 <MyCode>与<MyName>结果如下：

[复制代码](javascript:void(0);)

<MyHobby>  
  <MyCode>1</MyCode>  
  <MyName>爬山</MyName>  
</MyHobby>  
<MyHobby>  
  <MyCode>2</MyCode>  
  <MyName>游泳</MyName>  
</MyHobby>  
<MyHobby>  
  <MyCode>3</MyCode>  
  <MyName>美食</MyName>  
</MyHobby>

[复制代码](javascript:void(0);)

    噢！ 既然行的节点与列的节点我们都可以自定义，我们是否可以构建我们喜欢的输出方式呢？还是看代码：

SELECT '[ '+hName+' ]' FROM @hobby FOR XML PATH('')

    没错我们还可以通过符号+号，来对字符串类型字段的输出格式进行定义。结果如下：

[ 爬山 ][ 游泳 ][ 美食 ]

    那么其他类型的列怎么自定义？ 没关系，我们将它们转换成字符串类型就行啦！例如：

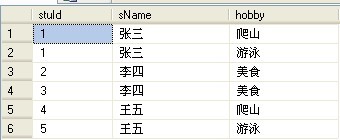
SELECT '{'+STR(hobbyID)+'}','[ '+hName+' ]' FROM @hobby FOR XML PATH('')

    好的 FOR XML PATH就基本介绍到这里吧，更多关于FOR XML的知识请查阅帮助文档！

    接下来我们来看一个FOR XML PATH的应用场景吧！那么开始吧。。。。。。

**二.一个应用场景与FOR XML PATH应用**

首先呢！我们在增加一张学生表，列分别为（stuID,sName,hobby）,stuID代表学生编号，sName代表学生姓名，hobby列存学生的爱好！那么现在表结构如下：

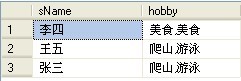


        这时，我们的要求是查询学生表，显示所有学生的爱好的结果集，代码如下：

[复制代码](javascript:void(0);)

SELECT B.sName,LEFT(StuList,LEN(StuList)-**1**) as hobby FROM (  
SELECT sName,  
(SELECT hobby+',' FROM student   
  WHERE sName=A.sName   
  FOR XML PATH('')) AS StuList  
FROM student A   
GROUP BY sName  
) B

[复制代码](javascript:void(0);)

         结果如下:

**分析：** 好的，那么我们来分析一下，首先看这句：

SELECT hobby+',' FROM student   
  WHERE sName=A.sName   
  FOR XML PATH('')

这句是通过FOR XML PATH 将某一姓名如张三的爱好，显示成格式为：“ 爱好1，爱好2，爱好3，”的格式！

那么接着看：

[复制代码](javascript:void(0);)

SELECT B.sName,LEFT(StuList,LEN(StuList)-**1**) as hobby FROM (  
SELECT sName,  
(SELECT hobby+',' FROM student   
  WHERE sName=A.sName   
  FOR XML PATH('')) AS StuList  
FROM student A   
GROUP BY sName  
) B

[复制代码](javascript:void(0);)

剩下的代码首先是将表分组，在执行FOR XML PATH 格式化，这时当还没有执行最外层的SELECT时查询出的结构为:



可以看到StuList列里面的数据都会多出一个逗号，这时随外层的语句:SELECT B.sName,LEFT(StuList,LEN(StuList)-**1**) as hobby  就是来去掉逗号，并赋予有意义的列明！

好啦，太晚啦就说到这里吧！

## [GROUPING](http://www.cnblogs.com/fangyz/p/5813916.html)

**1.grouping sets**

记得前几天第一次接触grouping sets时，笔者的感觉是一脸懵逼。

　　后来一不小心看到msdn上对grouping sets的说明，顿时豁然开朗，其实grouping sets就是由多个group by联合起来，关系如下。

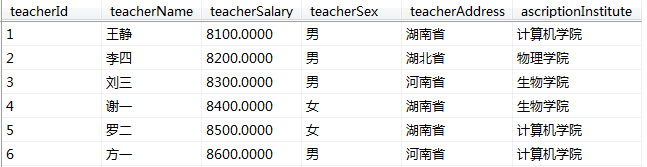
　　select A , B from table group by grouping sets(A, B)   等价于

　　select A , null as B  from table group by A

　　union all

　　select null as A ,  B  from table group by B

　　为了更好的理解我创建了teacher表，表数据如下，查询结果集中左边的为使用union all的group by字句，右边的为使用grouping sets的结果集。

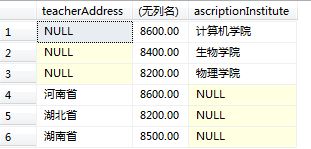


select null as teacherAddress,MAX(teacherSalary),ascriptionInstitute from teacher group by ascriptionInstitute

union all

select teacherAddress,MAX(teacherSalary),NULL as ascriptionInstitute from teacher group by teacherAddress

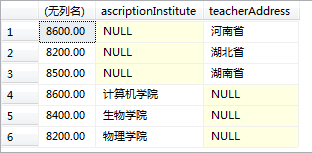
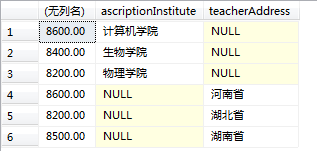
select teacherAddress,MAX(teacherSalary),ascriptionInstitute from teacher group by GROUPING SETS (ascriptionInstitute,teacherAddress)

 　　上面提到grouping sets是等价于带union all的group by子句，之所以是等价而不是等于，从两者结果集中的对比就可以一目了之，那就是它们的顺序不一样。这说明grouping sets并不只是group by的语法糖，这两者内部的执行过程应该是全然不同的，在百度过程中发现大多数答案都是这句话：“聚合是一次性从数据库中取出所有需要操作的数据，在内存中对数据库进行聚合操作并生成结果。而UNION ALL是多次扫描表，将返回的结果进行UNION操作。性能方面grouping sets能减少IO操作但会增加CPU占用时间”。我不理解的地方是一次性取出数据后，是如何在内存中进行聚合操作的？结果集虽然顺序不一样但数据是相同的，这说明依旧进行了联合操作而这个联合操作并不是多次扫描表，关键内部多次是如何扫描的我很好奇？对于性能我想知道为什么会这样子而不是看到现象。另外在grouping sets中如果将括号中的参数换个位置那么结果也将改变，这说明结果集中的顺序与参数的位置也有关，这让我更加好奇grouping sets的内部执行过程了。

select MAX(teacherSalary),ascriptionInstitute ,teacherAddress from teacher group by GROUPING SETS (ascriptionInstitute,teacherAddress)

select MAX(teacherSalary),ascriptionInstitute ,teacherAddress from teacher group by GROUPING SETS (teacherAddress,ascriptionInstitute)

**2.grouping( )**

　　grouping函数用来区分NULL值，这里NULL值有2种情况，一是原本表中的数据就为NULL，二是由rollup、cube、grouping sets生成的NULL值。

　　当为第一种情况中的空值时，grouping(NULL)返回0；当为第二种情况中的空值时，grouping(NULL)返回1。实例如下，从结果中可以看到第二个结果集中原本为null的数据由于grouping函数为1，故显示ROLLUP-NULL字符串。

[复制代码](javascript:void(0);)

select teacherAddress,ascriptionInstitute,COUNT(teacherId ) from teacher group by teacherAddress,ascriptionInstitute

select teacherAddress,ascriptionInstitute,COUNT(teacherId ) from teacher group by rollup(teacherAddress,ascriptionInstitute)

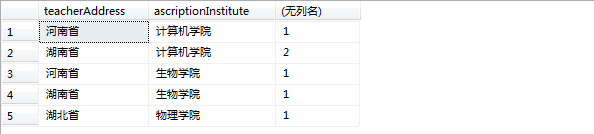
select ISNULL(teacherAddress,case when GROUPING(teacherAddress)=**1** then 'ROLLUP-NULL' end) as teacherAddress,

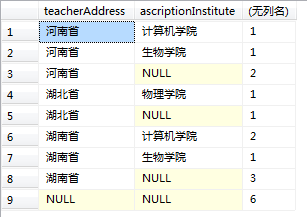
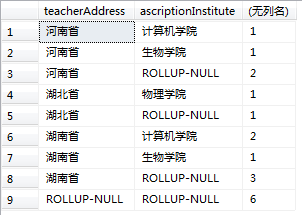
ISNULL(ascriptionInstitute,case when GROUPING(ascriptionInstitute)=**1** then 'ROLLUP-NULL' end) as ascriptionInstitute,

COUNT(teacherId )

from teacher group by rollup(teacherAddress,ascriptionInstitute)

[复制代码](javascript:void(0);)



**3.grouping\_id( )**

　　grouping\_id函数也是计算分组级别的函数，注意如果要使用grouping\_id函数那必须得有group by字句，而且group by字句的中的列与grouping\_id函数的参数必须相等。比如group by A,B，那么必须使用grouping\_id（A,B）。下面用一个等效关系来说明grouping\_id()与grouping()的联系，grouping\_id(A, B)等效于grouping(A) + grouping(B)，但要注意这里的+号不是算术相加，它表示的是二进制数据组合在一起，比如grouping（A）=1，grouping（B）=1，那么grouping\_id(A, B)=11B，也就是十进制数3。原来的表数据执行下面的sql语句结果太多效果不明显，所以我改了下表数据，不过对比两个结果集效果很明显。

[复制代码](javascript:void(0);)

select ISNULL(teacherAddress,case when GROUPING(teacherAddress)=**1** then 'ROLLUP-NULL' end) as teacherAddress,

ISNULL(ascriptionInstitute,case when GROUPING(ascriptionInstitute)=**1** then 'ROLLUP-NULL' end) as ascriptionInstitute,

ISNULL(teacherSex,case when GROUPING(teacherSex)=**1** then 'ROLLUP-NULL' end) as teacherSex,

COUNT(teacherId )

from teacher group by rollup(teacherAddress,ascriptionInstitute,teacherSex)

select ISNULL(teacherAddress,case when GROUPING(teacherAddress)=**1** then 'ROLLUP-NULL' end) as teacherAddress,

ISNULL(ascriptionInstitute,case when GROUPING(ascriptionInstitute)=**1** then 'ROLLUP-NULL' end) as ascriptionInstitute,

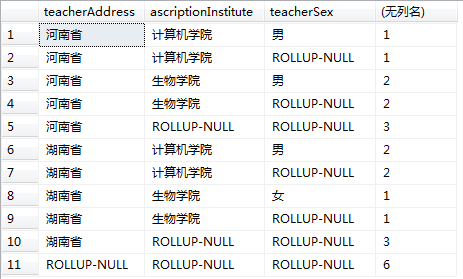
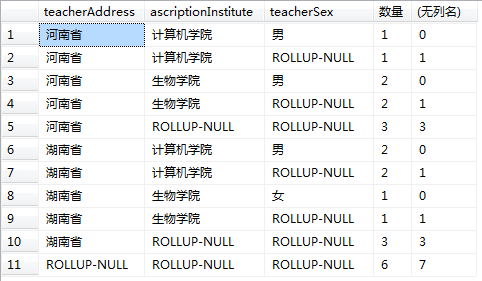
ISNULL(teacherSex,case when GROUPING(teacherSex)=**1** then 'ROLLUP-NULL' end) as teacherSex,

COUNT(teacherId ) as '数量' ,

GROUPING\_ID(teacherAddress,ascriptionInstitute,teacherSex)

from teacher group by rollup(teacherAddress,ascriptionInstitute,teacherSex)

[复制代码](javascript:void(0);)

## 删除xp\_cmdshell命令

use master    
sp\_dropextendedproc 'xp\_cmdshell'    
xp\_cmdshell是进入操作系统的最佳捷径，是数据库留给操作系统的一个大后门。如果你需要这个存储过程，请用这个语句也可以恢复过来。    
sp\_addextendedproc 'xp\_cmdshell', 'xpsql70.dll'

## 用存贮过程删除临时表中的数据

IF EXISTS (SELECT name

FROM sysobjects

WHERE name = 'Proc\_DeleteTmp'

AND type = 'P')

DROP PROCEDURE Proc\_DeleteTmp

GO

CREATE PROCEDURE Proc\_DeleteTmp

AS

-- 最后修改日期：20040925

Begin Tran

Declare

@SqlTxt nvarchar(4000),

@vTblName varchar(20)

Declare Sql\_Cursor Cursor For Select a.TableName From XTTABLE a Where a.A=9

Open Sql\_Cursor

Fetch Next From Sql\_Cursor Into @vTblName

While @@FETCH\_STATUS=0

Begin

Set @SqlTxt='Delete From ' + @vTblName

Exec sp\_executesql @SqlTxt

Fetch Next From Sql\_Cursor Into @vTblName

End

Close Sql\_Cursor

Deallocate Sql\_Cursor

if @@error<>0 rollback else Commit Tran

GO

## 存储过程编写经验和优化措施

　　一、适合读者对象：数据库开发程序员，数据库的数据量很多，涉及到对SP（存储过程）的优化的项目开发人员，对数据库有浓厚兴趣的人。

　　二、介绍：在数据库的开发过程中，经常会遇到复杂的业务逻辑和对数据库的操作，这个时候就会用SP来封装数据库操作。如果项目的SP较多，书写又没有一定的规范，将会影响以后的系统维护困难和大SP逻辑的难以理解，另外如果数据库的数据量大或者项目对SP的性能要求很，就会遇到优化的问题，否则速度有可能很慢，经过亲身经验，一个经过优化过的SP要比一个性能差的SP的效率甚至高几百倍。

　　三、内容：

　　1、开发人员如果用到其他库的Table或View，务必在当前库中建立View来实现跨库操作，最好不要直接使用“databse.dbo.table\_name”，因为sp\_depends不能显示出该SP所使用的跨库table或view，不方便校验。

　　2、开发人员在提交SP前，必须已经使用set showplan on分析过查询计划，做过自身的查询优化检查。

　　3、高程序运行效率，优化应用程序，在SP编写过程中应该注意以下几点：

　　a)SQL的使用规范：

　　　i.　尽量避免大事务操作，慎用holdlock子句，提高系统并发能力。

　　　ii.　尽量避免反复访问同一张或几张表，尤其是数据量较大的表，可以考虑先根据条件提取数据到临时表中，然后再做连接。

　　　iii.　尽量避免使用游标，因为游标的效率较差，如果游标操作的数据超过1万行，那么就应该改写；如果使用了游标，就要尽量避免在游标循环中再进行表连接的操作。

　　　iv.　注意where字句写法，必须考虑语句顺序，应该根据索引顺序、范围大小来确定条件子句的前后顺序，尽可能的让字段顺序与索引顺序相一致，范围从大到小。

　　　v.　不要在where子句中的“=”左边进行函数、算术运算或其他表达式运算，否则系统将可能无法正确使用索引。

　　　vi.　尽量使用exists代替select count(1)来判断是否存在记录，count函数只有在统计表中所有行数时使用，而且count(1)比count(\*)更有效率。

　　　vii.　尽量使用“>=”，不要使用“>”。

　　　viii.　注意一些or子句和union子句之间的替换

　　　ix.　注意表之间连接的数据类型，避免不同类型数据之间的连接。

　　　x.　注意存储过程中参数和数据类型的关系。

　　　xi.　注意insert、update操作的数据量，防止与其他应用冲突。如果数据量超过200个数据页面（400k），那么系统将会进行锁升级，页级锁会升级成表级锁。

　　b)索引的使用规范：

　　　i.　索引的创建要与应用结合考虑，建议大的OLTP表不要超过6个索引。

　　　ii.　尽可能的使用索引字段作为查询条件，尤其是聚簇索引，必要时可以通过index index\_name来强制指定索引

　　　iii.　避免对大表查询时进行table scan，必要时考虑新建索引。

　　　iv.　在使用索引字段作为条件时，如果该索引是联合索引，那么必须使用到该索引中的第一个字段作为条件时才能保证系统使用该索引，否则该索引将不会被使用。

　　　v.　要注意索引的维护，周期性重建索引，重新编译存储过程。

　　c)tempdb的使用规范：

　　　i.　尽量避免使用distinct、order by、group by、having、join、cumpute，因为这些语句会加重tempdb的负担。

　　　ii.　避免频繁创建和删除临时表，减少系统表资源的消耗。

　　　iii.　在新建临时表时，如果一次性插入数据量很大，那么可以使用select into代替create table，避免log，提高速度；如果数据量不大，为了缓和系统表的资源，建议先create table，然后insert。

　　　iv.　如果临时表的数据量较大，需要建立索引，那么应该将创建临时表和建立索引的过程放在单独一个子存储过程中，这样才能保证系统能够很好的使用到该临时表的索引。

　　　 v.　如果使用到了临时表，在存储过程的最后务必将所有的临时表显式删除，先truncate table，然后drop table，这样可以避免系统表的较长时间锁定。

　　　 vi.　慎用大的临时表与其他大表的连接查询和修改，减低系统表负担，因为这种操作会在一条语句中多次使用tempdb的系统表。

　　d)合理的算法使用：

　　根据上面已提到的SQL优化技术和ASE Tuning手册中的SQL优化内容,结合实际应用,采用多种算法进行比较,以获得消耗资源最少、效率最高的方法。具体可用ASE调优命令：set statistics io on, set statistics time on , set showplan on 等。

## 查询MDAC版本号

\HKEY\_LOCAL\_MACHINE\SOFTWARE\Micorsot\DataAccess\Version

## 占比达到80%的记录

IF Object\_ID('table2') IS NOT NULL

DROP TABLE table2

GO

CREATE TABLE Table2 (sName nvarchar(2),QTY int)

GO

INSERT INTO table2

SELECT N'张三',11 UNION ALL

SELECT N'张三',12 UNION ALL

SELECT N'李四',13 UNION ALL

SELECT N'李四',14 UNION ALL

SELECT N'王五',15 UNION ALL

SELECT N'王五',16 UNION ALL

SELECT N'赵六',17 UNION ALL

SELECT N'赵六',18 UNION ALL

SELECT N'赵六',19

-- SELECT \* FROM table2

Select

sName,SUM(QTY) AS 合计, 100\*SUM(QTY)/(SELECT SUM(QTY) FROM Table2) AS 占总百分比

FROM Table2

group by sName

HAVING 100\*SUM(QTY)/(SELECT SUM(QTY) FROM Table2)>=20

order by SUM(QTY) desc

-- 将数据汇总插入待计算表

Select

Row\_Number()Over(Order By SUM(QTY) DESC) As RowNumber,

sName,SUM(QTY) AS SumQty, 100\*SUM(QTY)/(SELECT SUM(QTY) FROM Table2) AS BL

Into Table3

FROM Table2

group by sName

Select \* From Table3

-- 通过循环计算满足条件的记录 、

;With C\_Cte As (

--取第一条记录

Select a.RowNumber,a.sName,a.SumQty,a.BL,a.BL As BLLJ

From Table3 a

Where RowNumber = 1

Union All

--循环提取下一条记录

Select b.RowNumber,b.sName,b.SumQty,b.BL,b.BL+c.BLLJ As BLLJ

From Table3 b

Inner Join C\_Cte c on c.RowNumber = b.RowNumber - 1

)

Select d.\* From C\_Cte d

Inner Join

(

--找到累计达到80%的第一笔

Select Min(t.RowNumber) As MinNumber From C\_Cte t Where t.BLLJ >= 80

) e On d.RowNumber <= e.MinNumber --显示包括80%的记录

/\*

DROP TABLE Table2

DROP TABLE Table3

\*/