|  |
| --- |
| --语 句 功 能 --数据操作 SELECT --从数据库表中检索数据行和列 INSERT --向数据库表添加新数据行 DELETE --从数据库表中删除数据行 UPDATE --更新数据库表中的数据 --数据定义 CREATE TABLE --创建一个数据库表 DROP TABLE --从数据库中删除表 ALTER TABLE --修改数据库表结构 CREATE VIEW --创建一个视图 DROP VIEW --从数据库中删除视图 CREATE INDEX --为数据库表创建一个索引 DROP INDEX --从数据库中删除索引 CREATE PROCEDURE --创建一个存储过程  DROP PROCEDURE --从数据库中删除存储过程 CREATE TRIGGER --创建一个触发器 DROP TRIGGER --从数据库中删除触发器 CREATE SCHEMA --向数据库添加一个新模式 DROP SCHEMA --从数据库中删除一个模式 CREATE DOMAIN --创建一个数据值域 ALTER DOMAIN --改变域定义 DROP DOMAIN --从数据库中删除一个域 --数据控制 GRANT --授予用户访问权限 DENY --拒绝用户访问 REVOKE --解除用户访问权限 --事务控制 COMMIT --结束当前事务 ROLLBACK --中止当前事务 SET TRANSACTION --定义当前事务数据访问特征 --程序化SQL DECLARE --为查询设定游标 EXPLAN --为查询描述数据访问计划 OPEN --检索查询结果打开一个游标 FETCH --检索一行查询结果 CLOSE --关闭游标 PREPARE --为动态执行准备SQL 语句 EXECUTE --动态地执行SQL 语句 DESCRIBE --描述准备好的查询  ---局部变量 declare @id char(10) --set @id = '10010001' select @id = '10010001'   ---全局变量 ---必须以@@开头    --IF ELSE declare @x int @y int @z int select @x = 1 @y = 2 @z=3 if @x > @y print 'x > y' --打印字符串'x > y' else if @y > @z print 'y > z' else print 'z > y'    --CASE use pangu update employee set e\_wage = case when job\_level = ’1’ then e\_wage\*1.08 when job\_level = ’2’ then e\_wage\*1.07 when job\_level = ’3’ then e\_wage\*1.06 else e\_wage\*1.05 end    --WHILE CONTINUE BREAK declare @x int @y int @c int select @x = 1 @y=1 while @x < 3 begin print @x --打印变量x 的值 while @y < 3 begin select @c = 100\*@x + @y print @c --打印变量c 的值 select @y = @y + 1 end select @x = @x + 1 select @y = 1 end    --WAITFOR --例 等待1 小时2 分零3 秒后才执行SELECT 语句 waitfor delay ’01:02:03’ select \* from employee --例 等到晚上11 点零8 分后才执行SELECT 语句 waitfor time ’23:08:00’ select \* from employee     \*\*\*SELECT\*\*\*    select \*(列名) from table\_name(表名) where column\_name operator value ex:(宿主) select \* from stock\_information where stockid = str(nid) stockname = 'str\_name'  stockname like '% find this %'  stockname like '[a-zA-Z]%' --------- ([]指定值的范围) stockname like '[^F-M]%' --------- (^排除指定范围) --------- 只能在使用like关键字的where子句中使用通配符) or stockpath = 'stock\_path' or stocknumber < 1000 and stockindex = 24 not stock\*\*\* = 'man' stocknumber between 20 and 100 stocknumber in(10,20,30) order by stockid desc(asc) --------- 排序，desc-降序，asc-升序 order by 1,2 --------- by列号 stockname = (select stockname from stock\_information where stockid = 4) --------- 子查询 --------- 除非能确保内层select只返回一个行的值， --------- 否则应在外层where子句中用一个in限定符 select distinct column\_name form table\_name --------- distinct指定检索独有的列值，不重复 select stocknumber ,"stocknumber + 10" = stocknumber + 10 from table\_name select stockname , "stocknumber" = count(\*) from table\_name group by stockname --------- group by 将表按行分组,指定列中有相同的值 having count(\*) = 2 --------- having选定指定的组    select \*  from table1, table2  where table1.id \*= table2.id -------- 左外部连接，table1中有的而table2中没有得以null表示 table1.id =\* table2.id -------- 右外部连接     select stockname from table1 union [all] ----- union合并查询结果集，all-保留重复行 select stockname from table2    \*\*\*insert\*\*\*    insert into table\_name (Stock\_name,Stock\_number) value ("xxx","xxxx") value (select Stockname , Stocknumber from Stock\_table2)---value为select语句    \*\*\*update\*\*\*    update table\_name set Stockname = "xxx" [where Stockid = 3] Stockname = default Stockname = null Stocknumber = Stockname + 4    \*\*\*delete\*\*\*    delete from table\_name where Stockid = 3 truncate table\_name ----------- 删除表中所有行，仍保持表的完整性 drop table table\_name --------------- 完全删除表    \*\*\*alter table\*\*\* --- 修改数据库表结构    alter table database.owner.table\_name add column\_name char(2) null ..... sp\_help table\_name ---- 显示表已有特征 create table table\_name (name char(20), age smallint, lname varchar(30)) insert into table\_name select ......... ----- 实现删除列的方法（创建新表） alter table table\_name drop constraint Stockname\_default ---- 删除Stockname的default约束    \*\*\*function(/\*常用函数\*/)\*\*\*    ----统计函数---- AVG --求平均值 COUNT --统计数目 MAX --求最大值 MIN --求最小值 SUM --求和    --AVG use pangu select avg(e\_wage) as dept\_avgWage from employee group by dept\_id --MAX --求工资最高的员工姓名 use pangu select e\_name from employee where e\_wage = (select max(e\_wage) from employee)    --STDEV() --STDEV()函数返回表达式中所有数据的标准差    --STDEVP() --STDEVP()函数返回总体标准差    --VAR() --VAR()函数返回表达式中所有值的统计变异数    --VARP() --VARP()函数返回总体变异数    ----算术函数----    /\*\*\*三角函数\*\*\*/ SIN(float\_expression) --返回以弧度表示的角的正弦 COS(float\_expression) --返回以弧度表示的角的余弦 TAN(float\_expression) --返回以弧度表示的角的正切 COT(float\_expression) --返回以弧度表示的角的余切 /\*\*\*反三角函数\*\*\*/ ASIN(float\_expression) --返回正弦是FLOAT 值的以弧度表示的角 ACOS(float\_expression) --返回余弦是FLOAT 值的以弧度表示的角 ATAN(float\_expression) --返回正切是FLOAT 值的以弧度表示的角 ATAN2(float\_expression1,float\_expression2)  --返回正切是float\_expression1 /float\_expres-sion2的以弧度表示的角 DEGREES(numeric\_expression) --把弧度转换为角度返回与表达式相同的数据类型可为 --INTEGER/MONEY/REAL/FLOAT 类型 RADIANS(numeric\_expression) --把角度转换为弧度返回与表达式相同的数据类型可为 --INTEGER/MONEY/REAL/FLOAT 类型 EXP(float\_expression) --返回表达式的指数值 LOG(float\_expression) --返回表达式的自然对数值 LOG10(float\_expression)--返回表达式的以10 为底的对数值 SQUARE(float\_expression)--返回表达式的平方 SQRT(float\_expression) --返回表达式的平方根 POWER(float\_expression,number) --返回表达式的number次方（number小于1即开方）  /\*\*\*取近似值函数\*\*\*/ CEILING(numeric\_expression) --返回>=表达式的最小整数返回的数据类型与表达式相同可为 --INTEGER/MONEY/REAL/FLOAT 类型 FLOOR(numeric\_expression) --返回<=表达式的最小整数返回的数据类型与表达式相同可为 --INTEGER/MONEY/REAL/FLOAT 类型 ROUND(numeric\_expression) --返回以integer\_expression 为精度的四舍五入值返回的数据 --类型与表达式相同可为INTEGER/MONEY/REAL/FLOAT 类型 ABS(numeric\_expression) --返回表达式的绝对值返回的数据类型与表达式相同可为 --INTEGER/MONEY/REAL/FLOAT 类型 SIGN(numeric\_expression) --测试参数的正负号返回0 零值1 正数或-1 负数返回的数据类型 --与表达式相同可为INTEGER/MONEY/REAL/FLOAT 类型 PI() --返回值为π 即3.1415926535897936 RAND([integer\_expression]) --用任选的[integer\_expression]做种子值得出0-1 间的随机浮点数     ----字符串函数---- ASCII() --函数返回字符表达式最左端字符的ASCII 码值 CHAR() --函数用于将ASCII 码转换为字符 --如果没有输入0 ~ 255 之间的ASCII 码值CHAR 函数会返回一个NULL 值 LOWER() --函数把字符串全部转换为小写 UPPER() --函数把字符串全部转换为大写 STR() --函数把数值型数据转换为字符型数据 LTRIM() --函数把字符串头部的空格去掉 RTRIM() --函数把字符串尾部的空格去掉 LEFT(),RIGHT(),SUBSTRING() --函数返回部分字符串 CHARINDEX(),PATINDEX() --函数返回字符串中某个指定的子串出现的开始位置 SOUNDEX() --函数返回一个四位字符码  --SOUNDEX函数可用来查找声音相似的字符串但SOUNDEX函数对数字和汉字均只返回0 值  DIFFERENCE() --函数返回由SOUNDEX 函数返回的两个字符表达式的值的差异 --0 两个SOUNDEX 函数返回值的第一个字符不同 --1 两个SOUNDEX 函数返回值的第一个字符相同 --2 两个SOUNDEX 函数返回值的第一二个字符相同 --3 两个SOUNDEX 函数返回值的第一二三个字符相同 --4 两个SOUNDEX 函数返回值完全相同     QUOTENAME() --函数返回被特定字符括起来的字符串 /\*select quotename('abc', '{') quotename('abc') 运行结果如下 ----------------------------------{ {abc} [abc]\*/    REPLICATE() --函数返回一个重复character\_expression 指定次数的字符串 /\*select replicate('abc', 3) replicate( 'abc', -2) 运行结果如下 ----------- ----------- abcabcabc NULL\*/    REVERSE() --函数将指定的字符串的字符排列顺序颠倒 REPLACE() --函数返回被替换了指定子串的字符串 /\*select replace('abc123g', '123', 'def') 运行结果如下 ----------- ----------- abcdefg\*/    SPACE() --函数返回一个有指定长度的空白字符串 STUFF() --函数用另一子串替换字符串指定位置长度的子串     ----数据类型转换函数---- CAST() 函数语法如下 CAST() (<expression> AS <data\_ type>[ length ]) CONVERT() 函数语法如下 CONVERT() (<data\_ type>[ length ], <expression> [, style])    select cast(100+99 as char) convert(varchar(12), getdate()) 运行结果如下 ------------------------------ ------------ 199 Jan 15 2000    ----日期函数---- DAY() --函数返回date\_expression 中的日期值 MONTH() --函数返回date\_expression 中的月份值 YEAR() --函数返回date\_expression 中的年份值 DATEADD(<datepart> ,<number> ,<date>)  --函数返回指定日期date 加上指定的额外日期间隔number 产生的新日期 DATEDIFF(<datepart> ,<number> ,<date>) --函数返回两个指定日期在datepart 方面的不同之处 DATENAME(<datepart> , <date>) --函数以字符串的形式返回日期的指定部分 DATEPART(<datepart> , <date>) --函数以整数值的形式返回日期的指定部分 GETDATE() --函数以DATETIME 的缺省格式返回系统当前的日期和时间    ----系统函数---- APP\_NAME() --函数返回当前执行的应用程序的名称 COALESCE() --函数返回众多表达式中第一个非NULL 表达式的值 COL\_LENGTH(<'table\_name'>, <'column\_name'>) --函数返回表中指定字段的长度值 COL\_NAME(<table\_id>, <column\_id>) --函数返回表中指定字段的名称即列名 DATALENGTH() --函数返回数据表达式的数据的实际长度 DB\_ID(['database\_name']) --函数返回数据库的编号 DB\_NAME(database\_id) --函数返回数据库的名称 HOST\_ID() --函数返回服务器端计算机的名称 HOST\_NAME() --函数返回服务器端计算机的名称 IDENTITY(<data\_type>[, seed increment]) [AS column\_name]) --IDENTITY() 函数只在SELECT INTO 语句中使用用于插入一个identity column列到新表中 /\*select identity(int, 1, 1) as column\_name into newtable from oldtable\*/ ISDATE() --函数判断所给定的表达式是否为合理日期 ISNULL(<check\_expression>, <replacement\_value>) --函数将表达式中的NULL 值用指定值替换 ISNUMERIC() --函数判断所给定的表达式是否为合理的数值 NEWID() --函数返回一个UNIQUEIDENTIFIER 类型的数值 NULLIF(<expression1>, <expression2>) --NULLIF 函数在expression1 与expression2 相等时返回NULL 值若不相等时则返回expression1 的值        sql中的保留字    action add aggregate all  alter after and as  asc avg avg\_row\_length auto\_increment  between bigint bit binary  blob bool both by  cascade case char character  change check checksum column  columns comment constraint create  cross current\_date current\_time current\_timestamp  data database databases date  datetime day day\_hour day\_minute  day\_second dayofmonth dayofweek dayofyear  dec decimal default delayed  delay\_key\_write delete desc describe  distinct distinctrow double drop  end else escape escaped  enclosed enum explain exists  fields file first float  float4 float8 flush foreign  from for full function  global grant grants group  having heap high\_priority hour  hour\_minute hour\_second hosts identified  ignore in index infile  inner insert insert\_id int  integer interval int1 int2  int3 int4 int8 into  if is isam join  key keys kill last\_insert\_id  leading left length like  lines limit load local  lock logs long longblob  longtext low\_priority max max\_rows  match mediumblob mediumtext mediumint  middleint min\_rows minute minute\_second  modify month monthname myisam  natural numeric no not  null on optimize option  optionally or order outer  outfile pack\_keys partial password  precision primary procedure process  processlist privileges read real  references reload regexp rename  replace restrict returns revoke  rlike row rows second  select set show shutdown  smallint soname sql\_big\_tables sql\_big\_selects  sql\_low\_priority\_updates sql\_log\_off sql\_log\_update sql\_select\_limit  sql\_small\_result sql\_big\_result sql\_warnings straight\_join  starting status string table  tables temporary terminated text  then time timestamp tinyblob  tinytext tinyint trailing to  type use using unique  unlock unsigned update usage  values varchar variables varying  varbinary with write when  where year year\_month zerofill  SQL语句导入导出大全  /\*\*\*\*\*\*\* 导出到excel EXEC master..xp\_cmdshell 'bcp SettleDB.dbo.shanghu out c:temp1.xls -c -q -S"GNETDATA/GNETDATA" -U"sa" -P""'  /\*\*\*\*\*\*\*\*\*\*\* 导入Excel SELECT \*  FROM OpenDataSource( 'Microsoft.Jet.OLEDB.4.0', 'Data Source="c:test.xls";User ID=Admin;Password=;Extended properties=Excel 5.0')...xactions  /\*动态文件名 declare @fn varchar(20),@s varchar(1000) set @fn = 'c:test.xls' set @s ='''Microsoft.Jet.OLEDB.4.0'', ''Data Source="'+@fn+'";User ID=Admin;Password=;Extended properties=Excel 5.0''' set @s = 'SELECT \* FROM OpenDataSource ('+@s+')...sheet1$' exec(@s) \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*导出到Access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ insert into openrowset('Microsoft.Jet.OLEDB.4.0',  'x:A.mdb';'admin';'',A表) select \* from 数据库名..B表  /\*\*\*\*\*\*\*\*\*\*\*\*\*导入Access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ insert into B表 selet \* from openrowset('Microsoft.Jet.OLEDB.4.0',  'x:A.mdb';'admin';'',A表)  文件名为参数 declare @fname varchar(20) set @fname = 'd:test.mdb' exec('SELECT a.\* FROM opendatasource(''Microsoft.Jet.OLEDB.4.0'', '''+@fname+''';''admin'';'''', topics) as a ')  SELECT \*  FROM OpenDataSource( 'Microsoft.Jet.OLEDB.4.0', 'Data Source="f:northwind.mdb";Jet OLEDB database Password=123;User ID=Admin;Password=;')...产品  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 导入 xml　文件  DECLARE @idoc int DECLARE @doc varchar(1000) --sample XML document SET @doc ='    Customer was very satisfied     Important Happy Customer.     ' -- Create an internal representation of the XML document. EXEC sp\_xml\_preparedocument @idoc OUTPUT, @doc  -- Execute a SELECT statement using OPENXML rowset provider. SELECT \* FROM OPENXML (@idoc, '/root/Customer/Order', 1) WITH (oid char(5),  amount float,  comment ntext 'text()') EXEC sp\_xml\_removedocument @idoc    ???????  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Excel导到Txt\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 想用 select \* into opendatasource(...) from opendatasource(...) 实现将一个Excel文件内容导入到一个文本文件  假设Excel中有两列，第一列为姓名，第二列为很行帐号(16位) 且银行帐号导出到文本文件后分两部分，前8位和后8位分开。   邹健： 如果要用你上面的语句插入的话,文本文件必须存在,而且有一行:姓名,银行账号1,银行账号2 然后就可以用下面的语句进行插入 注意文件名和目录根据你的实际情况进行修改.  insert into opendatasource('MICROSOFT.JET.OLEDB.4.0' ,'Text;HDR=Yes;DATABASE=C:' )...[aa#txt] --,aa#txt) --\*/ select 姓名,银行账号1=left(银行账号,8),银行账号2=right(银行账号,8)  from  opendatasource('MICROSOFT.JET.OLEDB.4.0' ,'Excel 5.0;HDR=YES;IMEX=2;DATABASE=c:a.xls' --,Sheet1$) )...[Sheet1$]    如果你想直接插入并生成文本文件,就要用bcp  declare @sql varchar(8000),@tbname varchar(50)  --首先将excel表内容导入到一个全局临时表 select @tbname='[##temp'+cast(newid() as varchar(40))+']' ,@sql='select 姓名,银行账号1=left(银行账号,8),银行账号2=right(银行账号,8)  into '+@tbname+' from  opendatasource(''MICROSOFT.JET.OLEDB.4.0'' ,''Excel 5.0;HDR=YES;IMEX=2;DATABASE=c:a.xls'' )...[Sheet1$]' exec(@sql)  --然后用bcp从全局临时表导出到文本文件 set @sql='bcp "'+@tbname+'" out "c:aa.txt" /S"(local)" /P"" /c' exec master..xp\_cmdshell @sql  --删除临时表 exec('drop table '+@tbname)  /\*\*\*\*\*\*\*\*\*\*\*\*\*导出到Access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ insert into openrowset('Microsoft.Jet.OLEDB.4.0',  'x:A.mdb';'admin';'',A表) select \* from 数据库名..B表  /\*\*\*\*\*\*\*\*\*\*\*\*\*导入Access\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ insert into B表 selet \* from openrowset('Microsoft.Jet.OLEDB.4.0',  'x:A.mdb';'admin';'',A表)  文件名为参数 declare @fname varchar(20) set @fname = 'd:test.mdb' exec('SELECT a.\* FROM opendatasource(''Microsoft.Jet.OLEDB.4.0'', '''+@fname+''';''admin'';'''', topics) as a ')  SELECT \*  FROM OpenDataSource( 'Microsoft.Jet.OLEDB.4.0', 'Data Source="f:northwind.mdb";Jet OLEDB database Password=123;User ID=Admin;Password=;')...产品  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 导入 xml　文件  DECLARE @idoc int DECLARE @doc varchar(1000) --sample XML document SET @doc ='    Customer was very satisfied     Important Happy Customer.     ' -- Create an internal representation of the XML document. EXEC sp\_xml\_preparedocument @idoc OUTPUT, @doc  -- Execute a SELECT statement using OPENXML rowset provider. SELECT \* FROM OPENXML (@idoc, '/root/Customer/Order', 1) WITH (oid char(5),  amount float,  comment ntext 'text()') EXEC sp\_xml\_removedocument @idoc    ???????  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Excel导到Txt\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 想用 select \* into opendatasource(...) from opendatasource(...) 实现将一个Excel文件内容导入到一个文本文件  假设Excel中有两列，第一列为姓名，第二列为很行帐号(16位) 且银行帐号导出到文本文件后分两部分，前8位和后8位分开。   邹健： 如果要用你上面的语句插入的话,文本文件必须存在,而且有一行:姓名,银行账号1,银行账号2 然后就可以用下面的语句进行插入 注意文件名和目录根据你的实际情况进行修改.  insert into opendatasource('MICROSOFT.JET.OLEDB.4.0' ,'Text;HDR=Yes;DATABASE=C:' )...[aa#txt] --,aa#txt) --\*/ select 姓名,银行账号1=left(银行账号,8),银行账号2=right(银行账号,8)  from  opendatasource('MICROSOFT.JET.OLEDB.4.0' ,'Excel 5.0;HDR=YES;IMEX=2;DATABASE=c:a.xls' --,Sheet1$) )...[Sheet1$]    如果你想直接插入并生成文本文件,就要用bcp  declare @sql varchar(8000),@tbname varchar(50)  --首先将excel表内容导入到一个全局临时表 select @tbname='[##temp'+cast(newid() as varchar(40))+']' ,@sql='select 姓名,银行账号1=left(银行账号,8),银行账号2=right(银行账号,8)  into '+@tbname+' from  opendatasource(''MICROSOFT.JET.OLEDB.4.0'' ,''Excel 5.0;HDR=YES;IMEX=2;DATABASE=c:a.xls'' )...[Sheet1$]' exec(@sql)  --然后用bcp从全局临时表导出到文本文件 set @sql='bcp "'+@tbname+'" out "c:aa.txt" /S"(local)" /P"" /c' exec master..xp\_cmdshell @sql  --删除临时表 exec('drop table '+@tbname)  SELECT cast(cast(科目编号 as numeric(10,2)) as nvarchar(255))+'　' 转换后的别名 FROM OpenDataSource( 'Microsoft.Jet.OLEDB.4.0', 'Data Source="c:test.xls";User ID=Admin;Password=;Extended properties=Excel 5.0')...xactions  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* EXCEL导到远程SQL insert OPENDATASOURCE( 'SQLOLEDB', 'Data Source=远程ip;User ID=sa;Password=密码' ).库名.dbo.表名 (列名1,列名2) SELECT 列名1,列名2 FROM OpenDataSource( 'Microsoft.Jet.OLEDB.4.0', 'Data Source="c:test.xls";User ID=Admin;Password=;Extended properties=Excel 5.0')...xactions   /\*\* 导入文本文件 EXEC master..xp\_cmdshell 'bcp dbname..tablename in c:DT.txt -c -Sservername -Usa -Ppassword'  /\*\* 导出文本文件 EXEC master..xp\_cmdshell 'bcp dbname..tablename out c:DT.txt -c -Sservername -Usa -Ppassword' 或 EXEC master..xp\_cmdshell 'bcp "Select \* from dbname..tablename" queryout c:DT.txt -c -Sservername -Usa -Ppassword'  导出到TXT文本，用逗号分开 exec master..xp\_cmdshell 'bcp "库名..表名" out "d:tt.txt" -c -t ,-U sa -P password'   BULK INSERT 库名..表名 FROM 'c:test.txt' WITH ( FIELDTERMINATOR = ';', ROWTERMINATOR = 'n' )   --/\* dBase IV文件 select \* from  OPENROWSET('MICROSOFT.JET.OLEDB.4.0' ,'dBase IV;HDR=NO;IMEX=2;DATABASE=C:','select \* from [客户资料4.dbf]') --\*/  --/\* dBase III文件 select \* from  OPENROWSET('MICROSOFT.JET.OLEDB.4.0' ,'dBase III;HDR=NO;IMEX=2;DATABASE=C:','select \* from [客户资料3.dbf]') --\*/  --/\* FoxPro 数据库 select \* from openrowset('MSDASQL', 'Driver=Microsoft Visual FoxPro Driver;SourceType=DBF;SourceDB=c:', 'select \* from [aa.DBF]') --\*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*导入DBF文件\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ select \* from openrowset('MSDASQL', 'Driver=Microsoft Visual FoxPro Driver; SourceDB=e:VFP98data; SourceType=DBF', 'select \* from customer where country != "USA" order by country') go /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 导出到DBF \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ 如果要导出数据到已经生成结构(即现存的)FOXPRO表中,可以直接用下面的SQL语句  insert into openrowset('MSDASQL', 'Driver=Microsoft Visual FoxPro Driver;SourceType=DBF;SourceDB=c:', 'select \* from [aa.DBF]') select \* from 表  说明: SourceDB=c: 指定foxpro表所在的文件夹 aa.DBF 指定foxpro表的文件名.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*导整个数据库\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  用bcp实现的存储过程   /\* 实现数据导入/导出的存储过程 根据不同的参数,可以实现导入/导出整个数据库/单个表 调用示例: --导出调用示例 ----导出单个表 exec file2table 'zj','','','xzkh\_sa..地区资料','c:zj.txt',1 ----导出整个数据库 exec file2table 'zj','','','xzkh\_sa','C:docman',1  --导入调用示例 ----导入单个表 exec file2table 'zj','','','xzkh\_sa..地区资料','c:zj.txt',0 ----导入整个数据库 exec file2table 'zj','','','xzkh\_sa','C:docman',0  \*/ if exists(select 1 from sysobjects where name='File2Table' and objectproperty(id,'IsProcedure')=1) drop procedure File2Table go create procedure File2Table @servername varchar(200) --服务器名 ,@username varchar(200) --用户名,如果用NT验证方式,则为空'' ,@password varchar(200) --密码 ,@tbname varchar(500) --数据库.dbo.表名,如果不指定:.dbo.表名,则导出数据库的所有用户表 ,@filename varchar(1000) --导入/导出路径/文件名,如果@tbname参数指明是导出整个数据库,则这个参数是文件存放路径,文件名自动用表名.txt ,@isout bit --1为导出,0为导入 as declare @sql varchar(8000)  if @tbname like '%.%.%' --如果指定了表名,则直接导出单个表 begin set @sql='bcp '+@tbname +case when @isout=1 then ' out ' else ' in ' end +' "'+@filename+'" /w' +' /S '+@servername +case when isnull(@username,'')='' then '' else ' /U '+@username end +' /P '+isnull(@password,'') exec master..xp\_cmdshell @sql end else begin --导出整个数据库,定义游标,取出所有的用户表 declare @m\_tbname varchar(250) if right(@filename,1)<>'' set @filename=@filename+''  set @m\_tbname='declare #tb cursor for select name from '+@tbname+'..sysobjects where xtype=''U''' exec(@m\_tbname) open #tb fetch next from #tb into @m\_tbname while @@fetch\_status=0 begin set @sql='bcp '+@tbname+'..'+@m\_tbname +case when @isout=1 then ' out ' else ' in ' end +' "'+@filename+@m\_tbname+'.txt " /w' +' /S '+@servername +case when isnull(@username,'')='' then '' else ' /U '+@username end +' /P '+isnull(@password,'') exec master..xp\_cmdshell @sql fetch next from #tb into @m\_tbname end close #tb deallocate #tb  end go   /\*\*\*\*\*\*\*\*\*\*\*\*\* Oracle \*\*\*\*\*\*\*\*\*\*\*\*\*\*/ EXEC sp\_addlinkedserver 'OracleSvr',  'Oracle 7.3',  'MSDAORA',  'ORCLDB' GO  delete from openquery(mailser,'select \* from yulin')  select \* from openquery(mailser,'select \* from yulin')  update openquery(mailser,'select \* from yulin where id=15')set disorder=555,catago=888  insert into openquery(mailser,'select disorder,catago from yulin')values(333,777)    补充：  对于用bcp导出,是没有字段名的.  用openrowset导出,需要事先建好表.  用openrowset导入,除ACCESS及EXCEL外,均不支持非本机数据导入 |