## [数据库](#目录)

### 概述



### MySQL

开源，体积小速度快，Web用；

### SQL Server 2008 R2

#### 安装

html；

#### 基础

三大范式：1，字段拆成原子；2，每一记录只和主键相关；3，是直接相关，不是间接；

08系统的体系结构：4种服务

数据库引擎、Analysis Services、Reporting Services、Integration Services；

数据库引擎（核心服务）：对数据的存储、处理和管理；

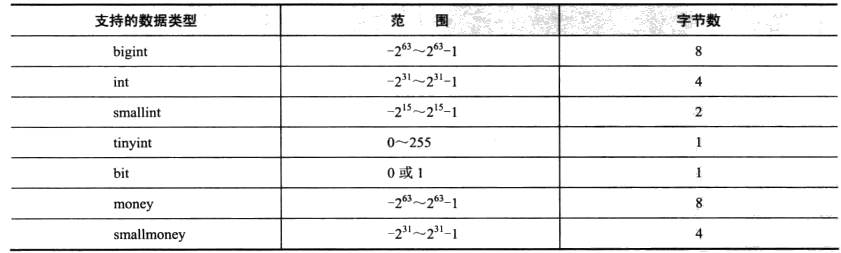
其它略；

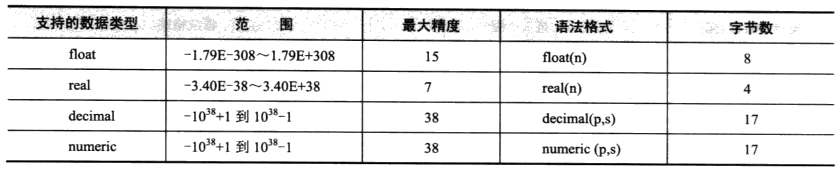
SQL Server Management Studio

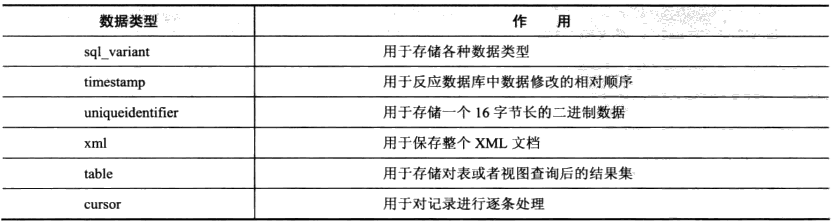
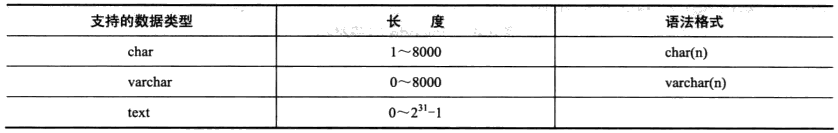
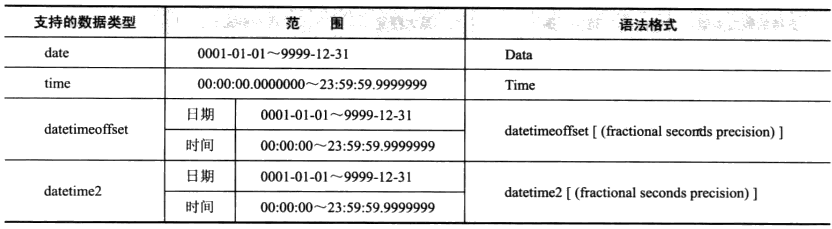
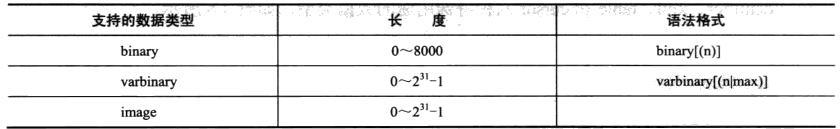
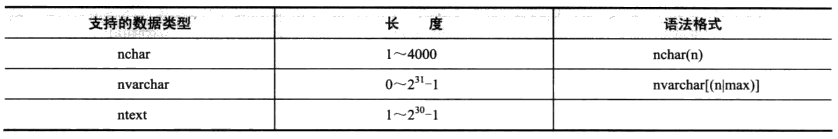
4个系统db；

主数据文件.mdf，辅助数据文件.mdf（可无）；事物日志文件.ldf；放在不同路径下；

#### 数据类型







#### T-SQL

Microsoft在SQL Server中使用的；Transact-SQL；SQL是有国际标准的，T-SQL的扩展包括自己的数据类型、表达式和关键字等；

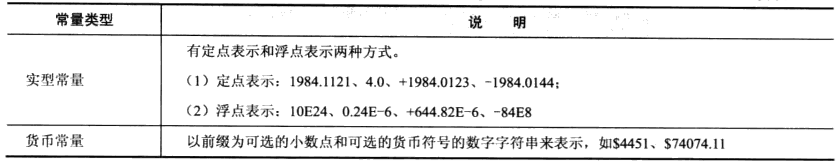
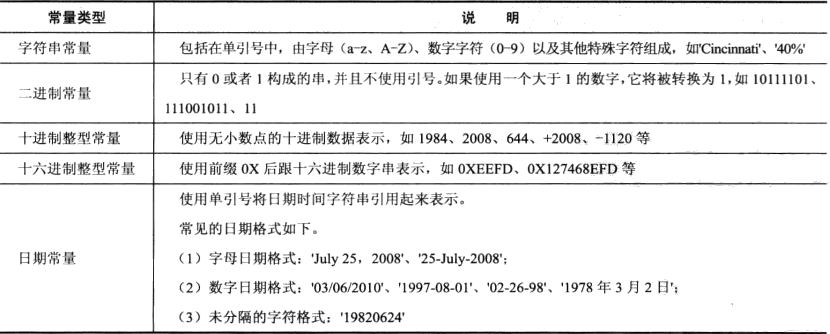
三类：

DDL data definition Language : CREATE、ALTER、DROP；

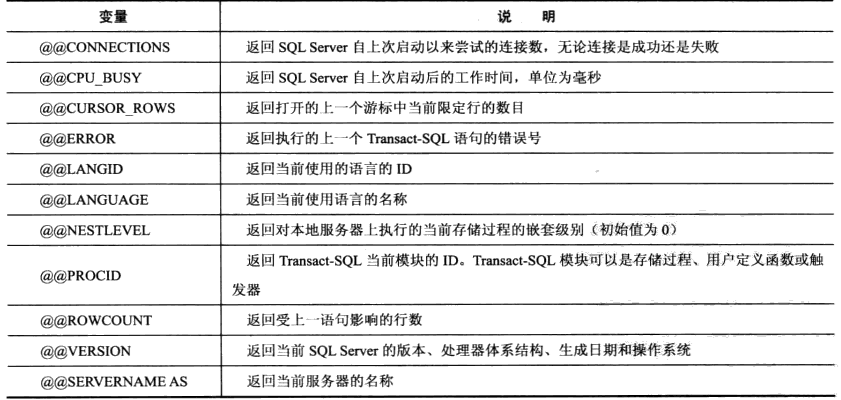
DDM Manipulation : SELECT、INSERT、UPDATE、DELETE；

DCL Control：sysadmin、dbcreator、db\_owner或db\_securityadmin等角色有权限；GRANT、REVOKE、DENY

#### 常量



#### 系统全局变量



#### 定义变量

DECLARE @变量名 变量类型

不能声明text、ntext、image；

#### 赋值变量

SET @变量名 = 值

#### 运算符

+ - \* / %

=、<、>、 !=、 返回Boolean类型：TRUE,FALSE,UNKNOWN

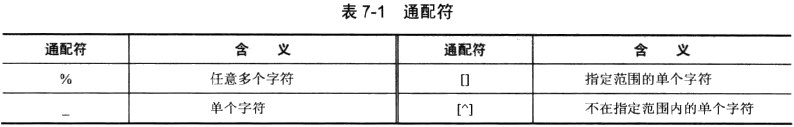
位：&、|、^、~

逻辑



NOT 可以配合：NOT IN、NOT LIKE、NOT BETWEEN…

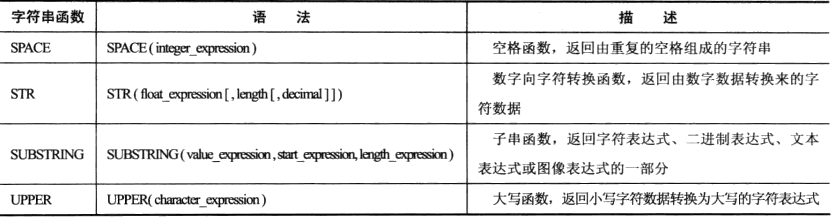
LIKE通配符：

IN: WHERE Author IN(‘a’, ‘b’, ‘c’) 列表中不允许有NULL值

IS NULL也是逻辑操作

#### 函数

聚合函数，与GROUP BY、HAVING子句一起使用；除了COUNT，都会忽略空值；



SQL Server 2008 中比较字符串不区分大小写！

charindex(expr1, expr2, [start]) expr1在expr2中开始位置(不是索引值)

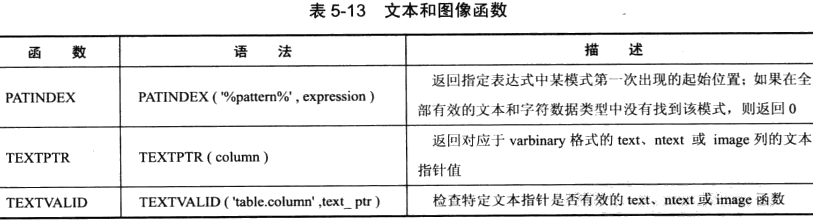
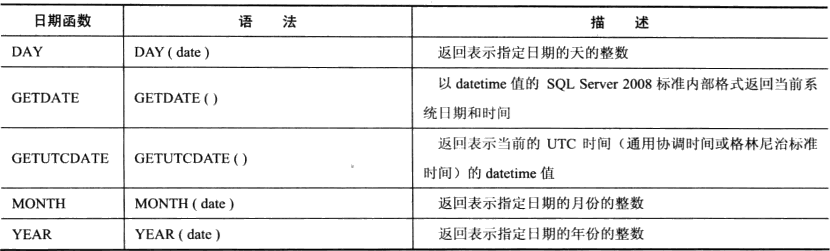
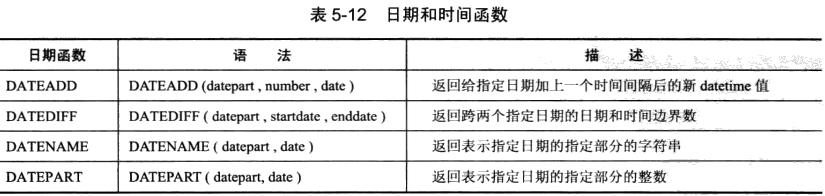
如: set @stagePos1 = charindex(',',@stagelist)

set @tmp = convert(int,substring(@upgradelist, @stagePos1 + 1, @stagePos2 - @stagePos1 - 1)) 开始位置，长度

类型转换：隐式

显示： CAST(表达式 AS 类型[(长度)])

CONVERT(类型[(length)], 表达式[, style])



##### 时间

sql ：Convert(varchar(10),getdate(),112) 变成 ‘20170210’

sql 前2天，dateadd(day,-2,getdate()) 20170208

#### 自定义函数

可以接受多个输入参数；返回值可以是数值、表；但不支持输出参数；

CREATE FUNCTION 函数名

( [ { @参数名 参数类型[ = default] } [,...n] ] )

RETURNS 返回值类型

[WITH ENCRYPTION]

[AS]

BEGIN

函数体

RETURN 表达式

END

返回表类型

CREATE FUNCTION 函数名

( [ { @参数名 参数类型[ = default] } [,...n] ] )

RETURNS TABLE

[WITH ENCRYPTION]

[AS]

RETURN (SELECT 语句)

#### 控制语句

BEGIN...END T-SQL语句块 下面提到的“语句块|SQL语句”都要用到；

IF bool表达式

语句块|SQL语句

[ELSE ...]

简单CASE语句：匹配就返回

CASE 表达式

WHEN 与表达式比较的表达式或值 THEN 返回的表达式

[,...n]

[ELSE 返回的表达式]

END

注意它是有返回的！ 所以可以这样用 SELECT CASET...

搜索CASE语句：找到第一个TRUE就返回

CASE

WHEN bool表达式 THEN 返回的表达式

[,...n]

[ELSE 返回的表达式]

END

WHILE bool表达式

语句块|SQL语句

[BREAK]

[CONTINUE]

#### 创建数据库

CREATE DATABASE [数据库名]

[ ON PRIMARY ( 主数据文件.mdf，PRIMARY是主数据文件

NAME = 文件名, N’db\_books’

FILENAME = 文件路径, N’E:\db\_books.mdf’

[SIZE = 文件初始大小,] 10240KB(10M)

[MAXSIZE = 文件最大容量,] 102400KB

[FILEGROWTH = 文件增量] )] 10240KB 或者写 10%

[, ...n] 辅助数据文件(.mdf有逗号)

[ LOG ON ( 事务日志文件.ldf

NAME = 文件名,

FILENAME = 文件路径,

[SIZE = 文件初始大小,]

[MAXSIZE = 文件最大容量,]

[FILEGROWTH = 文件增量] )]

CREATE DATABASE [name]，使用默认设置创建db；

选择数据库

USE 数据库名

GO

删除数据库

DROP DATABASE 数据库名[, ...n]

#### 创建表

CREATE TABLE [ 数据库名.[ owner]. | owner] 表名

(

{ <列定义:{列名 数据类型}> |

<约束 [ 默认值] | [自动编号 IDENTITY[ (seed起始值, increment) ] ]> |

列名 AS 计算列的表达式 } [, ...n]

| 或上的意思， BookNumber VARCHAR(10) NOT NULL PRIMARY KEY,

创建主键更灵活的方式： PRIMARY KEY(列名)，好多都可以这样用

创建自动编号：ID INT IDENTITY(1,1) NOT NULL,

创建外键： FOREIGN KEY REFERENCES(哪张表哪个列)

创建默认值： DEFAULT(值)

CHECK约束：CHECK(条件)

唯一性约束：UNIQUE，该列不允许重复

)

[ ON {filegroup | DEFAULT} ] 数据表所存储的文件组

[ TEXTIMAGE\_ON { filegroup | DEFAULT }] 数据表的Text和Image文件所在的文件组

#### 添加/删除/修改列

ALTER TABLE 表名 ADD 列名 列属性(类型，约束等)

ALTER TABLE 表名 DROP COLUMN 列名

ALTER TABLE 表名

ALTER COLUMN 列名 列属性

#### 删除表

DROP TABLE 表名

#### 插入数据

INSERT [INTO] 表 [( 列\_list )] VALUES( 值\_list )

如果没有指定列，那值要和列一一对上，自动编号的不用；

如果遗落了，会用默认值，如果没有默认值就设为NULL，如果不能为NULL，报错；

将一个表插入另一个表

INSERT...SELECT

INSERT表必须存在，数据类型，顺序一致；

INSERT 表名 SELECT 列\_list FROM 表\_list WHERE 条件

#### 更新数据

UPDATE [TOP] 表名

SET { 列名 = 表达式 | DEFAULT | NULL } [, …n]

[WHERE 条件]

#### 指定行

TOP(代表行的表达式) [PERCENT][WITH TIES]

WITH TIES用于返回额外的数据行

SELECT TOP n \* FROM

n 如果是变量  就要拼接了

declare @str varchar(1000)

set @str = ''

set @str = 'select top '+cast(10 as varchar(10))+' \* into #temp\_table\_user\_loginFailed from t\_user\_loginFailed where userID=@userID order by loginTime desc'

select @str

#### 删除数据行

DELETE 表名 WHERE 条件

只删除行；注意参照完整性；没有WHERE就是清空表；

TRUNCATE TABLE[[database.]owner.] 表名 清空表；

区别：一个写日志，可以ROLLBACK，一个不写；

#### SELECT

SELECT [ALL | DISTINCT] 列\_list all默认，distinct去重

FROM 表名

[WHERE 条件]

[GROUP BY 表达式] 对查询结果分组

[HAVING 条件] 与GROUP BY配合，如筛选>2的图书分类

[ORDER BY 表达式 [ASC | DESC]]

WHERE 配合逻辑操作符

order 可以别名；多个排序逗号隔开，可以跟DESC；NULL值最大；

GROUP BY 表达式 [WITH ROLLUP | CUBE]

SELECT Author,COUNT(\*) AS ‘Number’ FROM BookInfo GROUP BY Author 它会GROUP BY，同时也COUNT了，这个神奇；

其实是这样的：我要显示Author和Number，这两个要对上，COUNT正好对上；

SELECT 返回一张虚拟的表！

#### 取别名

列/表名 [AS] 别名

#### 子查询

SELECT list FROM 表名 WHERE 表达式 IN | NOT IN (子查询结果select)

EXISTS | NOT EXISTS(子查询结果)

ID = ANY(子查询结果) 或 单值子查询结果

SELECT \* FROM UserInfo AS UI WHERE EXISTS( SELECT UserID FROM CardInfo AS CI WHERE UI.ID = CI.UserID) WHERE好像可以循环一样，这个神奇；

select userid, pass from t\_au\_userinfo where userid in (select cardid from t\_au\_userlist where userid=%d and parterid=%d)

#### 临时表

select roomid into #t1 from t\_chnl\_roominfo

select distinct \* from #t1

drop table #t1

#### 多表查询

内连接

SELECT select\_list FROM 表1 INNER JOIN(或JOIN) 表2 ON 连接条件

[WHERE 条件]

[ORDER BY 条件]

列名重复了可以加表前缀，UI,CI是两张表

SELECT UI.\*, CI.\* FROM CardInfo CI INNER JOIN UserInfo UI ON CI.UserID = UI.ID AND CI.CreateTime < ‘2011—6-01’ 别名可以先用，这个神奇；

外连接

如果数据表中一些行在其他表中不存在匹配行，内连接通常会忽略，而外连接至少要返回一个表的全部，因此有主从之分；

左外连接：左全匹配，右边填NULL

SELECT select\_list FROM t1 LEFT(RIGHT/FULL) OUTER JOIN t2 ON 条件

[WHERE 条件]

[ORDER BY 条件]

交叉连接

CROSS JOIN 返回所有行的可能组合

自连接

一个表的字段间有关系；

SELECT p1.UserName ‘父亲’, p2.UserName ‘儿子’

FROM People p1, People p2

WHERE p1.ID = p2.FatherID

联合，交，差查询

将两个或两个以上SELECT结果合并

SELECT list FROM t [WHERE 条件] { UNION [ALL包括重复] SELECT list FROM t [WHERE] } [ORDER BY]

列名在第一个查询中指定；行不匹配报错；

INTERSECT 交查询

EXCEPT 差查询

#### 索引

加快查询速度；

表的存储由两部分组成，表页面和索引页面；当检索数据时，系统先搜索索引页面，找到数据指针，再获取数据，对编写T-SQL没有任何影响；

将表和索引分别存储在不同的文件组中，会大大提高数据操作速度；

细节：《轻松学》P191

#### 存储过程

T-SQL语句集合

执行语法：

[[EXE[CUTE]]

{

[@返回状态=] { 过程名[;number] | @局部变量名}

[[@参数=] {value | @variable[OUTPUT] | [DEFAULT] } [, …n]

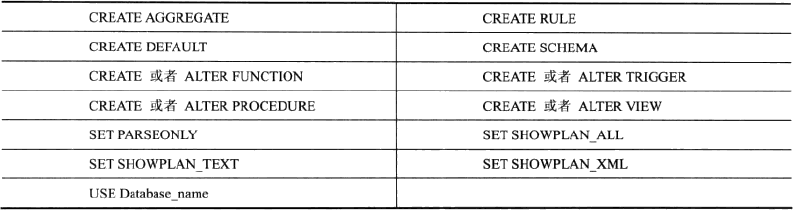
[WITH RECOMPILE]

}

number 同名过程的分组号

@参数 显示指定可以不按顺序给实参

自定义存储过程

下面的语句不能用：

CREATE PROC[EDURE] 过程名[;number]

[{@参数 类型} [VARYING] [=default] [OUTPUT]] [,… n]

[WITH { RECOMPILE | ENCRYPTION}]

[FOR REPLICATION]

AS

SQL语句

VARYING 游标使用

default 参数默认值

RECOMPILE、ENCRYPTION、FOR..高级

加密、临时、嵌套

CREATE PROC Proc\_FindBooks

@classify VARCHAR(50) = ‘unknown’ OUTPUT,

@author …

AS

BEGIN ..

END

#### 触发器

分类

DML：对表或视图的改动INSERT、UPDATE、DELETE；

DDL：CREATE、ALTER、DROP、GRANT、DENY、REVOKE等；

登录触发器

08为每个DML触发器创建：deleted和inserted表，系统维护，触发完删除；

创建

DML

CREATE TRIGGER 名

ON { 表｜视图 }

{

{ FOR | AFTER | INSTEAD OF } 触发时机

{ [DELETE], [INSERT],[UPDATE] }

AS

sql

}

INSTEAD OF :当进行INSERT、UPDATE、DELETE操作时，系统给触发器，可以定义更复杂的约束；

DDL

CREATE TRIGGER 名

ON { ALL SERVER | DATABASE } 作用域

WITH ENCRYPTION

{ FOR | AFTER | { event\_type }}

AS sql

Event\_type:

CREATE\_TABLE CREATE\_DATABASE

嵌套、递归触发器； P241

修改、启禁用、删除；

#### 事务

一组T-SQL命令，要么全部成功，要么撤销；

BEGIN TRANSACTION

COMMIT

ROLLBACK

BEGIN TRANSACTION

Sql

IF @ERROR = 0

BEGIN

COMMIT TRANSCATION

PRINT ‘成功’

END

ELSE

BEGIN

ROLLBACK TRANSCATION

END

数据锁

不同用户同时修改同一个数据；

#### 游标

操作查询结果的一种常用方式；

DECLARE 游标名 CURSOR [LOCAL | GLOBAL]

[FORWARD\_ONLY | SCROLL]

[static | keyset | dynamic | fast\_forward]

[READ\_ONLY | SCROLL\_LOCKS | OPTIMISTIC]

[TYPE\_WARNING]

FOR select语句

[FOR UPDATE [OF 列名[, …n]] 可更新的列

打开游标

OPEN{{ [GLOBAL] 游标名} ｜ 游标变量名 }

检索数据

FETCH[[ NEXT | PRIOR | FIRST | LAST | ABSOLUTE {n | @nvar} | RELATIVE {n | @nvar} ] FROM ]

{{ [GLOBAL] 游标名} | @游标变量名}

[INTO @变量名[, … n]]

@@FETCH\_STATUS 0 成功 -1失败 -2 行不存在

配合INTO 修改字段值，配合@@FETCH\_STATUS遍历；

关闭游标

CLOSE {{[GLOBAL] 游标名} | 游标变量名}

释放游标

DEALLOCATE {{[GLOBAL] 游标名} | 游标变量名}

#### 架构

对象的容器，名字空间；

### ODBC

ODBC-Open Database Connectivity，微软开放式数据互联。是一组访问和操作数据库的API。可以使用相同的API访问不同的数据库，前提是需要将数据库设置为ODBC数据源。只能访问关系型数据库。

配置ODBC数据库：

“控制面板（小图标显示）-管理工具-数据源32位（如果Access是32位的）-系统DSN-添加”，选择“Microsoft Access Driver(\*.mdb)”，输入数据源名称并找到数据库文件，确定。

如果没有找到驱动的话，详见《遇到的问题》。

在mfc中添加class CMedicSet : public CRecordset

打开类向导，发现没有CRecordset基类。

选择MFC ODBC Consumer，选择数据源，机器数据源（首先要配置ODBC数据库，sql server的配置类似）。

### ADO

ADO是一组动态链接库，在使用之前必须导入ADO并且初始化。

在stdafx.h中:

#import “c:\program files (或者x86)\common files\system\ado\msado15.dll” no\_namespace rename(“EOF”,”adoEOF”)

编译后系统生成msado15.tlh、msado15.tli两个C++头文件来定义ADO库，将其放在cpp目录(默认生成在Debug目录下)。为了避免冲突，将EOF改名adoEOF。

Mfc中初始化OLE/COM库环境比较简单，在InitInstance中 if(! AfxOleInit()){ return FALSE; }

ADO库包含三个基本接口:\_ConnectionPtr接口、\_CommandPtr接口和\_RecordsetPtr接口。

\_ConnectionPtr

使用它创建一个数据连接，然后使用其它对象执行数据输入输出操作。

\_CommandPtr

它提供了一种简单的方法来执行返回记录集的存储过程和SQL语句。

\_RecordsetPtr

它对记录集提供了更多的控制功能

**基本流程**

 (1)初始化COM库，引入ADO库定义文件

(2)用Connection对象连接数据库

(3)利用建立好的连接，通过Connection、Command对象执行SQL命令，或利用Recordset对象取得结果记

   录集进行查询、处理。

(4)使用完毕后关闭连接释放对象。

详见网页：用ADO操作数据库的方法步骤(转载) - 长寿梦的编程手记 - C++博客.html。

为了便于操作数据库，定义派生自CADORecordBinding的类，用于程序与数据库字段的交互(绑定)。

#include <icrsint.h>

Class CDevCodeRs : public CADORecordBinding

{

InstrumentManage <DataBinding.h>

}