本文档信息

文件状态： [ ] 草稿 [√] 正式发布 [] 正在修改	文件编号	
	文档名称	代码编写规范
	当前版本	**1.0**
	作者	
	完成日期	**2004-3-1**
审核人：		批准人：

历史版本信息

版本/状态	说明
1.0	第一次正式发布

目　　次

1 简介

编码中最重要的是要遵从相关开发工具的代码编写规范，养成良好的编程习惯，提高软件开发质量。

本规范由三大部分组成：

1 公共编码规范。公共编码规范主要介绍了所有开发工具共同遵守的编码规范。

2 详细编码规范。编码规范针对公司目前所用的开发工具（如 VC、VB、Delphi、PowerBuilder、Asp、Java 等），分别制定了对应的代码编码规范。

3 数据库编码规范。数据库编码规范包括数据库公共规范、MS SQL Server 和 Oracle 数据库的不同规范。

2 公共编码规范

公共编码规范总结了所有开发工具在编码过程中应该共同遵守的规范，包括基本规范和注释规范两部分。

2.1 基本规范

◇ 程序风格：程序结构清晰，简单易懂；单个函数或过程的程序行数（包括注释）不得超过 500 行、宽度不超过屏幕显示的宽度范围；一般不用递归模式；不建议使用 GOTO；循环、分支要简单：层次不得超过五层；

◇ 可读性：一定要注释清晰，利用缩进来显示程序的逻辑结构：根据不同开发工具利用空格（至少两个）或 Tab 字符来组织代码缩进结构；

◇ 代码简洁：减少不必要的代码量；程序目录中不得存在垃圾文件；数据库中不得存在垃圾表。

◇ 代码重用：提高函数或过程的共享，尽量简化编码，减少不必要的重复代码；

◇ 代码优化：注重执行效率高、力求代码精简，优化代码结构；

◇ 变量使用：变量使用遵循先声明后使用的原则，尽量使用局部变量，少用全局变量；

◇ 命名规范：所有对象的命名能准确清晰的代表当前对象的实际意义。对象命名可选用英文或拼音缩写，但在同一项目设计中必须遵循共同约定。

防止内存泄漏：申请内存，使用结束后，必须及时清理。

3 注释公共规范

代码一定要进行必要的注释，提高代码的可读性和可维护性。

3.1 基本规范

◇ 注释可以与语句在同一行，也可以在语句的上一行；

◇ 原则上使用中文注释；

◇ 不应出现冗余注释，用简练准确的语言说明代码；

◇ 禁止对简单明了的代码行简单翻译为注释，比如

```
错误注释例子：if student_sex = 0 then //如果 student_sex 等于 0
正确注释例子：if student_sex = 0 then //学生性别为“男”
```

◇ 注释规范包括文件头注释和代码修改处的注释两部分。

注释格式：

对于不同的开发工具有不同的注释符号，在下面的注释中，以 C 语言中的注释符号为代表说明。

3.2 文件头注释

文件头注释主要是用来描述当前文件特征的注释。

注释格式如下：

```
/*********************************************************************

名称：
功能描述：
创建日期：
作者：
==========================================================

----------------------修改记录----------------------------------

修改人:                                    修改时间:
修改原因:
----------------------------------------------------------------
==========================================================
备注：

*********************************************************************/
```

3.3 过程、函数注释

在每一个过程或函数的前面，注释如下：

```
/*********************************************************************

函数或过程名称：
输入参数：
输出参数：
功能描述：
创建日期：
作者：
==========================================================

----------------------修改记录----------------------------------

修改人:                                    修改时间:
修改原因:

*********************************************************************/
```

3.4 事件注释

在每一个事件的前面，注释如下：

```
/**************************************************************************

事件名称：
事件功能：
创建日期：
作者：
==========================================================

----------------------修改记录----------------------------------

修改人:                                    修改时间:
修改原因:
----------------------------------------------------------------
==========================================================
备注：

**************************************************************************/
```

3.5 代码修改注释

适用于版本封存以后的修改。根据以下不同情况来分别处理：

版本升级：版本升级的代码修改量都比较大，首先保留原版本（备份），然后在新版本代码中直接进行修改，在新版本中不保留源代码。

非升级型的大量修改维护：经过批准，对已经封存后的版本进行维护修改，修改量比较大时（比如一个函数 2/3 的代码都完全修改了）：首先修改者备份当前需要修改的源文件，然后在新文件中直接修改代码，不保留源代码，并且在新的代码段前加以下注释：

```
//修改人                                修改时间
```

非升级型的小量修改维护：经过批准，对已经封存后的版本进行维护修改，修改量比较小时（只是个别地方的修改）：修改者必须保留修改前的内容，不能自行删除或直接修改；并标识出修改、新增或删除的内容。修改者都必须在该项目中填写修改信息，首先必须在当前文件的文件头修改记录中加入修改信息，并且在代码修改处加上修改注释，注释根据不同的修改动作分以下三个方面：

◇ 新增代码行

在程序中新增若干代码行时，应该在新增代码行的前后加上注释行说明。

```
/**************************************************************************

----------------------修改记录----------------------------------

  修改人：                                  修改时间：
  修改说明（如何修改）：
----------------------------------------------------------------

**************************************************************************/

// 新增代码行
```

新增的代码行

// 修改结束

◇ 删除代码行

在程序中删除若干代码行时，应该在被删除的代码行前后用注释行说明。

/**

----------修改记录----------

修改人： 修改时间：
修改说明（如何修改）：

要删除的代码行

修改结束

**********/

◇ 修改代码行

在程序中修改代码行时，首先删除代码行，然后新增代码行。

/**

----------修改记录----------

修改人： 修改时间：
修改说明（如何修改）：

修改前的代码行：
修改结束

**********/

//修改后的代码行

修改后的代码行。。。。。

//修改结束

3.6 程序中的注释

在不同的代码段中第一次引用变量时，注释其含义；复杂处理过程说明；算法说明；

自定义的变量常量、类型、结构等应有其用途说明。

4 Delphi 代码规范

包括 Delphi 中的 Begin..End 配对和命名规范两部分。

4.1 Begin..End 配对

begin 语句和 end 语句在源程序中成对使用

Begin 子句应写在独立的一行。例如，下面第一种是错误的写法而第二种是正确的。

```
for I := 0 to 10 do begin  //错误，begin 同 for 在同一行
for I := 0 to 10 do
begin                      //正确，begin 出现在独立的一行
```

这个规则的例外是当 begin 子句的出现是作为一个 else 子句的一部分－参考例子：

```
if someStatement then
begin
  …
end
else begin
  someOtherStatement;
end;
```

end 语句永远出现在独立的一行。

当 begin 语句不是一个 else 子句的一部分时，相应的 end 语句永远缩进到与 begin 部分相对应的位置。

4.2 命名规范

命名应使用描述性的名字，而不是简写。这样其他开发人员更容易读懂你的程序，也帮助你自己记住该变量的用途。不要用保留字或关键词来命名。

主要包括过程和函数命名、变量命名、控件实例命名和文件命名四部分。

4.2.1 过程和函数命名

过程、函数的命名规则：

- ◇ 例程名应当有意义,以大写字母开始，且应在具有不同词义单词的第一个字母处用大写；例如：Procedure FormatHardDisk
- ◇ 设置输入参数值的例程名应当以 Set 为其前缀，获取数值的例程名应当以 Get 为其前缀；例如 SetNewPrice；GetTotalSales
- ◇ 例程的形参名称应当表达出它的用途，只要可能，形参的名称应以字母 A 为前缀；例如 SomeProc(Acountry, Astate, ACity)
- ◇ DELPHI 的 IDE 自动产生的事件句柄，名称和参数不可作改动。

4.2.2 变量命名

- ◇ 变量的名称应当能表达出它的用途。尽量不缩写，并使用名词作为变量名；比如 MasterSql；
- ◇ 循环控制变量可以是单个字母如 I、J、K，也可以使用具有含义的名称
- ◇ 布尔变量的名称必须能清楚的表示出 True 和 False 的意义
- ◇ 工程的全局变量应该以 G_作为前缀；
- ◇ 单元内的全局变量应该以 U_作为前缀；

例如：

全局变量：G_RateValue

4.2.3 控件实例命名

控件实例命名规则

◇ 控件实例命名规则：控件类型前缀 + 描述性名称（代表功能含义，见名知义）。如

edtName（输入名字的 Edit 框）；

◇ DELPHI 标准控件的类型前缀如表所示：

Standard Tab 页		Additional Tab 页	
控件类名称	前缀	控件类名称	前缀
TMainMenu	mm	TBitBtn	bbtn
TPopupMenu	pm	TSpeedButton	sb
TMainMenuItem	mmi	TMaskEdit	me
TLabel	lbl	TStringGrid	sg
TEdit	edt	TDrawGrid	dg
TMemo	mem	TImage	img
TButton	btn	TShape	shp
TCheckBox	cb	TBevel	Bvl
TRadioButton	rb	TScrollBox	Sbx
TListBox	lb	TCheckListbox	Clb
TComboBox	cb	TSplitter	Spl
TScrollBar	scb	TStaticText	Stx
TGroupBox	gb		
TRadioGroup	rg		
TPanel	pnl	TChart	Cht
TCommandList	cl		
Win32 Tab 页		Internet Tab 页	
控件类名称	前缀	控件类名称	前缀
TTabControl	tbc	TClientSocket	Csk
TPageControl	pgc	TServerSocket	Ssk
TImageList	il	TWebDispatcher	Wbd
TRichEdit	re	TPageProducer	Pp
TTrackBar	tbr	TQueryTableProducer	Tp
TProgressBar	prb	TDataSetTableProduce r	Dstp
TUpDown	ud	TNMDayTime	Nmdt
THotKey	hk	TNMEcho	Nec
TAnimate	ani	TNMFinger	Nf
TDateTimePicker	dtp	TNMFtp	nft p
TTreeView	tv	TNMHttp	Nhttp
TListView	lv	TNMMsg	NMsg
THeaderControl	hdr	TNMMSGServ	Nmsg
TStatusBar	stb	TNMNNTP	nntp
TToolBar	tlb	TNMPop3	Npop
TCoolBar	clb	TNMUUProcessor	Nuup
		TNMSMTP	Smtp
		TNMStrm	Nst
		TNMStrmServ	Nsts

Standard Tab 页		Additional Tab 页	
		TNMTime	Ntm
		TNMUdp	Nudp
		TPowerSock	Psk
		TNMGeneralServer	Ngs
		THtml	Html
		TNMUrl	url
		TsimpleMail	Sml
System Tab 页		Data Access Tab 页	
控件类名称	前缀	控件类名称	前缀
TTimer	tm	TDataSource	ds
TPaintBox	pb	TTable	Tbl
TMediaPlayer	mp	TQuery	Qry
TOleContainer	olec	TStoredProc	Sp
TDDEClientConv	ddcc	TDataBase	Db
TDDEClientItem	ddci	TSession	Ssn
TDDEServerConv	ddsc	TBatchMove	Bm
TDDEServerItem	ddsi	TUpdateSQL	usql
Data Controls Tab 页		Decision Cube Tab 页	
控件类名称	前缀	控件类名称	前缀
TDBGrid	dbg	TDecisionCube	Dcb
TDBNavigator	dbn	TDecisionQuery	Dcq
TDBText	dbt	TDecisionSource	Dcs
TDBEdit	dbe	TDecisionPivot	Dcp
TDBMemo	dbm	TDecisionGrid	Dcg
TDBImage	dbi	TDecisionGraph	dcgr
TDBListBox	dblb		
TDBComboBox	dbcb		
TDBCheckBox	dbch		
TDBRadioGroup	dbrg		
TDBLookupListBox	dbll		
TDBLookupComboBox	dblc		
TDBRichEdit	dbre		
TDBCtrlGrid	dbcg		
TDBChart	dbch		
Win31 Tab 页		Samples Tab 页	
控件类名称	前缀	控件类名称	前缀
TDBLookupList	dbll	TGauge	Gg
TDBLookupCombo	dblc	TColorGrid	Cg
TTabSet	ts	TSpinButton	Spb
TOutline	ol	TSpinEdit	Spe
TTabbedNoteBook	tnb	TDirectoryOutline	Dol
TNoteBook	nb	TCalendar	Cal

Standard Tab 页		Additional Tab 页	
THeader	hdr	TIBEventAlerter	ibea
TFileListBox	flb		
TDirectoryListBox	dlb		
TDriveComboBox	dcb		
TfilterComboBox	fcb		
ActiveX Tab 页		Midas Tab 页	
控件类名称	前缀	控件类名称	前缀
TChartFX	cfx	TDatasetProvider	Psp
TVSSpell	vsp	TClientDataSet	Cds
TF1Book	f1b	TDCOMConnection	dcom
TVTChart	vtc	TOleEnterpriseConnection	olee
Tgraph	grp	TSocketConnection	Sck
		TRemoteServer	Rms
		TmidasConnection	mid
ADO Tab 页		InterBase Tab 页	
控件类名称	前缀	控件类名称	前缀
ADOConnection	adocn	IBTable	ibt
ADOCommand	adocm	IBQuery	ibq
ADODataSet	adods	IBStoreProc	ibsp
ADOTable	adot	IBDatabase	ibd
ADOQuery	adoq	IBTransaction	ibts
ADOStoreProc	adosp	IBUpdateSQL	ibu
RDSConnection	rdsc	IBDataSet	ibds
		IBSQL	ibsql
		IBDatabaseInfo	ibdi
		IBSQLMonitor	ibsm
		IBEvents	ibe

4.2.4 文件命名

文件命名规范：

- ◇ 工程文件：工程文件应该赋予有意义的名字。可以取工程的全称组成单词的开始字符的大写，或者每个单词的缩写。也可以直接用单个单词命名；
- ◇ 窗体文件：窗体文件的名称必须能表达 Form 的用途，并且具有 frm 前缀；如 frmMain；
- ◇ 数据模块文件：Data Module 文件的名称必须能表达数据模块的用途，并且具有 dm 前缀；如 dmData
- ◇ 远程数据模块文件：Remote Data Module 文件的名称必须能表达远程数据模块的用途，并且具有 rdm 前缀；如 rdmData
- ◇ 单元文件：窗体及数据模块的单元文件名称应与相应的 Form 和数据模块名称相同。通用单元（如公共函数库、变量库、常量库等）的命名应尽量表达它的含义，并以 Lib 作为前缀。

5 PowerBuilder 命名规范

包括 PowerBuilder 的控件前缀约定、变量前缀约定、控件命名规则、变量命名规则、Pbl 文件命名规则五部分。

5.1 PB 控件命名前缀约定

窗体上不同对象控件的命名均以对象名称索引为前缀，PowerBuilder 对象控件前缀如表所示。

控 件 类 型	前 缀	控 件 类 型	前 缀
CommandButton	Cb_	PictureButton	Pb_
CheckBox	Cbx_	RadioButton	Rb_
StaticText	St_	Picture	P_
GroupBox	Gb_	Line	Ln_
Oval	Oval_	Rectangle	R_
RoundRectangle	Rr_	SingleLineEdit	Sle_
EditMask	Em_	MultiLineEdit	Mle_
RichTextEdit	Rte_	HScrollBar	Hsb_
VscrollBar	Vsb_	DropDownListBox	Ddlb_
DropDownPictureListBox	Ddplb_	ListBox	Lb_
PictureListBox	Plb_	ListView	Lv_
TreeView	Tv_	Graph	Gr_
DataWindow	Dw_	UserObject	Uo_
OleControl	Ole_	Tab	Tab_
TabPage	Tp_	DataStore	Ds_

5.2 PB 变量命名前缀约定

变 量 作 用 域		
作 用 域	前 缀	备 注
局部变量	L	Local
实例变量	I	Instance
全局变量	G	Global
共享变量	S	Share
变 量 类 型		
名 称	前 缀	备 注
Blob	Bl	大二进制对象
Boolean	B	布尔值
Char	C	字符
Date	Dt	日期
Datetime	Dtm	日期时间
Decimal	De	单精度数字
Double	D	双精度数字
Integer	I	短整型
Long	L	长整型

Real	R	单精度浮点数
String	S	字符串
Time	T	时间
Unsignedinteger	Ui	无符短整型
Unsignedlong	Ul	无符长整型
UserObject	Uo	自定义对象
Structure	Str	结构体
Any	An	任意类型变量

5.3 常用控件命名规则

该部分控件包括：命令按钮、选择框、单选按钮、静态文本、组合框、单行编辑器、多行编辑器等可以放置到窗体内的控件。名称格式为：

Prefix_Meaning（对象前缀+下画线+名称）

其中，Prefix 为前缀，必须严格遵循表 4-3-1 列的前缀约定（包括大小写）；Meaning 为代表实际控件的动作和意义，采用英文或英文缩写（如打印命令按钮名称定义为 cb_Print），每个英文字的第一个字母必须大写，禁止自然序号出现在名称内（例如 Cb_1）。常用命令按钮控件的 Meaning 约定如表所示，供参考使用。

控件动作或意义	Meaning 约定	控件动作或意义	Meaning 约定	控件动作或意义	Meaning 约定
增加、插入	Insert	复位	Reset	编辑	Edit
删除	Delete	检索	Retrieve	文件	File
保存、存盘	Save	上一步/个	Previous	设置	Setting
修改	Update	下一步/个	Next	其它、更多	More
确定、完成	OK	打开	Open	帮助	Help
取消	Cancel	新建	New	关于	About
退出	Exit	预览	Preview	暂停	Pause
返回、关闭	Close	复制	Copy	打印设置	PrinterSetup
打印	Print	另存为	SaveAs	发送	Send
运行	Run	停止	Stop	查询	Search
播放	Play	剪切	Cut	粘贴	Paste
终止	Abort	重试	Retry	忽略	Ignore

5.4 PB 变量命名规则

变量名称由 5 部分组成：

前缀+（A）+变量类型+下划线+变量名称

括号中的“A”代表数组，如果是数组加上“A”，反之不加。

变量名称应尽量使用英文进行标识，每个英文字的第一个字母必须是大写。前缀名必须严格遵循表 4-2 的约定（包括大小写）。如果使用拼音必须加上详细的注释；对于仅仅作为循环使用的变量，必须使用 Li_Cyc01、Li_Cyc02、……、Ll_Cyc01、Ll_Cyc02、……。

如表所示：

变量名称	变量说明含义说明
Li_Cyc01	局部-短整型循环变量 1
Ll_Cyc01	局部-长整型循环变量 1

Li_RecordCount	局部-数据数量
Ls_Name	局部-名称
LDtm_Date	局部-日期
LD_Money	局部-金额
LStr_Person	局部-结构体变量
LAs_Unit	局部-字符型数组-单位

对于不同作用域的变量，将“L”更换成“I”或“G”即可。
在定义变量时，必须在变量定义的前一行或定义变量的当前行进行变量的注释。例如：
公共对象命名规则
凡是在 PowerBuilder 的 PBL 文件中能够看到的对象称为公共对象。公共对象命名关键是避免重复，下表列出了不同类型对象的命名规则。

对象类型	命名规则	示例	备注
全局函数	函数缩写（f）+下划线+函数名称+下划线+（作者名字的缩写）	f_CentralWindow	作者缩写可无，以下同
自定义对象函数	函数缩写（uf）+下划线+函数名称	uf_Save	
窗体函数	函数缩写（wf）+下划线+函数名称	wf_Save	
窗口	窗体缩写（W）+下划线+窗体名称+下划线+（作者名字的缩写）	W_InputUnitInfo	
数据窗口	D+下划线+数据窗口名称+下划线+（作者名字的缩写）	D_InputUnitInfo	
数据管道	P+下划线+数据管道名称+下划线+（作者名字的缩写）	P_TestResult	
数据查询	Q+下划线+数据查询名称+下划线+（作者名字的缩写）	Q_UserName	
自定义对象	UO 缩写（UO）+下划线+UO 名称+下划线+（作者名字的缩写）	UO_SingleLineText	
结构	结构体缩写（Str）+下划线+结构体名称+下划线+（作者名字的缩写）	Str_UserInfo	
菜单	菜单缩写（m）+下划线+菜单名称+下划线+（作者名字的缩写）	m_selectdatetype	
应用	应用缩写（app）+下划线+应用名称	app_PowerManage	
工程项目	工程缩写（P）+下划线+工程名称	p_PowerManage	

5.5 Pbl 文件命名原则

任何一个子系统的 PBL 源文件有 3 部分组成：主应用文件、公共对象和分类对象文件。

主应用 PBL

命名规则是：App+下划线+应用名称缩写+下划线+V+版本号
例如：App_Ykbz_V3.Pbl 代表业扩报装应用文件，版本 3.0。

公共对象

公用的基础对象，由公司统一规划、开发、维护和发布，其它人员未经授权不得修改现有对象或增加、删除现有对象。所有公共对象均保留在公共 PBL 文件中。公共 PBL 名称规则是：
Base+对象类型+下划线+V+版本号
目前公司保留使用的公共 PBL 文件及内容如表所示：

文件名	意义
BaseWin_V3.PBL	公共窗口对象
BaseDW_V3.PBL	公共数据窗口对象
BaseFun_V3.PBL	公共函数对象
BaseUO_V3.PBL	公共用户定义对象
Base_Other_V3.PBL	公共其它对象（菜单）

分类对象

分类对象是除主应用文件和公共对象 Pbl 文件外，子系统或模块所需要的用于保存其他对象的文件。分类 PBL 文件名称由三部分组成：
应用名称缩写+下划线+类型名称缩写+（编号）
其中，应用名称缩写是以模块为单位而言，例如：Ykbz、 Yhda 分别代表代表业扩报装和用户档案缩写。类型名称缩写按照对象类型分类，均以对象类型名称的首字母作为缩写，例如数据窗口缩写为 D，则 Yhda_D.Pbl 代表用户档案子系统中的数据窗口文件。
在一个子系统中，如果一个类型的对象数量太多，导致查找和管理不方便时，需要将同一类型的对象保存多个 PBL 文件中，同一类型的 PBL 文件名按照序号区分，例如 Yhda_W1.PBL、Yhda_W2.PBL。原则上每个 pbl 文件大小不能超过 3M，或当某个 PBL 文件超过 3M 时，需要进行对象的分散保存。
在一个子系统中，菜单、结构、应用、工程等数量很少的对象，没有必要单独建立一个 PBL 文件，一般集中保存在主应用文件中。但是主应用文件的大小应以 3M 为限度，超过限度时，应将其中数量最多的对象单独建立 PBL 文件保存。

6 Asp 代码规范

在 Asp 中，主要包括 Asp 命名基本规范、JavaScript 命名规范、VbScript 命名规范。

6.1 Asp 命名基本规范

包括变量命名规范、函数与过程命名规范。

6.1.1 变量命名规范

变量类型前缀

如表所示

类型	前缀
Integer（整型）	int
Float（浮点型）	flt
Double（双精度型）	dbl
Object（对象引用变量）	obj
String（字符串型）	str
DateTime（日期型）	dtm
Boolean(逻辑型)	bln

◇ 常量以及全局变量必须全部使用大写字母；
◇ 常量必须使用 CONST_前缀；
◇ 全局变量必须使用 G_前缀；
◇ 变量名首字母必须小写；
◇ 变量名除第一个单词的首字母小写外，其他单词首字符必须大写。
◇ 变量名使用类型前缀+功能描述名称（必须采用有意义的单词或其缩写命名，由一个或多个单词组成，每个单词的第一个字母大写，其余的全部小写）
◇ 除了被广泛了解的单词以外，所有使用单词变量名必须在定义时给出注释；

举例：

```
var
  strAdName //用于表示 Administrator 帐户的名称
```

6.1.2 函数以及子过程命名规范

◇ 函数命名必须使用有意义的英文单词或其组合，函数命名第一个单词的首字母小写，后面每一个单词的首字母大写，并且能够体现函数的功能；

6.2 客户端 JavaScript 对象命名规范

各种页面对象如 text 输入框、按钮、下拉选择框在命名时必须使用以下对应前缀+功能描述性单词（每个单词的第一个字母大写，其余的全部小写）： 如 txtName(名称的输入框)；页面对象前缀如表所示：

对象	前缀
Text 输入框	txt
button 按钮	btn
select 下拉选择框	sel
option 项	opt
form 表单	frm
frame 框架	fra
hidden 表单项	hdn
div 标记	div
span 标记	span
对话框对象	dlg
窗口对象	win

6.3 VBScript 对象命名规范

各种对象如 Connection、Recordset、Command 在命名时必须使用以下对应前缀：

说明：包含在[]中的部分为可省略部分，Name 代表的名称每个英文单词首字母大写，其余的全部小写。如表所示：

对象	前缀	举例
Connection 对象	conn	conn[Name]：Name 为所连接数据库的服务器名字
Recordset 对象	rs	rs[Name]:Name 为自定义的同 rs 存储内容有关的英文单词组合
Command 对象	cmd	cmd[Name]:Name 为自定义的同 command 目的有关的英文单词组合
Parameter 对象	param	Param[Name]：Name 为功能意义英文单词组合
Field 对象	fld	fld[Name]：Name 为功能意义英文单词组合
Error 对象	err	err[Name]：Name 为功能意义英文单词组合

6.4 Asp 注释

Asp 的所有注释必须写在服务器脚本中。

提供注释样例如下：

```
<%
……
'定义变量
dim rs,sql
'创建记录集对象
set rs = server.createobject("adodb.recordset")
'查询某某表
sql = "select * from table"
'实例化记录集对象
rs.open sql,conn,1,1
.'关闭记录集对象
rs.close
'关闭数据库连接对象
conn.close
'释放记录集对象
set rs=nothing
'释放数据库连接对象
set conn=nothing
……
%>
```

7 VB 对象命名规范

包括变量和常量范围前缀、常量命名、变量命名、函数和过程命名、自定义类型命名、常用控件命名、文件命名七部分。

7.1 变量和常量范围前缀

变量和常量范围前缀如表所示：

范围	前缀	举例
全局	g	gstrUserName
模块级	m	mblnCalcInProgress
本地到过程	无	dblVelocity

7.2 常量命名

常量名的主体是大小写混合的，每个单词的首字母大写，也需要遵循与变量相同的规则。

7.3 变量命名

用下列前缀来指明一个变量的数据类型（对于简单的循环变量可以例外，如 m、n、i），如表所示：

数据类型	前缀	举例
Boolean	bln	blnFound
Byte	byt	bytRasterData
Collection	col	colWidgets
Currency	cur	curRevenue
DateTime	dtm	dtmStart
Double	dbl	dblTolerance
Error	err	errOrderNum
Integer	int	intQuantity
Long	lng	lngDistance
Object	obj	objCurrent
Single	sng	sngAverage
String	str	strFName
User-Define Type	udt	udtEmployee

7.4 函数和过程命名

变量或过程名的主体应该使用大小写混合形式，并且应该足够长以描述它的作用。而且，函数名应该以一个动词起首，如 InitNameArray 或 CloseDialog。

对于频繁使用的或长的项，推荐使用标准缩略语以使名称的长度合理化。一般来说，超过 32 个字符的变量名在 VGA 显示器上读起来就困难了。

当使用缩略语时，要确保它们在整个应用程序中的一致性。在一个工程中，如果一会儿使用 Cnt， 一会儿使用 Count，将导致不必要的混淆。

7.5 自定义类型命名

在一项有许多用户定义类型的大工程中，常常有必要给每种类型一个它自己的三个字符的前缀。如果这些前缀是以 "u" 开始的，那么当用一个用户定义类型来工作时，快速识别这些类型是很容易的。例如，ucli 可以被用来作为一个用户定义的客户类型变量的前缀。

7.6 常用控件命名

应该用一致的前缀来命名对象，使人们容易识别对象的类型。下面列出了 Visual Basic 支持的一些推荐使用的对象约定，如表所示。

控件类型	前缀	举例
3D Panel	Pnl	pnlGroup
ADO Data	Ado	adoBiblio
Animated button	Ani	aniMailBox
Check box	Chk	chkReadOnly
Combo box, drop-down list box	Cbo	cboEnglish
Command button	Cmd	cmdExit
Common dialog	Dlg	dlgFileOpen

Communications	Com	comFax
Control	Ctr	ctrCurrent
Data	Dat	datBiblio
Data-bound combo box	dbcbo	dbcboLanguage
Data-bound grid	dbgrd	dbgrdQueryResult
Data-bound list box	dblst	dblstJobType
Data combo	Dbc	dbcAuthor
Data grid	Dgd	dgdTitles
Data list	Dbl	dblPublisher
Data repeater	Drp	drpLocation
Date picker	Dtp	dtpPublished
Directory list box	Dir	dirSource
Drive list box	Drv	drvTarget
File list box	Fil	filSource
Flat scroll bar	Fsb	fsbMove
Form	Frm	frmEntry
Frame	Fra	fraLanguage
Gauge	Gau	gauStatus
Graph	Gra	graRevenue
Grid	Grd	grdPrices
Hierarchical flexgrid	flex	flexOrders
Horizontal scroll bar	Hsb	hsbVolume
Image	Img	imgIcon
Image combo	imgcbo	imgcboProduct
ImageList	Ils	ilsAllIcons
Label	Lbl	lblHelpMessage
Lightweight check box	lwchk	lwchkArchive
Lightweight combo box	lwcbo	lwcboGerman
Lightweight command button	lwcmd	lwcmdRemove
Lightweight frame	lwfra	lwfraSaveOptions
Lightweight horizontal scroll bar	lwhsb	lwhsbVolume
Lightweight list box	lwlst	lwlstCostCenters
Lightweight option button	lwopt	lwoptIncomeLevel
Lightweight text box	lwtxt	lwoptStreet
Lightweight vertical scroll bar	lwvsb	lwvsbYear
Line	Lin	linVertical
List box	Lst	lstPolicyCodes
ListView	Lvw	lvwHeadings
MAPI message	Mpm	mpmSentMessage
MAPI session	Mps	mpsSession
MCI	Mci	mciVideo

Menu	Mnu	mnuFileOpen
Month view	Mvw	mvwPeriod
MS Chart	Ch	chSalesbyRegion
MS Flex grid	Msg	msgClients
MS Tab	Mst	mstFirst
OLE container	Ole	oleWorksheet
Option button	Opt	optGender
Picture box	Pic	picVGA
Picture clip	Clp	clpToolbar
ProgressBar	Prg	prgLoadFile
Remote Data	Rd	rdTitles
RichTextBox	Rtf	rtfReport
Shape	Shp	shpCircle
Slider	Sld	sldScale
Spin	Spn	spnPages
StatusBar	Sta	staDateTime
SysInfo	Sys	sysMonitor
TabStrip	Tab	tabOptions
Text box	Txt	txtLastName
Timer	Tmr	tmrAlarm
Toolbar	Tlb	tlbActions
TreeView	Tre	treOrganization
UpDown	Upd	updDirection
Vertical scroll bar	Vsb	vsbRate

7.7 文件命名规范

◇ **文件名应描述其中代码的大致作用，但不易太长，一般由两、三个单词组合而成，**每个单词首字母应大写；

◇ 文件夹和文件命名绝对不可使用汉字。

8 VC 规范

8.1 注释规范

请遵循公共注释规范。

8.2 命名规范

8.2.1 文件命名规范

工程文件：在 VC 之中，工程名为最后可执行文件名，所以项目名最好以最终的可执行文件名一致。

其它文件：文件名的语义应该能概括表达本文件的功能。文件名用小写英文字母表达，严禁使用中文；对于几个单词组合表达的文件名，单词之间用下划线分开。

对于规范的 VC 派生类，尽量用 Class Wizard 生成文件格式，避免用手工制作头文件实现文件。在编写代码时最好与 MFC 这种风格一致。

8.2.2 函数、规程命名规范

名称反映功能。

api 函数的命名规则：

函数一律以 Api 开头；后面的命名以函数语义为基准，如创建函数名为 ApiCeatePoint()。

8.2.3 变量命名规范

变量尽量采用匈牙利命名法，同时结合 VC 的原则；一般情况下，变量的取名方式为：<scope_>范围前缀_ + <prefix_>类型前缀_ + <qualifier>限定词。

范围前缀:

如表所示

前缀	类型	例子
g_	全局作用域	g_Servers
m_	成员变量	m_pDoc,
l_	局部作用域	l_strName

特殊的类型命名,前缀表示：

类、接口如表所示：

类型	前缀	例子	备注
Class	cls	clsObject	表示类型本身不与范围前缀结合使用
Interface	I	IUnknown	

注：类名前缀改为 cls，对于非全局的类最好有语义表示其所属模块。类的实例命名与类名大致相同，只是类名语义表示类的通用含义，而类名表示此实例的具体语义。如类名 clsSketPoint。

类型前缀

常用的一般数据类型的前缀表示(这只是一部分)，如表

类型	前缀	内存规格描述	例子
Char	ch	8-bit character	chGrade
TCHAR	ch	16-bit character if _UNICODE is defined	chName
BOOL	b	Boolean value	bEnabled
Int	n	Integer (size dependent on operating system)	nLength
UINT	n	Unsigned value (size dependent on operating system)	nLength
WORD	w	16-bit unsigned value	wPos
LONG	l	32-bit signed integer	lOffset
DWORD	dw	32-bit unsigned integer	dwRange
*	p	Ambient memory model pointer	pDoc
FAR*	lp	Far pointer	lpDoc
LPSTR	lpsz	32-bit pointer to character string	lpszName
LPCSTR	lpsz	32-bit pointer to constant character string	lpszName

LPCTSTR	lpsz	32-bit pointer to constant character string if _UNICODE is defined	lpszName
handle	h	Handle to Windows object	hWnd
(*fn)()	lpfn	callbackFar pointer to CALLBACK function	lpfnAbort

常用 Windows 对象名称缩写：

这些名称缩写很多情况下可直接使用，直接作变量的名称。如表

Windows 对象	例子变量	MFC 类	例子对象
HWND	hWnd;	CWnd*	pWnd;
HDLG	hDlg;	CDialog*	pDlg;
HDC	hDC;	CDC*	pDC;
HGDIOBJ	hGdiObj;	CGdiObject*	pGdiObj;
HPEN	hPen;	CPen*	pPen;
HBRUSH	hBrush;	CBrush*	pBrush;
HFONT	hFont;	Cfont*	pFont;
HBITMAP	hBitmap;	CBitmap*	pBitmap;
HPALETTE	hPalette;	Cpalette*	pPalette;
HRGN	hRgn;	CRgn*	pRgn;
HMENU	hMenu;	Cmenu*	pMenu;
HWND	hCtl;	CStatic*	pStatic;
HWND	hCtl;	CButton*	pBtn;
HWND	hCtl;	Cedit*	pEdit;
HWND	hCtl;	CListBox*	pListBox;
HWND	hCtl;	CcomboBox*	pComboBox;

其它的宏定义

建议全用大写字母，用下划线分隔。 Visual C++常用宏定义命名见表：

前缀	符号类型	符号例子	范围
IDR_	标识多个资源共享的类型	IDR_MAINFRAME	1 to 0x6FFF
IDD_	对话框资源(Dialog)	IDD_SPELL_CHECK	1 to 0x6FFF
HIDD_	基于对话框的上下文帮助(Context Help)	HIDD_SPELL_CHECK	0x20001 to 0x26FF
IDB_	位图资源(Bitmap)	IDB_COMPANY_LOGO	1 to 0x6FFF
IDC_	光标资源(Cursor)	IDC_PENCIL	1 to 0x6FFF
IDI_	图标资源(Icon)	IDI_NOTEPAD	1 to 0x6FFF
ID_IDM_	工具栏或菜单栏的命令项	ID_TOOLS_SPELLING	0x8000 to 0xDFFF
HID_	命令上下文帮助(Command Help context)	HID_TOOLS_SPELLING	0x18000 to 0x1DFFF
IDP_	消息框提示文字资源	IDP_INVALID_PARTNO	8 to 0xDFFF
HIDP_	消息框上下文帮助(Message-box Help context)	HIDP_INVALID_PARTNO	0x30008 to 0x3DFFF

IDS_	字符串资源(String)	IDS_COPYRIGHT	1 to 0x7FFF
IDC_	对话框内的控制资源(Control)	IDC_RECALC	8 to 0xDFFF

9 Java 编程规范

9.1 命名规范

9.1.1 Class/Interface/Package/Method 命名规范

Class/Interface：必须以大写字母开头，如：

```
Class MainThread extends Thread
Interface LoadOption
```

Package：规定的 package 层次格式为 com.geasp.项目.具体模块

命名必须全部用小写字母，可以用”_”作为分割符，例如：

```
com.geasp.util.*;
com.geasp.beans.*;
com.geasp.shopping_center.client.*;
```

Operations(方法，函数)：必须以小写字母开头，采用大小写混合形式，并且应足够长以描述它的作用. 而且，Method 名应以一个动词起首，如

```
getUserRight()
exitProgram()
```

对于比较长的单词推荐使用缩略语以使名称的长度合理化. 当使用缩略语时，要确保它在整个使用程序中的一致性. 如果一会儿使用 Cnt，一会儿使用 Count，将导致不必要的混淆.

9.1.2 常量和变量命名约定

常量：必须全部大写，如：

```
public static final int MAXVALUE;
```

变量：以小写字母开头,加数据类型前缀，如：

```
String strName;
```

常用数据类型前缀请遵守如下规范：

类型	前缀	例子
常用数据类型		
Boolean / Boolean	.bl	blOK
char / Char	.ch	chRead
byte / Byte	.b	bWrite
int / Integer	.n	nLength
Short / Short	.s	sCount
long / Long	.l	lTotal
float / Float	.f	fSum
double / Double	.d	dSum
String	.str	. strNameArray
ArrayList	.al	.alData

Vector	.v	. vData
HashTable	. ht	. htRecord
HashMap	. hm	. hmRecord
Calender	. cal	, cal
与 SQL 相关的类		
Connection	. conn	. conn1
Statement	. stmt	. stmtQuery
PreparedStatement	. pstmt	. pstmtQuery
CallableStatement	. cstmt	. cstmtQuery
ResultSet	. rs	. rsQuery

加变量范围，成员变量加 m_，如:
String m_strName;
请使用标准的英文 Name，如果用缩略语也请统一使用

9.1.3 文件名约定

非 Source 相关文件，如 jar 文件(.jar),图档(.gif)，设置文件(.ini)，数据库文件(.sql)等文件名必须全部为小写. 单词间用”_”分隔.

9.2 Java 编码规范

9.2.1 三种 java 注释格式

Comment Type	Usage	Example
Documentation Use 文档格式	文档注释直接写在 interfaces, classes, member functions, fields 的前面 这种注释可以通过**javadoc** 的工具生成文档	/** Customer - A customer is any person or organization that we sell services and products to. @author S.W. Ambler */
C style Use C 语言风格	多行注释. 可以用来描述算法等多行的文字	/* This code was commented out by J.T. Kirk on Dec 9, 1997 because it was replaced by the preceding code. Delete it after two years if it is still not applicable. . . . (the source code) */
Single line Use	单行注释	// Apply a 5% discount to

单行注释		all invoices // Feb. of 1995.

对于单行注释要补充的是，尽量不要把注释写在一行的代码的后面. 否则会破坏程序的结构. 应该是写在程序代码的上方并且和代码左靠齐.

9.2.2 javadoc

javadoc 是一种文档产生工具，它可以方便的将源文件中的注释转成标准的 java 文档. 只要在写注释的时候按照一定标准来写就可以了. 下表是 javadoc 支持的一些标签. 我们会在自己的注释中应用 javadoc 的一些规范.

Tag 标签	Used for 使用对象	Purpose 目的
@author name	Interfaces, Classes,	标识作者 一个@author 对应一个作者
@deprecated	Interfaces, Classes, Member Functions	表示这个 API 已经不推荐使用,请使用其他类型代替
@exception name description	Member Functions	描述方法会抛出的异常. 一个标签对应一个异常 name 应该是异常类的全名
@param name description	Member Functions	用来描述参数，包括它的类型和使用方法
@return description	Member Functions	返回值说明，包括类型，和代表的意义.
@since	Interfaces, Classes, Member Functions	显示该项目什么时候开始存在的 例如 @since JDK 1.1
@see ClassName	Classes, Interfaces, Member Functions, Fields	建立一个超级连接到指定的 class 作为参考
@see ClassName#member functionName	Classes, Interfaces, Member Functions, Fields	建立一个超级连接到指定的某一个方法中去
@version text	Classes, Interfaces	版本号

9.2.3 Class/Interface 文件头注释约定

所有类,接口的开始都要有关于这个类(接口)的注释

```
/**
 * Title:        Pushclass
 * Description:  a session bean for push class
 * Copyright:    Copyright (c) 2000
 * Company:      GE.Corp
 * @author:      raogaohua
```

```
 * @version:       1.0
 */
```

方法注释规范

所有的 Method 的开始都应该有描述这段代码的功能的一段简明注释. 但是这种描述不应该包括具体执行过程，因为这常常是随时间而变的，可能会成为错误的注释.

```
/**
* Name:        compString
* Description:  找出一个 String 在一个 String[]中位置 index
* Author:  zhaoshouiang
* @param       strArray  String 数组
* @param     strFind      需要找的 String
* @return             >=0     :    找到, 返回 strDind 在 strArray 中的 index
*                -1   :    没找到
*/
public int compString(String[] strArray, String strFind)
```

在方法中的注释要求做到下列几点:

A.每一个重要变量的声明应该包括一个嵌入注释，来描述变量的使用.
B.变量、控件及 Method 的命名应该足够清楚, 使得只有复杂的执行细节才需要嵌入注释.
C.列举主要数据对象、Method、算法、数据库及系统需求. 一段描述算法的伪代码能会有所帮助。

10 数据库规范

数据库的规范包括一般规范、命名规范、注释规范和建模规范；该部分规范包含了 SqlServer 和 Oracle 数据库设计时应该共同遵守的规范。

10.1 一般规范

◇ 所有对象必须给以有意义的名称，能描述清楚当前对象的功能，达到见名知义的效果；

◇ 每一个数据库表都必须给出详细的功能说明；

◇ 数据库表的每一个字段，都必须给出详细的注释，包括意义、是否允许空值、初值、枚举值及其意义；

◇ 未经批准，任何人不得对已定义的库表结构进行增删和改变字段意义、以及不按其原义使用；

◇ 数据库表必须设置主键，而且主键中不允许包含日期、浮点类型的字段。

◇ 设计数据库时，尽量考虑到不同数据库规范之间的通用性，以方便将来转换数据库，比如在 SqlServer 数据库设计时，应尽量避免使用对 SqlServer 非常依赖的特性，比如自增字段、TEXT、LONG、TINYINT 等字段；

10.2 事务处理

在 SQL 脚本中，要进行事务的定义、提交、回卷，避免对某些数据的锁

定，注意事务的嵌套，应该避免，如果确实需要，要使用保存点来处理。另外通常情况下事务的隔离级应该使用 READCOMMITTED，不要随意改动。如果在数据处理时出现错误，应当返回错误信息，并向用户进行提示。

10.3 功能实现原则

在数据库系统中，数据一致性、完整性保证，系统功能、数据检索等实现的途径有视图、触发器、存储过程、永久数据库表等。究竟什么条件下采用什么方式实现所需要的功能，下面将针对数据库功能实现方式给出一般规定和注释要求。

10.4 对象使用原则

包括视图、存储过程、索引和触发器等对象的使用原则。

10.4.1 视图使用原则

视图是虚拟的数据库表，在使用时要遵循以下原则：

- 从一个或多个库表中查询部分数据项；
- 为简化查询，将复杂的检索或字查询通过视图实现；
- 提高数据的安全性，只将需要查看的数据信息显示给权限有限的人员；
- 视图中如果嵌套使用视图，级数不得超过 3 级；
- 由于视图中只能固定条件或没有条件，所以对于数据量较大或随时间的推移逐渐增多的库表，不宜使用视图；
- 除特殊需要，避免类似 Select * from [TableName] 而没有检索条件的视图；

10.4.2 存储过程原则

存储过程的建立如同其它语言形式的编程过程，适合采用模块化设计方法；当具体算法改变时，只需要修改需要存储过程即可，不需要修改其它语言的源程序。当和数据库频繁交换数据是通过存储过程可以提高运行速度。由于只有被授权的用户才能执行存储过程，所以存储过程有利于提高系统的安全性。

存储过程是宿主于服务器端的 SQL 脚本集合，在使用时要遵循以下原则：

存储过程必须检索数据库记录，甚至修改（执行 Insert、Delete、Update、Drop、Create 等操作）数据库信息。如果某项功能不需要和数据库打交道，则不得通过存储过程的方式实现。

实现数据库级别的某项具体功能，功能的实现是修改数据库信息或对指定数据库信息加工后将加工结果返回给调用者。

10.4.3 索引使用原则

根据索引的类型分别列出相应的原则。一般性原则是根据查询的条件，或经常按照什么需要进行查询，则建立和查询条件相关的索引。索引建立的合理可以提高检索速度，但会增加数据库存储空间、建立数据记录修改或插入记录时的时间。

建立族索引的字段必须是查询或检索语句的条件字句中用到的字段。

族索引原则

在尽可能少的列（字段）上建立族索引。

可能进行批量数据修改的字段，不适宜建立族索引。

包含大量非重复值的列，在进行单值（返回）查询的时候，群集索引应当更有效，所以一般情况下，默认的会在主键上建立集群索引，由于 sqlserver 上一个表只能够建立一个群集索引，通常建议不需要改动。而对于 oracle，机制有些不同，就普通的来说，位映射索

引是非常紧凑的，适合于取值范围比较小，很少改动，匹配比较标准的列进行索引，当使用一个或几个来一起解决查询问题时（可以是 or 查询，也可以是 and 查询），位映射索引实际上起了很大的作用，位映射索引适合于数据仓库应用。B-tree 索引适合于值重复比较少的列上，更新的成本比较低，适合联机事务处理应用。

10.4.4 触发器使用原则

作为一种特殊的存储过程，触发器通过数据的修改而触发执行，起作用是为确保数据的完整性和一致性不被破坏而创建，实现数据的完整约束。触发器的另一个作用用来备份关键的数据，例如电费、电价的修改，系统自动保存修改前的结果，实现对修改结果的稽查为恢复数据提供依据。避免通过触发器实现某些特定功能。开发人员可以提出建立触发器的要求而不得自行建立，触发器的建立必须由公司统一规划建立、修改和发布。

触发器不允许创建数据库对象。

触发器中一般不使用 SELECT 语句返回信息。

对 Oracle 中具有以下结构的视图：

：类似 AVG 和 COUNT 的集合函数；

：GROUP BY 字句；

：DISTINCT 关键字；

：包含多种类型的连接；

建立触发器时，必须使用 INSTEAD-OF 触发器，INSTEAD-OF 触发器提供的代码可以由 Oracle 用户编写的任何 DML 语句中得到执行。

注释内容：

触发器的注释，必须准确描述触发器的目的、涉及的相关数据库表，变量定义、实现方法。此外对触发器的创建时间、创建人员等信息描述清楚。

10.5 脚本注释规范

对于数据库脚本程序块，例如存储过程、自定义函数等，需要有标准的注释，以便所有软件人员都能一目了然的看懂当前程序块的作用及流程。

10.5.1 存储过程

```
/**********************************************************************
过 程 名 称：
调用方式：
参数说明：
返回结果：
过程功能：
操作说明：
设计时间：
代码设计者：
==========================================================================
实现方法：
------------------------------修改记录-----------------------------------
  修改人              修改时间                  修改原因
--------------------------------------------------------------------------
==========================================================================
备注：
```

**********/

说明：操作说明中必须给出当前存储过程检索到的数据库表（或视图）、修改（删除、增加）记录的库表、执行的其它存储过程。

10.5.2 触发器

/**********

触发器名称：

触发器功能：

操作说明：

设计时间：

代码设计者：

****

实现方法：

----------修改记录----------

修改人　　修改时间　　修改原因

==========

备注：

**********/

10.5.3 视图

/**********

视图名称：

目　　的：

设计时间：

视图设计者：

（表、）

****

实现方法：

==========

----------修改记录----------

修改人　　修改时间　　修改原因

==========

备注：

**********/

10.5.4 其它

◇ 修改处的注释遵照代码规范的修改处注释，参见本文档的3.5 代码修改注释；

◇ 在每一个变量的行后部或上一行，说明变量的用途；

◇ 程序中尽可能随时注释当前行或当前代码块的作用。

10.6 命名规范

原则：

◇ 对数据库所有对象的命名，要本着清晰、简短的原则，名称反映功能，力求见名知义；如果名称太长，可以使用部分重点单词相连，能接近本意即可。

◇ 所有字母全部小写，合理使用下划线进行分割。

10.6.1 表命名

使用描述性的单词或其缩写，所有字母全部小写，多个单词之间以下划线连接，达到见名知义的效果。

如 student_grade：学生成绩表

10.6.2 字段命名

使用功能描述性的单词或其缩写，所有字母全部小写，多个单词之间以下划线连接，达到见名知义的效果。例如 class_name

10.6.3 视图命名

视图名称由三部分组成：

前缀（v）+下划线+描述性名称

例如：v_student_class

10.6.4 存储过程命名

存储过程名称由四部分组成：

前缀（sp）+下划线+描述性名称

例如：

10.6.5 索引命名

索引过程名称由五部分组成：

前缀（i）+是否唯一（u、d）+是否聚簇（c、n）+下划线+索引名称

例如 iuc_name 创建的唯一且是聚簇型的索引；idn_usename 创建的非唯一且不是聚簇型的索引。

10.6.6 触发器命名

触发器名称由四部分组成：

前缀（t）+触发器类型（i、u、d）+下划线+库表名称

例如 ti_student_base 在库表上创建的插入触发器；tu_student_detail 在库表上创建的修改触发器。

10.7 脚本书写规范

◇ 脚本中所有的字符串都用小写；

◇ 同级别的对齐，下级代码使用至少四个空格来缩进结构；

◇ 随时注释；

下面以程序主体片断举例说明：

……

```
select reserved_id into reserve_key
from books_reserved
where reserved_book_id=book_num;  --注释
select book_loan_id INTO loan_key
from books
where book_id = book_num;  --注释
```

```
if loan_key > 0 OR reserve_key > 0 THEN  --注释
    if loan_key > 0 then  --注释
        Statement
    end if;
    if reserve_key > 0 then  --注释
        Statement
    end if;
end if;
……
```

10.8 PowerDesigner 建模规范

必须统一使用 PowerDesigner 建模工具来设计数据库结构，并严格遵照数据库对象的命名规范和注释规范，具体包括以下事项：

- ◇ 不得使用数据库工具（比如 Sql Server、Oracle）直接创建以下数据库对象：表、视图、存储过程、自定义函数、触发器、关联、序列（Oracle）以及关键字、索引等，必须使用建模工具对整个数据库结构进行设计、创建；
- ◇ 整个软件开发过程中，当需要更新数据库对象，即增加、修改、删除等数据库对象时，必须使用 PowerDesigner 工具来实现，不得直接使用 SQLServer 或 Oracle 工具更新；
- ◇ 组员需要更新数据库对象时必须经过项目负责人的批准，并由专人负责对服务器上数据库对象的更新；
- ◇ 使用PowerDesigner建表时，字段的Name列必须输入代表对应字段含义的中文，Code列必须输入对应字段的英文名称（详见数据库命名规范）。

10.9 Oracle 数据库表空间

表空间命名：

见名知义，名称反映当前表空间的作用；使用有明确意义的单词来命名，多个单词之间以下划线相连；表空间的所有字母全部大写。例如 CHECK_HISTORY

表空间大小：

用户所有的数据都存在于表空间内，所以表空间的大小要根据实际情况合理估算大小。为了保证表空间中存储对象数据的可扩充性，在定义表空间时需要将存储页设置为数据文件满后自动扩展，并按照实际情况，合理分配扩展空间的大小。

创建表空间举例说明：

```
CREATE TABLESPACE "CHECK_HISTORY"  /*表空间名称*/
    DATAFILE 'C:\ORACLE\ORADATA\ORADB\CHECK_HISTORY.ora'
    /*数据文件名及路径*/
    SIZE 100M /*数据文件初始大小*/
    AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
    /*数据文件自动扩展，每次扩展 100M，无最大限制*/
    MINIMUM EXTENT 25M /*最小范围 25M
```

表空间碎片：

删除表空间中的对象后，表空间内会产生碎片，对于很小的碎片（例如 5M）ORACLE 通常不能使用。所以必须确保即使碎片产生，也要保证碎片范围要在可用范围之内。为了阻止

表空间被分割成许多很小不能用的范围，ORACLE 允许为表空间指定最小范围。例如上例中，最小范围规定为 25M，那么 ORACLE 不允许任何小于 25M 的范围被分配到表空间 CHECK_HISTORY 中。当然，用初始容量 5M 能成功创建一个表，但是 ORACLE 将会默认分配一个 25M 的初始范围，因为这是该表空间允许的最小范围。

10.10 Sql Server 数据库属性设置

数据库大小

数据库大小随着系统运行会逐步膨胀，所以数据库最大容量不能限制，如图 9-9-1 中椭圆区域的设置。

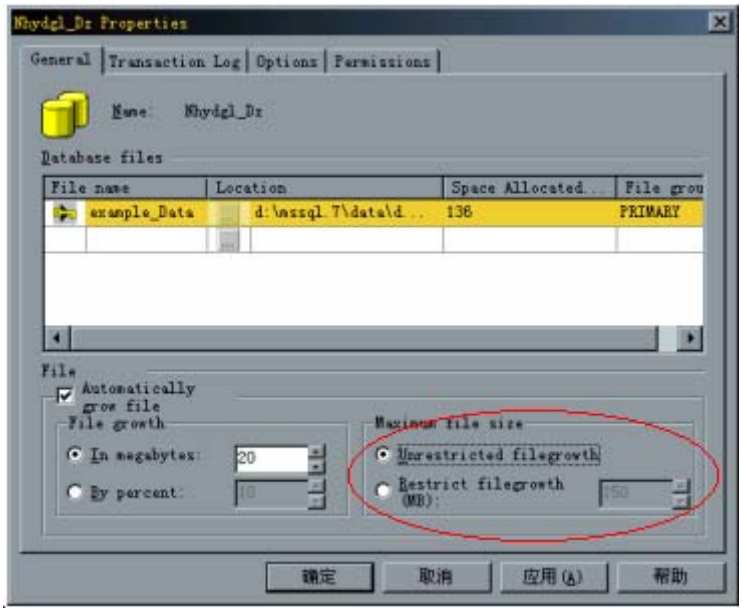

图 9-9-1 数据库最大容量设置

数据库膨胀的幅度

由于数据库的大小随数据量的增多而增大，所以在创建数据库时，要设置让 SQL Server 自动维护数据库大小，且根据数据库大小的百分数（20%-30%）进行维护。如图 9-9-2 所示。

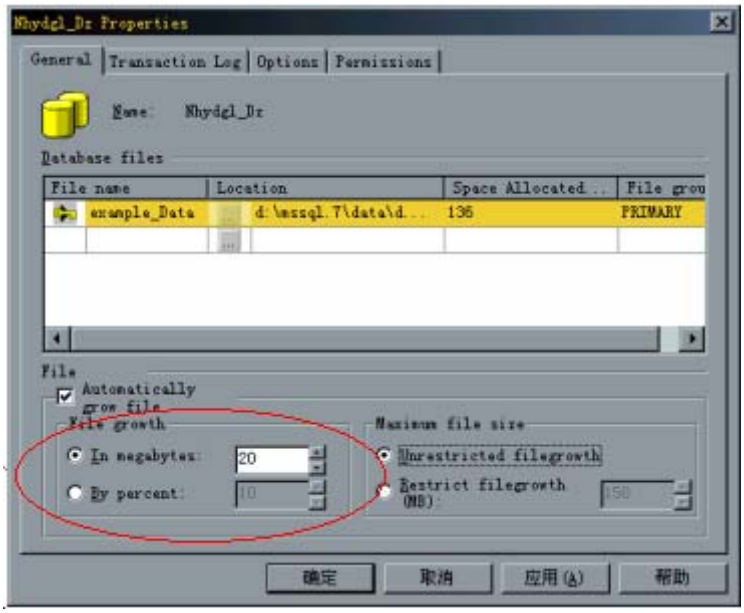

图 9-9-2 数据库增长速度设置