## [\_variant\_t、CComVariant与COleVariant、CDBVariant](http://blog.csdn.net/veryhehe2011/article/details/8141264)

目前计算机语言多种多样，如C++、Java、Basic、Pascal等，此外还有JavaScript、VBScript、ActionScript等脚本语言，它们各自维护自己的数据类型，当使用C++这样强类型的语言来读取数据库或者与其他语言之间来交换数据时，它很有可能不知道获取到的数据的具体类型，这个时候必须借助于变体类型读取数据。VARIANT数据类型就具有跨语言的特性，同时它可以表示（存储）任意类型的数据。其在Visual C++中的定义：

typedef tagVARIANT VARIANT;

typedef struct tagVARIANT VARIANTARG;

VARIANT 其实是一个结构，结构中用一个vt成员表示数据的类型，同时真正的数据则存储在union空间中。一般我们使用VARIANT的步骤如下所示。

定义一个VARIANT变量，如：var。

通过vt成员设定VARIANT变量的数据类型，如：var.vt = VT\_I4。

通过对应的union成员设定数据内容，如：var.lVal = 100。

综上所述，

利用VARIANT表示一个整型数据：

1. VARIANT var;
2. var.vt = VT\_I4;     //指明整型数据
3. var.lVal = 100;     //赋值

利用VARIANT表示一个布尔值：

1. VARIANT var;
2. var.vt = VT\_BOOL;                   //指明整型数据
3. var.boolVal = VARIANT\_TRUE;     //赋值

利用VARIANT保存一个字符串：

1. VARIANT var;
2. var.vt = VT\_BSTR;
3. var.bstrVal = SysAllocString(L"hello, world!");

根据以上的代码，读者可能会猜到，VARIANT的定义可能类似于如下：

1. struct VARIANT
2. {
3. VARTYPE vt;                     //数据类型
4. union
5. {
6. LONG            lVal;       //VT\_I4
7. VARIANT\_BOOL    boolVal     //VT\_BOOL
8. BSTR            bstrVal;    //VT\_BSTR
9. }
10. };

实际上，VARIANT的定义就是这样的！只不过由于它需要支持的类型太多，所以它包含的联合成员会更多。限于篇幅，在此不再附出。  
  
  
VARIANT支持的类型，也就是vt成员的取值如表4-3所示。

表4-3 VARIANT支持的类型  
  
  
类型名  
  
  
含义  
  
  
VT\_EMPTY  
  
  
指示未指定值  
  
  
VT\_NULL  
  
  
指示空值（类似于 SQL 中的空值）  
  
  
VT\_I2  
  
  
指示 short 整数  
  
  
VT\_I4  
  
  
指示 long 整数  
  
  
VT\_R4  
  
  
指示 float 值  
  
  
VT\_R8  
  
  
指示 double 值  
  
  
VT\_CY  
  
  
指示货币值  
  
  
VT\_DATE  
  
  
指示 DATE 值  
  
  
VT\_BSTR  
  
  
指示 BSTR 字符串  
  
  
VT\_DISPATCH  
  
  
指示 IDispatch 指针  
  
  
VT\_ERROR  
  
  
指示 SCODE  
  
  
VT\_BOOL  
  
  
指示一个布尔值  
  
  
VT\_VARIANT  
  
  
指示 VARIANTfar 指针  
  
  
VT\_UNKNOWN  
  
  
指示 IUnknown 指针  
  
  
VT\_DECIMAL  
  
  
指示 decimal 值  
  
  
VT\_I1  
  
  
指示 char 值  
  
  
  
  
（续表）  
  
  
  
  
  
  
类型名  
  
  
含义  
  
  
VT\_UI1  
  
  
指示 byte  
  
  
VT\_UI2  
  
  
指示 unsignedshort  
  
  
VT\_UI4  
  
  
指示 unsignedlong  
  
  
VT\_I8  
  
  
指示 64 位整数  
  
  
VT\_UI8  
  
  
指示 64 位无符号整数  
  
  
VT\_INT  
  
  
指示整数值  
  
  
VT\_UINT  
  
  
指示 unsigned 整数值  
  
  
VT\_VOID  
  
  
指示 C 样式 void  
  
  
VT\_HRESULT  
  
  
指示 HRESULT  
  
  
VT\_PTR  
  
  
指示指针类型  
  
  
VT\_SAFEARRAY  
  
  
指示 SAFEARRAY  
  
  
VT\_CARRAY  
  
  
指示 C 样式数组  
  
  
VT\_USERDEFINED  
  
  
指示用户定义的类型  
  
  
VT\_LPSTR  
  
  
指示一个以 NULL 结尾的字符串  
  
  
VT\_LPWSTR  
  
  
指示由 nullNothingnullptrnull  
  
  
引用（在 Visual Basic  
  
  
中为 Nothing） 终止的宽字符串  
  
  
VT\_RECORD  
  
  
指示用户定义的类型  
  
  
VT\_FILETIME  
  
  
指示 FILETIME 值  
  
  
VT\_BLOB  
  
  
指示以长度为前缀的字节  
  
  
VT\_STREAM  
  
  
指示随后是流的名称  
  
  
VT\_STORAGE  
  
  
指示随后是存储的名称  
  
  
VT\_STREAMED\_OBJECT  
  
  
指示流包含对象  
  
  
VT\_STORED\_OBJECT  
  
  
指示存储包含对象  
  
  
VT\_BLOB\_OBJECT  
  
  
指示 Blob 包含对象  
  
  
VT\_CF  
  
  
指示剪贴板格式  
  
  
VT\_CLSID  
  
  
指示类 ID  
  
  
VT\_VECTOR  
  
  
指示简单的已计数数组  
  
  
VT\_ARRAY  
  
  
指示 SAFEARRAY 指针  
  
  
VT\_BYREF  
  
  
指示 值为引用

4.2.5  \_variant\_t、CComVariant与COleVariant、CDBVariant

从上面可以看出VARIANT这种类型使用起来比较复杂，其实有简单的办法，那就是采用VARIANT的封装类\_variant\_t。\_variant\_t的构造函数接受基本数据类型的数据作为参数，如下列出其中的一小部分：

1. \_variant\_t(
2. **short** sSrc,
3. VARTYPE vtSrc = VT\_I2
4. );
6. \_variant\_t(
7. **long** lSrc,
8. VARTYPE vtSrc = VT\_I4
9. );
11. \_variant\_t(
12. **float** fltSrc
13. ) **throw**( );
15. \_variant\_t(
16. **double** dblSrc,
17. VARTYPE vtSrc = VT\_R8
18. );

另一方面，\_variant\_t提供了反向的转换函数，如将一个\_variant\_t转换成一个short数值，如下列出其中的一小部分：

1. operator **short**( ) **const**;
2. operator **long**( ) **const**;
3. operator **float**( ) **const**;
4. operator **double**( ) **const**;

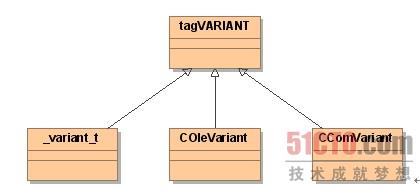
因此可以看出，利用\_variant\_t可以很方便地实现VARIANT类型和基本数据类型之间的转换，如：

1. **long** l = 123;
2. \_variant\_t lVal(l);
3. **long** m = lVal;

也可以用COleVariant和CComVariant来简化对VARIANT的操作，代码参考如下：

1. COleVariant v3 = \_T("hello, world!");
2. COleVariant v4 = (**long**)1999;
3. CString str = (BSTR)v3.pbstrVal;
4. **long** i = v4.lVal;

VARIANT类图如图4-7所示。



此外，在MFC ODBC 编程中，我们还会接触到CDBVariant，CDBVariant没有任何基类，它的功能与COleVariant相似，唯一的差别是它不使用OLE。可以看出，Visual C++为变体提供了太多的封装类，如果可行的话，我们建议读者尽量在自己的代码中采用统一的类，如：\_variant\_t。