Problem report - pgSchema

pgDatabase

(declarations of pgDatabase)

```
23 Object variable declared As New: cnDatabase
```

pgDatabase.KillTypeCache

```
   Dead procedure
```

pgDatabase.LookupType

```
   Dead procedure
 4 Object variable declared As New: rs
```

pgDatabase.KillLanguageCache

```
   Dead procedure
```

pgDatabase.LookupLanguage

```
   Dead procedure
 4 Object variable declared As New: rs
```

pgDatabase.KillOperatorCache

```
   Dead procedure
```

pgDatabase.LookupOperator

```
   Dead procedure
 4 Object variable declared As New: rs
```

pgDatabase.LookupComment

```
 5 Object variable declared As New: rs
```

pgDatabase.Comment [Get]

```
 5 Consider short circuit with nested Ifs
```

pgDatabase.Vacuum

```
13 Consider short circuit with nested Ifs
```

pgDatabase.DatabaseVarList [Get]

```
   Dead procedure
```

pgDatabase.Grant

```
 8 Object variable declared As New: rs
```

pgDatabase.Revoke

```
 8 Object variable declared As New: rs
```

Databases

Databases.iAdd

```
   Function without type specification
```

Databases.Add

```
 7 Object variable declared As New: rs
 8 Object variable declared As New: rsComment
 9 Dead variable: rsUser
 9 Object variable declared As New: rsUser
```

Databases.Initialize

```
 6 Dead variable: szSQL
 7 Object variable declared As New: rs
```

basGlobal

(declarations of basGlobal)

```
18 Dead constant: ODBC_CONNECT_OPTIONS
24 Dead constant: SQL_GET_USERS
25 Dead constant: SQL_GET_GROUPS
26 Dead constant: SQL_GET_SEQUENCES
27 Dead constant: SQL_GET_VIEWS7_1
28 Dead constant: SQL_GET_VIEWS7_3
29 Dead constant: SQL_GET_TYPES7_1
30 Dead constant: SQL_GET_TYPES7_3
31 Dead constant: SQL_GET_DOMAINS
32 Dead constant: SQL_GET_FUNCTIONS7_1
33 Dead constant: SQL_GET_FUNCTIONS7_3
34 Dead constant: SQL_GET_OPERATORS
35 Dead constant: SQL_GET_RULES7_1
36 Dead constant: SQL_GET_RULES7_3
37 Dead constant: SQL_GET_TRIGGERS
40 Dead constant: SQL_GET_COLUMNS7_1
41 Dead constant: SQL_GET_COLUMNS7_2
42 Dead constant: SQL_GET_COLUMNS7_3
43 Dead constant: SQL_GET_INDEXES
44 Dead constant: SQL_GET_INDEX_COLUMNS
45 Dead constant: SQL_GET_CHECKS7_2
46 Dead constant: SQL_GET_CHECKS7_3
47 Dead constant: SQL_GET_INHERITED_TABLES
48 Dead constant: SQL_GET_AGGREGATES7_1
```

```
49 Dead constant: SQL_GET_AGGREGATES7_3
50 Dead constant: SQL_GET_FOREIGN_KEYS
51 Dead constant: SQL_GET_NAMESPACES
52 Dead constant: SQL_GET_CASTS
53 Dead constant: SQL_GET_CONVERSIONS
```

basGlobal.GetVersionEx

```
   Dead declaration (called by dead only)
```

basGlobal.GetModuleFileName

```
   Dead declaration
```

basGlobal.fmtID

```
16 Unicode function is faster: AscW
19 Consider short circuit with nested Ifs
```

basGlobal.fmtTypeID

```
 4 Dead variable: iLen
20 Unicode function is faster: AscW
23 Consider short circuit with nested Ifs
```

basGlobal.ULEncode

```
11 Unicode function is faster: AscW
11 Unicode function is faster: AscW
11 Consider short circuit with nested Ifs
```

basGlobal.GetUniqueID

```
   Dead procedure
```

basGlobal.WinVer

```
   Dead procedure
```

basGlobal.WinBuild

```
   Dead procedure
```

basGlobal.WinName

```
   Dead procedure
```

basGlobal.WinInfo

```
   Dead procedure
11 Constant available: vbNullChar
```

Tables

Tables.iAdd

```
 7 Dead variable: szSQL
 8 Object variable declared As New: rs
```

Tables.Add

```
 7 Object variable declared As New: rs
```

Tables.Rename

```
 7 Object variable declared As New: objTable
```

Tables.Initialize

```
 6 Object variable declared As New: rs
```

pgTable

pgTable.Rows [Get]

```
 7 Object variable declared As New: rs
```

pgTable.Grant

```
 8 Object variable declared As New: rs
```

pgTable.Revoke

```
 8 Object variable declared As New: rs
```

pgTable.SQL [Get]

```
10 Dead variable: objRelationship
44 Consider short circuit with nested Ifs
```

pgRelationship

pgRelationship.NamespaceOID [Let]

```
   Dead procedure
```

pgRelationship.Namespace [Let]

```
   Dead procedure
```

pgRelationship.Connection [Set]

```
   Dead procedure
```

pgRelationship.Identifier [Let]

```
Dead procedure
```

pgRelationship.LocalColumn [Let]

```
Dead procedure
```

pgRelationship.ReferencedColumn [Let]

```
Dead procedure
```

pgNamespace

pgNamespace.Connection [Set]

```
Dead procedure
```

pgNamespace.Oid [Let]

```
Dead procedure
```

pgNamespace.Name [Let]

```
Dead procedure
```

pgNamespace.Identifier [Let]

```
Dead procedure
```

pgNamespace.SystemObject [Let]

```
Dead procedure
```

pgNamespace.Owner [Let]

```
Dead procedure
```

pgNamespace.ACL [Let]

```
Dead procedure
```

pgNamespace.Grant

```
 8 Object variable declared As New: rs
```

pgNamespace.Revoke

```
 8 Object variable declared As New: rs
```

DatabaseVars

(declarations of DatabaseVars)

```
 9 Dead variable: szName
10 Dead variable: szValue
```

DatabaseVars.iAdd

```
 2 Parameter without type specification: szValue
 6 Object variable declared As New: objVar
```

DatabaseVars.Refresh

```
 6 Dead variable: szSQL
 8 Object variable declared As New: rs
```

DatabaseVars.Initialize

```
 6 Dead variable: szDatabaseVars
29 Consider short circuit with nested Ifs
```

basDepend

basDepend.DepRef

```
 9 Object variable declared As New: colDep
10 Variable without type specification: objTmp
```

basDepend.AddObjDepend

```
 2 Parameter without type specification: ObjFind
 6 Variable without type specification: objTmp
```

basDepend.GetObjectTypePgClass

```
   Function without type specification
 8 Variable without type specification: objTmp
```

Problematic areas

File	Problem count	
basGlobal	45	********
pgDatabase	16	***
pgNamespace	9	**
DatabaseVars	8	*
Databases	7	*
pgRelationship	6	*
basDepend	6	*
pgTable	5	*
Tables	5	*

Problem summary

Problem	Type	Count

```
Consider short-circuited logic
  Optim.         7
Constant available
  Optim.         1
Dead constant
  Optim.        29
Dead procedure/declaration/event
  Optim.        26
Dead procedure/declaration/event (called by dead only)
  Optim.         1
Dead variable/parameter
  Optim.         9
Function without type specification
  Optim.         2
Object variable declared As New
  Optim.        23
Unicode function is faster
  Optim.         4
Variable without type specification
  Optim.         5
```

Type	Count	
Optimization	107	**********
Total	107	
Problems/logical code line	0.03	

Filter: <Default>

Problem descriptions

Consider short-circuited logic

In the expressions (x And y), (x Or y), both operands (x, y) are evaluated.
Short-circuiting means rewriting this so that when x=False in (x And y), y is
not evaluated. The same goes for x=True in (x Or y). This saves CPU cycles,
especially if y is a complex expression. In VB.NET, consider replacing And with
AndAlso, and Or with OrElse. In VB Classic, consider splitting an If ..And..
condition as two nested Ifs. Short-circuiting If ..Or.. yields more complex
code, usable case by case. Risks: Short-circuiting changes the logic. If the
second operand calls a function, this call may not execute. Read VB help for
differences between And/AndAlso and Or/OrElse. Optimization. Severity: Info.

Constant available

A constant is available in place of a function call. Use a string constant
instead of Chr/ChrW. The available string constants and their ASCII values are:

vbNullChar (0), vbBack (8), vbTab (9), vbLf (10), vbVerticalTab (11), vbFormFeed (12), vbCr (13), vbCrLf or vbNewline (13 & 10). vbNewline is faster than vbCrLf. Successive Chr(13) & Chr(10) should be replaced by vbNewline, not vbCr & vbLf. - Instead of a call such as Asc("A"), use a numeric constant such as Const ascA = 65. - These rules apply to VB 4-6. In VB.NET the compiler takes care of optimizing the use of these functions. Optimization. Severity: Info.

Dead constant
A variable or constant is not used. You may remove it if you are sure you won't need it later. The removal doesn't affect the functionality of your program. Optimization. Severity: Warning.

Dead procedure/declaration/event
A procedure, a DLL declaration or an Event declaration is not used by the project. It is not called by the code nor executed by any other means. You may remove it if you are sure you won't need it later. The removal doesn't affect the functionality of your program. - Event declarations are reported dead only if they are not raised nor handled. See the problem Event not raised for events that would be handled but that don't fired. Optimization. Severity: Warning.

Dead procedure/declaration/event (called by dead only)
You should remove this procedure along with its callers, provided that you are sure you won't need any of the callers later. Optimization. Severity: Warning.

Dead variable/parameter
A variable or constant is not used. You may remove it if you are sure you won't need it later. The removal doesn't affect the functionality of your program. Optimization. Severity: Warning.

Function without type specification
A function does not have a defined return data type. By default, the type is Variant. Variant needs more memory than other types. Decide what type you need and write it to the function declaration. Besides, upgrading to VB.NET will be easier if you use explicit data types. Fix recommended before upgrade. Optimization. Severity: Warning.

Object variable declared As New

In VB Classic, declaring an object variable As New creates an auto-instantiating variable. Each time you read the contents of the variable, VB first checks if the variable contains an object, and creates one if not. This adds overhead, thus slowing your program down. To achieve better performance, remove the word New from the declaration, and instantiate your variable (Set x = New Class) before it is used. It makes sense to test with 'If x Is Nothing Then' before accessing the variable, to avoid the run-time error 'Object variable not set'. In addition, VB.NET has different semantics for As New. Applies to VB 3-6. Optimization. Severity: Warning.

Unicode function is faster

The wide functions AscW and ChrW/ChrW$ are faster than the Asc/Chr/Chr$ alternatives. VB works internally in Unicode, so the unicode versions run faster. They are not the same functions though. If you're handling ASCII characters from 0 to 127, you're safe to replace Asc with AscW and Chr with ChrW/ChrW$. Applies to VB4 and later. Optimization. Severity: Info.

Variable without type specification

A variable does not have a defined data type. By default, the type is Variant. Variant needs more memory than other types. Decide what type you need and write it to the variable declaration. Besides, upgrading to VB.NET will be easier if you use explicit data types. Fix recommended before upgrade. Optimization. Severity: Warning.