The ACCESS table

Column	Description	Remarks
module	The complete name of the file where the access of the variable occurs, including the full path to the file.	
start	The line number for the beginning of the definition of the function where the access takes place.	
start_char	The character position where the access begins	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop	The line number where the access ends.	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop_char	The character position where the access ends	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
package	For C++ projects this column will contain the path to the source file, relative to the root directory from where the parsing began. For Java projects this will have the usual package semantic	This is useful for the measurement of subsystem couplling.
function	The name of the function where the access ocurred.	For constructors, the name of the class prefixes the function name (e.g. "`String::String`"), due to compatibility reasons to *Sema-Audit*
signature	The signature of the function where the access ocurred.	This field is void ("") if the function has no parameters.
name	The name of the accessed variable	
type	The data-type of the accessed variable	
provider_package	This is the name of the package (subsystem) where the accessed variable is defined.	This is useful for the measurement of subsystem couplling.
provider_class	The name of the class where the accessed variable was defined in.	
use (name will be probably changed to **access_specifier**)	Indicates what kind of variable was accessed (parameter, global variable, attributes etc.)	The column must have one following values: • *global* – for global variables • *param* – for parameters • *local* – for local variab. • *attr-public* – for public attributes • *attr-private* – for private attrib. • *attr-protected* – for protected attr.
is_static	Specifies if the accessed variable is static or not	1 – if static; 0 – if not static (usual)
is_complex	Specifies if the the type of the accessed variable is a predefined or user-defined.	1 – if the type is user-defined (i.e. a class type) 0 – if the type is predefined (e.g. int, char etc)
is_interface	This field differentiates between accessing the interface or the data of a class	*Obsolete*. Was used exclusively for the implementation of the CDBC metric.
how_many	The number of accesses to the same variable within one function	

The CALL table

Column	Description	Remarks
module	The complete name of the file where the call occurs, including the full path to the file.	
start	The line number for the beginning of the definition of the function where the call takes place.	
start_char	The character position where the function begins	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop	The line number where the function ends.	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop_char	The character position where the function ends	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
package	see description for the ACCESS table	
class	The name of the class where the call occurs.	If the namespace for the class is not global this column will contain the full name: *namespace_name::class_name*
function	The name of the function where the call ocurrs.	For constructors, the name of the class prefixes the function name (e.g. "`String::String`"), due to compatibility reasons to *Sema-Audit*
signature	The signature of the function where the call ocurrs.	This field is void ("") if the function has no parameters.
access_specifier	Specifies the kind of the function where the access occurs	The column must have one following values: • *single-function* – for global functions • *public-method* – for a public method • *private-method* – for a private meth. • *protected-method* – for a protected method
called_package	This is the name of the package (subsystem) where the called function is defined.	In C++ for library function this field is void. In Java the name of the package is provided.
called_class	The name of the class where the called function is defined.	
called_function	The name of the called function	
called_signature	The signature of the called function	This field is void if the function has no parameters.
called_access_specifier	Specifies the kind of function where the access occurs	The column must have one following values: • *single-function* – for global functions • *public-method* – for a public method • *private-method* – for a private meth. • *protected-method* – for a protected meth. • *library-function* – for functions used, but not defined in the within the project.
how_many	The number of invocations of the same method within the same caller function	Regardless of the number of calls to one function, we will have for it/them just one entry in the table.
is_overloaded	Indicates if the specified **called_function** is overloaded or not. The called_function is overloaded, *only if there is more than one implementation having the same number of parameters*.	In C++ the detection of the *exact* called signature is hard. Therefore where ambiguities appear – ie. where the called function is overloaded – we consider a call to *each* of the overloaded functions and consequently set this field to 1. In all the other cases, where there are no ambiguities this field is 0.

The CLASSES table

Column	Description	Remarks
module	The complete name of the file where the class is *declared* (in C++ usually this is the header file).	
start	The line number for the beginning of the class declaration.	
start_char	The character position where the class declaration begins	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop	The line number where the class declaration ends.	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop_char	The character position where the class declaration ends	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
package	see description for the ACCESS table	
class	The name of the class.	If the namespace for the class is not global this column will contain the full name: *namespace_name::class_name*
scope	For *inner* classes, this field keeps the name of the "host" class, which directly contains the class.	For all the other classes, except the internal ones, this field is void ("")
is_abstract	Specifies if the class is abstract	1 – if the class is abstract 0 – if the class is not abstract (is an implementation class)
is_template	Specifies if the class is a *generic (template) class* (For Java this field is always 0)	0 – if the class is generic 1 – if the class is not
NEW COLUMN (not added yet)	**Description**	**Remarks**
namespace	The name of the namespace in which the class is defined.	This field is specific for C++ The **package** and the **namespace** fields are somehow related, as both might be seen as mechanisms for encapsulating subsystems; while **package** is rather physical, **namespace** is logical. Yet namespace is a "weaker" mechanism, therefore package should be used for the analysis at the subsystem level.

The DECLARE table

Column	Description	Remarks
module	The complete name of the file where the variable is *declared*	
start	The line number for the beginning of the variable declaration.	
start_char	The character position where the variable declaration begins	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop	The line number where the variable declaration ends.	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop_char	The character position where the variable declaration ends	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
namespace	The name of the namespace in which the variable is defined.	If namespace is the default one the field is void ("")
package	see description for the ACCESS table	
class	The name of the class where the variable (i.e. attribute, parameter of a method or local var. in a method)	This field is void for global variables (in C++)
function	The name of the function where the variable (i.e. parameter or local variab.) is defined.	This field is void for global variables (in C++) and for attributes For constructors, the name of the class prefixes the function name (e.g. "`String::String`"), due to compatibility reasons to *Sema-Audit*
signature	The signature of the function.	This field is void if the function has no parameters. This field is void for global variables (in C++)
name	The name of the declared variable	
type	The *base* data-type of the variable	Does not include the pointer sign (*) or reference sign (&) . (e.g. the base-type for "`char**`" is `char`)
type_compl	The complete data-type of the variable	In this field the type is represented exactly as it appears in the source code.
use (name will be probably changed to **access_specifier**)	Indicates what the kind of variable was defined.	The column must have one following values: • *global* – for global variables • *param* – for parameters • *local* – for local variab. • *attr-public* – for public attributes • *attr-private* – for private attrib. • *attr-protected* – for protected attr.
is_complex	Specifies if the the type of the variable is predefined or a user-defined.	1 – if the type is user-defined (i.e. a class type) 0 – if the type is predefined (e.g. int, char, double etc)
is_template	Specifies if the type of the variable is a *generic (template) type*	0 – if the type of the var. is generic 1 – if the type of the var. is not

The FUNCS table

Column	Description	Remarks
module	The complete name of the file where the function/method is *defined* (implemented)	
start	The line number for the beginning of the function definition.	
start_char	The character position where the function definition begins	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop	The line number where the function definition ends.	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop_char	The character position where the function definition ends	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
package	see description for the ACCESS table	
class	The name of the class	This field is void for global functions (in C++) If the namespace for the class is not global this column will contain the full name: *namespace_name::class_name*
function	The name of the function	For constructors, the name of the class prefixes the function name (e.g. "`String::String`"), due to compatibility reasons to *Sema-Audit*
signature	The signature of the function.	This field is void if the function has no parameters.
return	The type of the object returned by the function.	The return type is prefixed with the '`const`' specifier if the function was defined so. This field is void for constructors and destructors.
use (name will be probably changed to **access_specifier**)	Specifies the kind of the function	The column must have one following values: • *single-function* – for global functions • *public-method* – for a public method • *private-method* – for a private meth. • *protected-method* – for a protected meth. • *method-definition* – this value was introduced in order to deal with the erroneous cases in C++ where a method is defined without being declared in the class. Because of the missing declaration the access-specifier cannot be set.
storage (name will be probably changed to **storage_specifier**)	This field keeps the storage-specifier for a function (see Remark)	The column must have one following values: • ' ' – for the usual, non-virtual functions • *virtual* – for virtual functions • *static* – for static functions • *const* – if the function is constant.
ct_cyclo	This field stores the value of the *cyclomatic number* for the function.	This is a very used procedural metric defined by McCabe, known also as the cyclomatic complexity.

The INH table

Column	Description	Remarks
module	The complete name of the file where the derived class is *declared*	
start	The line number for the beginning of the class declaration.	
start_char	The character position where the class declaration begins	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop	The line number where the class declaration ends.	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
stop_char	The character position where the class declaration ends	*Obsolete*. Kept only for compatibility reasons to *Sema-Audit*
package	see description for the ACCESS table	
class	The name of the derived class.	If the namespace for the class is not global this column will contain the full name: *namespace_name::class_name*
parent	The name of the direct or indirect ancestor class	The description wants to emphasize the fact that the table contains the transitive closure of the inheritance relations
attribute	Specifies the way the derived class inherits from the parent class. This attribute influences the visibility of the members from the parent class in the derived class.	The column must have one following values: • *public* • *private* • *protected*
ct_dit	Specifies the "vertical" distance in the inheritance tree between the parent class and the child class	This field is in fact the DIT value for the class.