[General and NASIS Specific Data Modeling Conventions 5](#_Toc9516223)

[General NASIS Logical and Physical Naming Conventions 6](#_Toc9516224)

[System and Data Model Related 6](#_Toc9516225)

[Specifying the Data Model Name and Version 6](#_Toc9516226)

[Specifying a System Sequence 6](#_Toc9516227)

[NASIS Data Object (Editor) Related 6](#_Toc9516228)

[Specifying that a NASIS Data Object should be "Selectable for Replication" in NASIS (NASIS Specific) 7](#_Toc9516229)

[Specifying that a NASIS Data Object should be "AutoReplicated" to the Local NASIS Database (NASIS Specific) 7](#_Toc9516230)

[Specifying that a NASIS User should not be allowed to Insert Records into a Table (NASIS Specific) 7](#_Toc9516231)

[Specifying that a Table is Specific to a NASIS Client Database (NASIS Specific) 7](#_Toc9516232)

[Specifying that a Table is Specific to the NASIS Server Database (NASIS Specific) 7](#_Toc9516233)

[Specifying the SQL Statement whose Output is Used as Part of an Import Diagnostic Message for a Table Whose Contents can be Imported from a Pedon PC Database (NASIS Specific) 8](#_Toc9516234)

[Specifying that a "Table" should be created as a View 8](#_Toc9516235)

[Specifying that a Table is a Temporary Table 8](#_Toc9516236)

[Column Related 8](#_Toc9516237)

[Which Columns should typically be Included in a NASIS Data Object's Root Table? (NASIS Specific) 8](#_Toc9516238)

[Which Columns should typically be Included in a NASIS Table that is not a NASIS Data Object's Root Table? (NASIS Specific) 9](#_Toc9516239)

[Specifying that a Column should be "Not Null" 9](#_Toc9516240)

[Specifying that a Column corresponds to a Spatial Data Type 9](#_Toc9516241)

[Specifying that a Column should be Visible in NASIS (NASIS Specific) 10](#_Toc9516242)

[Specifying that a Column should not be Editable in NASIS (NASIS Specific) 10](#_Toc9516243)

[Specifying that a Column's Value can be generated by a Calculation (NASIS Specific) 10](#_Toc9516244)

[Specifying which NASIS Data Object Root Table Columns should be Included in that NASIS Data Object Table's Replication Select List (NASIS Specific) 10](#_Toc9516245)

[Specifying that a Column has an Object Modify Default (NASIS Specific) 11](#_Toc9516246)

[Specifying that a Column has a Record Modify Default (NASIS Specific) 11](#_Toc9516247)

[Specifying Which Columns in one or more Related Tables should be displayed in lieu of a Foreign Key Column, aka "Column Lookups" (NASIS Specific) 11](#_Toc9516248)

[The Gory Details 12](#_Toc9516249)

[Index, Primary Key, Foreign Key and Unique Constraint Related 13](#_Toc9516250)

[Index, Primary Key, Unique Constraint and Spatial Index Naming Conventions 13](#_Toc9516251)

[Duplicate and Performance Indexes 13](#_Toc9516252)

[Primary Key Constraints 13](#_Toc9516253)

[Unique Constraints 14](#_Toc9516254)

[Spatial Indexes 14](#_Toc9516255)

[Primary Key Conventions 14](#_Toc9516256)

[Specifying a Table's Primary Key 14](#_Toc9516257)

[Specifying that an Index should be Clustered 15](#_Toc9516258)

[Specifying an Index as a Performance Index 15](#_Toc9516259)

[Specifying a "Deferrable Unique Constraint" (NASIS Specific) 15](#_Toc9516260)

[Specifying that an Index Corresponds to a Foreign Key 15](#_Toc9516261)

[Specifying that an Index is a Spatial Index 15](#_Toc9516262)

[Relationship Related 16](#_Toc9516263)

[Foreign Key Constraint Naming Conventions 16](#_Toc9516264)

[Specifying that a Relationship is Mandatory 16](#_Toc9516265)

[Specifying that a Relationship is an In Hierarchy Relationship (NASIS Specific) 16](#_Toc9516266)

[Specifying which Table is the "Favorite Child" for a Child Table with Siblings (NASIS Specific) 16](#_Toc9516267)

[Specifying that a Relationship is Traversable, and related Relationship Naming Considerations (NASIS Specific) 17](#_Toc9516268)

[Specifying the Cardinality of a Relationship 17](#_Toc9516269)

[Specifying when a Record in a Table can be inserted by Pasting a Record from a Related Table (NASIS Specific) 17](#_Toc9516270)

[TABLE COLLECTION TABLE 19](#_Toc9516271)

[TABLE COLLECTION HISTORY 21](#_Toc9516272)

[SYSTEM TABLE TABLE 22](#_Toc9516273)

[Specifying that a table should be Selectable" (NASIS Specific) 22](#_Toc9516274)

[Specifying that a Column should not be Editable in NASIS (NASIS Specific) 23](#_Toc9516275)

[Specifying that a Table is a NASIS Data Object (Editor) Root Table (NASIS Specific) 25](#_Toc9516291)

[Specifying that a Table is a Temporary Table 27](#_Toc9516386)

[Specifying that a "Table" should be created as a View 28](#_Toc9516387)

[Import Pedon Diagnostic Message SQL Statement 30](#_Toc9516455)

[SYSTEM TABLE TABLE HISTORY 33](#_Toc9516456)

[Attribute Related 34](#_Toc9516457)

[ATTRIBUTE TABLE 34](#_Toc9516458)

[Attribute Logical Name 34](#_Toc9516459)

[Specify the logical data type; current options are: 35](#_Toc9516460)

[Process for adding/changing logical data types. 36](#_Toc9516461)

[Attribute Longer Date Time Interval Qualifier (NASIS) 37](#_Toc9516462)

[Minimum 37](#_Toc9516463)

[Maximum 37](#_Toc9516464)

[ATTRIBUTE HISTORY 39](#_Toc9516465)

[TABLE COLLECTION HISTORY 40](#_Toc9516466)

[TABLE COLUMN TABLE 41](#_Toc9516467)

[Process for adding/changing physical data types. 41](#_Toc9516470)

[TABLE COLUMN TABLE HISTORY 48](#_Toc9516540)

[INDEX MASTER TABLE 49](#_Toc9516541)

[Specifying an Index as a Performance Index 50](#_Toc9516542)

[Specifying that an Index is a Spatial Index 50](#_Toc9516543)

[INDEX DETAIL 52](#_Toc9516544)

[INDEX DETAIL HISTORY 53](#_Toc9516545)

[TABLE COLUMN TABLE HISTORY 53](#_Toc9516546)

[COLUMN LOOKUP 55](#_Toc9516547)

[COLUMN LOOKUP HISTORY 57](#_Toc9516548)

[RELATIONSHIP MASTER 58](#_Toc9516549)

[Specifying that a Relationship is Mandatory 59](#_Toc9516550)

[Specifying that a Relationship is an In Hierarchy Relationship (NASIS Specific) 59](#_Toc9516551)

[Specifying which Table is the "Favorite Child" for a Child Table with Siblings (NASIS Specific) 60](#_Toc9516552)

[Specifying that a Relationship is Traversable, and related Relationship Naming Considerations (NASIS Specific) 60](#_Toc9516553)

[Specifying when a Record in a Table can be inserted by Pasting a Record from a Related Table (NASIS Specific) 61](#_Toc9516554)

[RELATIONSHIP MASTER HISTORY 63](#_Toc9516555)

[DOMAIN GROUP 64](#_Toc9516556)

[DOMAIN 64](#_Toc9516557)

[Domain Integrity 64](#_Toc9516558)

[Domain Documentation in the Data Model Repository 64](#_Toc9516559)

[Specifying that a Domain is Logically Ordered 65](#_Toc9516560)

[Specifying that both the Shorter and Longer Version of a Choice should be Displayed in a Domain's Choice List (NASIS Specific) 65](#_Toc9516561)

[*Customizable?* 66](#_Toc9516562)

[Specifying that a Domain is Customizable 66](#_Toc9516563)

[*Description* 66](#_Toc9516564)

[DOMAIN DETAILS 68](#_Toc9516565)

[DOMAIN DETAIL HISTORY 70](#_Toc9516566)

[DOMAIN HISTORY 70](#_Toc9516567)

[UNIT OF MEASURE 72](#_Toc9516568)

[There are a few General Principles when working in the NASIS Repo. 73](#_Toc9516569)

[Originating/Contact Person, Pending Action, Pending Status Conventions 74](#_Toc9516570)

[How to delete a System 79](#_Toc9516571)

[Steps for deleting an attribute without a domain. 80](#_Toc9516572)

[Steps for deleting an attribute with a domain. 80](#_Toc9516573)

[Deleting Domains 82](#_Toc9516574)

# General and NASIS Specific Data Modeling Conventions

Note: ***Required***means required by the database.

Note: ***Optional/Required*** means optional by the database but required by policy or convention.

Note that the term "**NASIS Specific**" encompasses the NASIS, Staging Server and Soil Data Warehouse data models. These three data models are identical to a large degree.

Most of this document describes/defines the columns in their respective tables. The sections are named like: **ATTRIBUTE TABLE** or **TABLE COLUMN TABLE** or **DOMAIN HISTORY**.

The descriptions/definitions will have the following format.

##### Table Collection Name

***Required***

The unique name for a table collection (business object). It is often the same as the name of the table collection's root table.

Some columns will have extra text having more explanation of their purpose.

Through page 14, there are explanations of various conventions and more in-depth discussion about a number of things.

Note: Many of the attributes and columns have several items that are either TRUE (YES) or FALSE (NO). Represented by a **Checkbox**. Many of the column labels are confusing because the default for all of these fields is FALSE. These are Boolean fields and are displayed as checkboxes. If it is not checked then you are to read it has 'NO' or 'FALSE'. Many times, the column label is a just one word followed by a '?', or it looks like a negative. Like 'Not Null'. The default is No, so record reads like No, it is Not Null. Meaning, of course, that it can be Null. Or, 'Visible?' where if you want to see the item in question there must be a checkmark in the box. With a checkmark, this field is read as 'Yes, it should be visible'. Otherwise it would read as 'It is not visible.' To change the state of the checkbox, you can either click the box with the mouse. Or, with that field having the focus you can press the Spacebar.

# General NASIS Logical and Physical Naming Conventions

Traditionally in NASIS, Logical Names include underscores, and Physical Names and Column Labels do not. At this point in time, I generally make a physical name the same as the corresponding logical name, minus the underscores. This convention is not verified by any QA query. The Column Label can be up to 80 characters in length and so it can be more specific than either the logical or physical name.

## System and Data Model Related

### Specifying the Data Model Name and Version

System Name/Data Model Name, and Data Model Version is specified in "System Name" and "System Version" fields in the "System" grid editor, e.g. "NASIS" and "NASIS 7.3.3".

### Specifying a System Sequence

The sequence in which systems/data models are displayed is controlled alphanumerically using the System Version column. By default, the data models are ordered by System and then by System Version. Where System Version is ordered using an alphanumeric sort, from least to most recent. The system/data model with the highest sequence number is the most recent.

## NASIS Data Object (Editor) Related

Note that NASIS and the Staging Server are our only two data models for which Data Objects/Editors are defined. Therefore, everything below is NASIS specifying that a NASIS Data Object should be Conditionally Visible in NASIS (NASIS Specific)

There is a checkbox in the System Model Editor grid editor, "Conditionally Visible?", for indicating that a NASIS Data Object should be conditionally visible, i.e. visible only to users who are a member of the NASIS Site identified. To make a NASIS Data Object conditionally visible, it appears that both "Visible in Grid Editor?" and "Conditionally Visible?" should be checked.

At the time this was written, the only conditionally visible NASIS Data Objects were "Domain Group", "System" and "Unit of Measure", the NASIS Data Objects that make up the NASIS Data Model Repository. Only NASIS users who are a member of NASIS Site "Soil Metadata Repository" can see these NASIS Data Objects.

### Specifying that a NASIS Data Object should be "Selectable for Replication" in NASIS (NASIS Specific)

There is a checkbox in the System Model Editor grid editor, "Selectable for Replication?" for indicating that a NASIS user should be able to select which instances of this NASIS Data Object should be downloaded to their local NASIS database.

### Specifying that a NASIS Data Object should be "AutoReplicated" to the Local NASIS Database (NASIS Specific)

*'(*

There is a checkbox in the System Model Editor grid editor, "Autoreplicate?", for indicating that all instances of the corresponding NASIS Data Object should be automatically downloaded to a NASIS user's local NASIS database.

A checkbox that indicates if the corresponding column should be included in the corresponding table's replication selection list.

### Specifying that a NASIS User should not be allowed to Insert Records into a Table (NASIS Specific)

There is a checkbox in the System grid editor, "No Insert?", for indicating that a NASIS user should not be allowed to insert records into the corresponding table. At the time this was written examples included the tables created to record an export request and all Forest Service related lookup tables.

### Specifying that a Table is Specific to a NASIS Client Database (NASIS Specific)

There is a checkbox in the System grid editor, "Client DB Only?", for indicating that a NASIS table should exist only in a local NASIS (client) database. At the time this was written, the only two such tables were "keyrangeclient" and "replicationparameter''.

### Specifying that a Table is Specific to the NASIS Server Database (NASIS Specific)

There is a checkbox in the System grid editor, "Server DB Only?", for indicating that a NASIS table should exist only in the central NASIS (server) database. At the time this was written, this included all tables whose name ends in "\_d", "\_I" and "\_x", and tables "contextsuffix", "keyrangeserver", "nasismessage" and "nasismessageuser''.

### Specifying the SQL Statement whose Output is Used as Part of an Import Diagnostic Message for a Table Whose Contents can be Imported from a Pedon PC Database (NASIS Specific)

There is a field, in the System grid editor, "IPDiagMsgSQL", for recording a query whose output is used as part of an import diagnostic message for a table whose contents can be imported from a Pedon PC database. This control should be populated for any table whose contents can be imported from a Pedon PC database.

### Specifying that a "Table" should be created as a View

The checkbox in the System grid editor, "View?", indicates that the corresponding table should be created as a View. At the time this was written, there are no views in any data model.

### Specifying that a Table is a Temporary Table

There is a checkbox in the System grid editor, "Temp Table?", for indicating that the corresponding table is a temporary table created only when needed. At the time this was written, the only such table is "icomponent" in the NASIS data model.

## Column Related

Most, but not all, of the concepts below are NASIS specific. Which concepts are NASIS specific will be explicitly indicated.

### Which Columns should typically be Included in a NASIS Data Object's Root Table? (NASIS Specific)

An integer primary key column whose default type is "Identity" (autonumber). That column's physical name is typically like "\*iid".

A foreign key column that identifies the NASIS site that owns that record. That column's physical name is typically like "\*dbiidref'.

A foreign key column that identifies the NASIS group that owns that record. That column's physical name is typically "grpiidref'.

A datetime column that indicates when that NASIS Data Object instance was last updated. That column's physical name is typically "objwlupdated".

A foreign key column that indicates which NASIS user last updated that NASIS Data Object. That column's physical name is typically "objuseriidref'.

A datetime column that indicates when the root table record of that NASIS Data Object instance was last updated. That column's physical name is typically "recwlupdated".

A foreign key column that indicates which NASIS user last updated the root table record of that NASIS Data Object instance. That column's physical name is typically "recuseriidref'.

### Which Columns should typically be Included in a NASIS Table that is not a NASIS Data Object's Root Table? (NASIS Specific)

An integer primary key column whose default type is "Identity" (autonumber). That column's physical name is typically like "\*iid".

An integer foreign key column identifying its "in hierarchy" parent table. That column's physical name is typically like "\*iidref', and its corresponding label is typically "Lineage".

A datetime column that indicates when a record in that table was last updated. That column's physical name is typically "recwlupdated".

A foreign key column that indicates which NASIS user last updated a record in that table. That column's physical name is typically "recuseriidref'.

Many non-root tables in NASIS include a column that allows an authorized user to explicitly sequence the child table records for the corresponding parent table record, because that column is also typically that table's primary sort key. That column's physical name is typically "seqnum".

### Specifying that a Column should be "Not Null"

There is a checkbox in the Table Column grid editor, "Not Null?", for indicating that non-null value for the corresponding column is required.

### Specifying that a Column corresponds to a Spatial Data Type

There is a checkbox in the Table Column grid editor, "Spatial Data Type?", for indicating when a column corresponds to a SOL Server spatial data type. For such a column, the corresponding physical data type should be "Geometry". When "Spatial Data Type?" is checked, text box control "SRID?" (spatial reference ID) on this same form should also be populated. For a spatial data type column, the SQL scripts used to create an instance of the corresponding database will include a related check constraint.

### Specifying that a Column should be Visible in NASIS (NASIS Specific)

There is a checkbox in the Table Column grid editor, "Visible?", for indicating that a column should be visible in its corresponding table in the NASIS interface.

Sometimes we leave an obsolete column in a table to preserve its final value but make that column non-editable and non-visible. Often just non-editable. An example are the HEL columns.

### Specifying that a Column should not be Editable in NASIS (NASIS Specific)

There is a checkbox in the Table Column grid editor, "Protected?", for indicating a column should not be user editable in its corresponding table in the NASIS interface. In general, the value of such columns is automatically provided by the NASIS application. The NASIS database also includes some columns that use to be editable, but no longer are. Typically, such a column is also hidden.

### Specifying that a Column's Value can be generated by a Calculation (NASIS Specific)

There is a checkbox in the Table Column grid editor, "Calculable?", for indicating that a column's value can be derived by a "calculation" script. At the time that this was written, the value in all but 8 of the 76 non-obsolete calculable columns could also be entered directly by a user. The non-obsolete calculable columns that cannot also be entered directly by a user are:

chtexturegrp.texture pedon.taxonkind

component.taxclname pedon.taxonname

ecologicalsite.ecositenm petaxhistory.taxclname

pedon.taxclname phorizon.texture

### Specifying which NASIS Data Object Root Table Columns should be Included in that NASIS Data Object Table's Replication Select List (NASIS Specific)

There is a checkbox in the Column Table grid editor, "Include in Rep Select List?", for indicating which columns in a NASIS Data Objects Root Table should be included in that NASIS Data Object's "Replication Select List". These are the columns whose value a NASIS user can specify in order to indicate which instances of that NASIS Data Object should be downloaded to the user's local NASIS database.

Note that this option does pertain to NASIS Data Objects whose contents are automatically replicated to a local NASIS database.

### Specifying that a Column has an Object Modify Default (NASIS Specific)

There is a checkbox in the Column Table grid editor, "Object Modify Default?", for indicating which columns in a NASIS Data Object's root table should have their default values automatically updated whenever any change is made to an instance of the corresponding NASIS Data Object. As well as which NASIS User made the change.

These two columns are typically the ID of the NASIS user who last modified an instance of any table in that NASIS Data Object, and the date and time when that change occurred.

### Specifying that a Column has a Record Modify Default (NASIS Specific)

There is a checkbox in the Column Table grid editor, "Record Modify Default?", for indicating which columns in a NASIS table should have their default values automatically updated whenever any change is made to an instance of the corresponding table.

These two columns are typically the ID of the NASIS user who last modified an instance of that table, and the date and time when that change occurred.

### Specifying Which Columns in one or more Related Tables should be displayed in lieu of a Foreign Key Column, aka "Column Lookups" (NASIS Specific)

We have come to, what is the most difficult and confusing task associated with the NASIS Data Model Repository - entering "Column Lookups". Let me start with an example.

In NASIS, a record in table "lmapunit" (legend map unit) always references a record in table "mapunit". Every record in table "lmapunit" stores a copy of the primary key of the corresponding record in table "mapunit". In table "lmapunit", the copy of the mapunit table's primary key is referred to as a "foreign key".

This "foreign key" is a meaningless integer value, so displaying that value in table "lmapunit" does not meaningfully identify the corresponding map unit to an end user. So in lieu of that foreign key, we display some columns from the corresponding record in table "mapunit" that hopefully meaningfully identify that map unit to an end user.

In the NASIS Data Model Repository, a person must somehow indicate which columns in a related table should be displayed in lieu of a foreign key (lineage). This is done by populating a metadata table we refer to as the "column lookup" table.

In the NASIS Data Model Repository, the name of this table is "Column Lookup". In the NASIS 6 Style Metadata standard, the name of this table is "MetadataColumnLookup".

### The Gory Details

Column lookups are entered by selecting a foreign key column from the from the Display Table Name column drop down choice list. Doing that will automatically select the other information needed. Let's look at the case where we want to display a set of columns in lieu of the foreign key to table "mapunit" in table "lmapunit".

In lieu of the foreign key to table "mapunit" (lmapunit.muiidref), we want to display the NASIS site that owns that related map unit, along with national map unit symbol and map unit name, in this order.

Then enter an integer to indicate what order of the "Display Sequence" indicates the sequence in which the lookup columns should be displayed.

In the choice list displayed to the NASIS user, by default, the corresponding column label, identified by control "Lookup Table Column Label" would be "NASIS Site Name", but we want to emphasize that this column corresponds to the NASIS site to which the related map unit belongs, therefore in the control labeled "Column Label Override", we enter "Mapunit NASIS Site Name".

Once the user has selected a map unit, we don't want "Mapunit NASIS Site Name" to actually be displayed in table "lmapunit", so "Choice List Only?" is checked.

For the next two lookups, we select "national\_mapunit\_symbol" and "mapunit\_name".

Note that the two lookup tables involved in this series of lookups are directly related via a single branch or pathway, i.e. "nasis\_site" to "mapunit". In other words, there is only "branch", and the first branch in a series of lookups is always branch zero.

A series of column lookups may reference more than one branch. See the column lookups for lmuaoverlap.lareaoviidref for an example.

Our current column lookup scheme replaces our original primitive scheme. The current column lookup scheme was created by Gary Spivak, and he is the one and only source for all questions regarding column lookups.

## Index, Primary Key, Foreign Key and Unique Constraint Related

The only thing in this section that is NASIS specific is the concept of a deferrable unique constraint.

### Index, Primary Key, Unique Constraint and Spatial Index Naming Conventions

For selected metadata entities, standard naming conventions are used. The metadata data entities for which standard naming conventions are used are indexes, primary key constraints, unique constraints and foreign key constraints.

### Duplicate and Performance Indexes

The name of every duplicate (non-unique) and/or performance index is in the form: DI\_correspondingtablephysicalname\_indexoddkey

Example: Dl\_mapunit\_18657

If a duplicate index is not a foreign key, it is typically a performance index.

### Primary Key Constraints

The name of every primary ken constraint is in the grid editor.: DI\_correspondingtablephysicalname

Example: PK\_mapunit

A table may have one and only one primary key constraint. In theory, every table should have a primary key, but there are rare, probably unjustified, exceptions to this rule.

### Unique Constraints

The name of every unique constraint is in the form: UC\_correspondingtablephysicalname\_indexoddkey Example: UC\_mapunit\_24933

A table may have zero or more unique constraints in addition to its primary key.

### Spatial Indexes

The name of every spatial index is in the form:

SI\_correspondingtablephysicalnam\_eindexoddkey Example: Sl\_mupolygon\_24876

A table won't have a spatial index unless that table includes a column whose SOL Server data type is "Geometry". A table may include more than one geometry columns, and therefore more than one spatial index.

### Primary Key Conventions

For all of our data models, our convention is that a table's primary key should always be a single column whose data type is some variant of "integer'' and whose default type is "Identity", which is SOL Server for "autonumber''. We do not use business oriented unique constraints as primary keys. All business oriented unique constraints should be implemented as an alternate unique constraint rather than as a primary key.

### Specifying a Table's Primary Key

There is a checkbox in the Index Master grid editor, "Primary Key?", for indicating that an index should serve as a table's primary key. In general, a primary key should be defined for every table in a database. For any primary key, the control labeled "Unique Constraint", on that same grid editor, should be set to "Unique". Why I labeled this control "Unique Constraint" is beyond me. There should probably be a "Unique Constraint" checkbox in lieu of this control.

### Specifying that an Index should be Clustered

There is a checkbox in the Index Master grid editor, "Clustered?", for indicating that an index should be clustered. In general, each table should have one clustered index. Typically, for a non-root table, a table's foreign key for its primary parent table is clustered. Susan McGlasson, our DBA, often tells me which index for a table should be clustered, based on performance considerations.

### Specifying an Index as a Performance Index

There is a checkbox in the Index Master grid editor, "Performance?", for indicating that an index is a "performance index". This checkbox exists merely as a reminder why the corresponding index exists to begin with. A performance index is almost always a duplicate (non-unique) index that is not also a foreign key.

### Specifying a "Deferrable Unique Constraint" (NASIS Specific)

There is a checkbox in the Index Master grid editor, "Deferrable?", for indicating that a unique constraint is "deferrable". This designation allows records that would violate a unique constraint to be pasted, so that a user can manually resolve any duplicates after a set of records has been pasted, but before the results of a paste operation can be saved. Obviously, all columns in a deferrable unique constraint must be user editable.

### Specifying that an Index Corresponds to a Foreign Key

There is a checkbox in the Index Master grid editor, "Foreign Key?", for indicating that the columns in an index represent a foreign key. The status of this checkbox has nothing to do with how the corresponding index is crated. This designation lets a person know when an index corresponds to a foreign key, and this designation is leveraged by one or more data model QA queries.

### Specifying that an Index is a Spatial Index

There is a checkbox in the Index Master grid editor, "Spatial Index?", for indicating that an index is a spatial index. This designation must be explicit because the process for generating the SQL to create a spatial index is different from the process for generating the SQL to create all other indexes. This designation controls which process creates the corresponding index.

## Relationship Related

Some, but not all, of the concepts below are NASIS specific. Which concepts are NASIS specific will be explicitly indicated.

### Foreign Key Constraint Naming Conventions

The name of every foreign key constraint is in the form: FKC\_childtablephysicalname\_parenttablephysicalname\_foreignkeyconstraintoddkey Example: FKC\_correlation\_mapunit\_3

Any table may potentially have multiple parent tables and any table may potentially have multiple child tables.

### Specifying that a Relationship is Mandatory

There is a checkbox in the Relationship Master grid editor, "Mandatory?", for indicating when a relationship is mandatory. A mandatory relationship is when a record cannot be added to a table that is a child in a relationship with another table, without having a record in the corresponding parent table. This is mostly a documentation issue, but this specification is also leveraged by one or more data model QA queries.

### Specifying that a Relationship is an In Hierarchy Relationship (NASIS Specific)

There is a checkbox in the Index Master grid editor, "In Hierarchy?", for indicating that the two tables identified by this relationship are part of the NASIS data model's directed acyclic graph (DAG). This attribute is used by the function that verifies if the tables involved in all relationships where this attribute is set conform to a directed acyclic graph. There are a number of data model QA queries related to this attribute.

### Specifying which Table is the "Favorite Child" for a Child Table with Siblings (NASIS Specific)

There is a checkbox in the Index Master grid editor, "Favorite Child?", for indicating which child table for a given parent table, is the "favorite child". A table cannot have more than one favorite child. In the NASIS interface, this determines which child table tab is the leftmost tab for the corresponding parent table. All tabs to the right of the favorite child table tab are in alphabetical order.

### Specifying that a Relationship is Traversable, and related Relationship Naming Considerations (NASIS Specific)

There is a checkbox in the Index Master grid editor, "Traverse?", for indicating if a relationship is "traversable". In NASIS, this means that the "Find Related" and "Load Related" functions are available for the two tables involved in that relationship.

Every relationship in the NASIS database has a name. When there is only one relationship between a given pair of tables, the default name of that relationship is "default".

If there is more than one relationship between the same two tables, each of those relationships must have a name other than "default", and one and only one of those relationship can be traversed via "Find Related" or "Load Related".

For example, each NASIS Data Object root table typically has two relationships with table "nasisuser''. One of those relationships corresponds to the NASIS user who last updated any table in that NASIS Data Object instance. For this case, the relationship name is always "object last updated by". The other relationship corresponds to the NASIS User who last updated that NASIS Data Object root table record instance. For this case, the relationship name is always "record last updated by".

### Specifying the Cardinality of a Relationship

There is a checkbox in the Relationship Master grid editor, "Cardinality?", for indicating the cardinality of a relationship. The allowable choices are:

One and Only One

One or More

Zero or More

Zero or One

This is mostly a documentation issue, but cardinality is verified by a data model QA queries.

### Specifying when a Record in a Table can be inserted by Pasting a Record from a Related Table (NASIS Specific)

There is a checkbox in the Relationship Master grid editor, "Paste?", for indicating when a record can be inserted into the parent table in a relationship by pasting a record instance from the child table involved in that relationship.

For example, "Paste?" is set for the relationship between tables "mapunit" and "lmapunit" (legend mapunit). One can insert a new record in table "lmapunit" by copying and pasting a record from table "mapunit".

# Description of each attribute in each table

#### TABLE COLLECTION TABLE

##### Table Collection Name

***Required***

The unique name for a table collection (business object). It is often the same as the name of the table collection's root table.

##### Table Collection Sequence

***Required***

The sequence in which table collections should be displayed.

##### Table Collection Insert Sequence

***Required***

The sequence in which the corresponding table collection should be inserted into the database, relative to other table collections.

***Restricted NASIS ID***

##### Optional

The ID of the NASIS site to which creation of new instances of the corresponding table collection is restricted. A NASIS site is an administrative entity by which table collection instance ownership is managed.

##### Visible When Restricted?

***Required***

A checkbox that indicates whether or not the corresponding table collection should be visible to users who cannot edit any table in that collection.

##### Load All?

***Required***

A checkbox that indicates whether or not all data for the root table in the corresponding table collection should be automatically loaded whenever that table is first viewed.

##### Visible in Grid Editor?

***Required***

A checkbox that indicates whether or not the table in in the corresponding table collection can be edited via the general NASIS grid editor. Some table collections have their own specialized edit interface.

Specifying that a NASIS Data Object should be Visible in NASIS (NASIS Specific)

There is a checkbox in the Table Collection Editor grid editor, "Visible in Grid Editor?", for indicating that a NASIS Data Object should be visible in NASIS.

At the time this was written, the only NASIS Data Objects that are not visible in NASIS are "Edit Setup", "Pedon DB Metadata", "Choice List Set", and "phorizon\_vnir\_scan\_raw\_data".

##### Table Collection Replication Type

***Required***

Indicates what type of replication is required. Either "autoreplication" or "selectable for replication"; selectable from a drop-down choice list for an instance of the corresponding NASIS object. Can be selected for inclusion in a NASIS user's replicate or if the corresponding object should be automatically replicated to all NASIS client databases.

***Customization Query***

##### Optional

A query that specifies which columns should be displayed, and the default sort order of records in a customizable table collection choice list. The query may include columns from related table collections.

***Customization Columns***

##### Optional

The record ID of the column whose values should be stored to indicate which records should be displayed for a customized table collection.

***Table Collection Id***

##### Optional/Required

An integer value that uniquely identifies a collection of tables in the NASIS-style metadata. A table can be a member of no more than one collection.

#### TABLE COLLECTION HISTORY

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### SYSTEM TABLE TABLE

##### Logical Name

***Required***

No description available.

##### Table Physical Name

***Required***

The physical (SQL) name of a database table.

##### Table Label

***Required***

A descriptive business-oriented name for a database table.

##### Help Text

##### Optional

No description available.

##### Soil Entity Type

##### Optional/Required

No description available. But it is needed to run some of the NREPO status checks.

##### Import Export File Name

***Required***

The base part of the file name of a table's associated ASCII pipe delimited import/export file. The complete name of a table's associated import/export file is the base name followed by the characters ".txt". For example, if the base name is " area ", the name of the associated import/export file is " area.txt". Usually the import export file name is the same as the table physical name. Must be unique within a database. This is used for SSURGO exports.

##### Visible?

***Required***

For a table, a checkbox that indicates whether or not the corresponding table should be visible in the NASIS grid editor. A table that is not visible in the NASIS grid editor should not be able to be selected as the target table for a NASIS query.

For a column, a checkbox that indicates whether or not the corresponding column should be visible in either the NASIS grid editor or one the special NASIS editors.

##### Selectable?

***Required***

### Specifying that a table should be Selectable" (NASIS Specific)

There is a checkbox in the System Table grid editor, "Selectable?", indicating that a NASIS Data Object should be selectable in NASIS.

This checkbox that indicates whether or not records from the corresponding table can potentially be created, read, updated or deleted in NASIS. A table can be selectable but not visible in the NASIS grid editor. Such a table is typically managed by a special editor in NASIS. A table that is not selectable cannot be referenced in a NASIS query.

##### Editable?

***Required***

### Specifying that a Column should not be Editable in NASIS (NASIS Specific)

There is a checkbox in the Column Table grid editor, "Protected?", for indicating a column should not be user editable in its corresponding table in the NASIS interface. In general, the value of such columns is automatically provided by the NASIS application. The NASIS database also includes some columns that use to be editable, but no longer are. Typically, such a column is also hidden.

For NASIS 7.3.3, the following tables were set to visible = false or

selectable = false or editable = false.

|  |  |  |  |
| --- | --- | --- | --- |
| Table physical name | Visible | Selectable | Editable |
| Contextsuffix | False | false | false |
| Customchoicedomain | False | true | false |
| Customchoicelistset | False | true | true |
| Customchoicelookup | False | true | false |
| Distcompmd | True | true | false |
| Distinterpmd | True | true | false |
| Distlegendmd | True | true | false |
| Distmapunitmd | True | true | false |
| Disttextmd | True | true | false |
| Editsetup | False | true | true |
| Editsetupelement | False | true | false |
| Editsetuptable | False | true | false |
| Formfavorites | False | true | false |
| Keyrangeclient | False | false | false |
| Keyrangeserver | False | false | false |
| Nasismessage | False | false | false |
| Nasismessageuser | False | false | false |
| Nasissession | False | false | false |
| pedondbcolumnmetadata | False | true | false |
| Pedondbtablemetadata | False | true | false |
| Phvnirscanrawdata | False | true | false |
| Queryfavorites | False | true | false |
| Replicationparameter | False | false | false |
| Reportcolumn | False | true | false |
| Reportfavorites | False | true | false |
| Ruleevalcomp | False | true | false |
| Rulerulecomp | False | true | false |
| Siteiristubeimage | False | true | false |
| Soilseries | True | true | false |
| Soilseriesmlrasusing | True | true | false |
| soilseriesstatesusing | True | true | false |
| soilseriestaxfamother | True | true | false |
| soilseriestaxmineralogy | True | true | false |
| soilseriesURL | False | false | false |

##### No Insert?

***Required***

A checkbox that indicates whether or not end users should be able to insert records into the corresponding table.

In NASIS 7.3.3, following tables where set to No Insert? = true:

|  |
| --- |
| Table physical name |
| Distcompmd |
| Distinterpmd |
| Distlegendmd |
| Distmapunitmd |
| Disttextmd |
| Ncsslayerlabdata |
| Ncsspedonlabdata |
| Usfseclevel |
| Usfsecoclass |
| Usfsectype |
| Usfsicat |
| Usfsinterp |
| Usfsirclass |
| Usfsirestrict |

##### Root Table?

***Required***

### Specifying that a Table is a NASIS Data Object (Editor) Root Table (NASIS Specific)

Each NASIS Data Object must have one and only one root table.

A checkbox indicates if the corresponding table is the root table of an object hierarchy. Each NASIS table collection must have one and only one root table. Root tables in the NASIS database require additional supporting tables, see “XXXX” for more details.

In NASIS 7.3.3, the following tables were set to root table = true:

|  |  |
| --- | --- |
| Table Collection Name | Table Physical Name |
| Area Type | areatype |
| Calculation | calculation |
| Choice List Set | customchoicelistset |
| Data Mapunit | datamapunit |
| Distribution | distmd |
| Domain Group | domaingroup |
| Ecological Site | ecologicalsite |
| Edit Setup | editsetup |
| Evaluation | evaluation |
| Form | form |
| Geomorph Feat Type | geomorfeattype |
| Legend | legend |
| Local Plant | localplant |
| Mapunit | mapunit |
| Milestone Type | milestonetype |
| NASIS Site | nasissite |
| NASIS User | nasisuser |
| NCSS Lab Data | ncsspedonlabdata |
| Other Veg Class Type | othvegclasstype |
| Pedon | pedon |
| Pedon DB Metadata | pedondbtablemetadata |
| Plant | plant |
| Project | project |
| Project Concern Type | projectconcerntype |
| Project Data Type | projectdatatype |
| Project Type | projecttype |
| Property | property |
| Query | query |
| Report | report |
| Rule | rule |
| Site | site |
| Site Association | siteassoc |
| Soil Series | soilseries |
| System | system |
| Tech Soil Service | techsoilservice |
| Tech Soil Serv Type | techsoilservicetype |
| Tech SS Prog Benefit | techsoilservprogbenefit |
| Transect | transect |
| Unit of Measure | Uom |
| Table Collection Name | Table Physical Name |
| USFS Eco Class Type | Usfsectype |
| USFS Interp Category | Usfsicat |
| USFS Interp Restriction | Usfsirestrict |
| USGS Quad Type | Usgsquadrangletype |
| Vegetation Plot | Vegplot |
| WS Import Map | Wsimportmap |

Depending on if a table is a root table or not a root table, certain columns are typically created, see “Which Columns should typically be Included in a NASIS Data Object’s Root Table? (NASIS Specific)” or “Which Columns should typically be Included in a NASIS Table that is not a NASIS Data Object’s Root Table? (NASIS Specific)“ for more details.

##### Temp Table?

***Required***

### Specifying that a Table is a Temporary Table

There is a checkbox in the System Table grid editor, "Temp Table?", for indicating that the corresponding table is a temporary table created only when needed. At the time this was written, the only such table is "icomponent" in the NASIS data model. The “NREPO -SQL Script – CreateTables” report ignores any temporary tables.

##### Aliased?

***Required***

A checkbox that indicates the table is created as an alias. The “NREPO - SQL Script – CreateTables” report ignores any aliased tables. Currently, the only database that has any aliases defined is the Staging Server. The “NREPO - SQL Script - CreateStagingAliases - create synonyms” report is used to create the Staging Server synonyms using the tables that are set as aliased. The list of aliased tables in the Staging Server should match the list of Staging Alias Required (NASIS) tables in NASIS.

As of Staging Server 7.3.3.1 following tables, using physical names, have Aliases.

|  |  |  |
| --- | --- | --- |
| Evaluation | Nasisuser | Rule |
| Nasisgroup | Property | Ruleevalcomp |
| Nasisgroupmember | Nasissite | Ruleratingclass |
| Nasissession | Nasissiteadmin | Rulerulecomp |

##### Static?

***Required***

A checkbox that indicates whether or not the table's contents are static.

In NASIS 7.3.3, the following tables, using physical names, are set as static tables:

|  |  |  |
| --- | --- | --- |
| Contextsuffix | Pedondbcolumnmetadata | Pedondbtablemetadata |

##### Client Only?

***Required***

For the NASIS database, a checkbox that, when set, indicates that the corresponding table is present only in a NASIS client database.

When this was written, there were only two tables that are set as client only, keyrangeclient and replicationparameter.

##### Server Only?

***Required***

For the NASIS database, a checkbox that, when set, indicates that the corresponding table is present only in the central NASIS server database.

In NASIS 7.3.3, the following tables, using physical names, are set as server only tables:

|  |  |  |
| --- | --- | --- |
| Contextsuffix | Nasismessage | soilseriesURL |
| Keyrangeserver | Nasismessageuser |  |

For the Staging Server database, a checkbox that, when set, indicates that the contents of the corresponding table should not be copied from the Staging Server database to a table with the same name in the Soil Data Warehouse database.

##### Created As View?

***Required (Currently not used)***

### Specifying that a "Table" should be created as a View

There is a checkbox in the System Table grid editor, "View?", for indicating that the corresponding table should be created as a View. At the time this was written, there was no such table in any data model.

NASIS code makes no distinction between tables and views, so this attribute is only relevant to the database creation script. The “NREPO - SQL Script - Create Tables” report ignores any views

##### Staging Alias Required?

***Required***

A checkbox indicates that the corresponding table in the staging database requires an alias.

A checkbox indicates that the corresponding table in the staging database requires an alias. This is used to document NASIS tables that are used directly in the Staging Server database. This is done by creating synonyms in the Staging Server database using NASIS as the source of the data.

In 7.3.3, the following tables, using physical names, are set as staging alias tables:

|  |  |  |
| --- | --- | --- |
| Evaluation | Nasis\_site | Rule |
| Nasisgroup | Nasissiteadmin | Ruleevalcomp |
| Nasisgroupmember | Nasisuser | Ruleratingclass |
| Nasissession | Property | Rulerulecomp |

##### Staging Counterpart Required?

***Required***

A checkbox indicates that a table of the same name and similar structure exists in the Staging Server database. This is used to document tables in NASIS that are also created in the Staging Server database.

A checkbox indicates that a table of the same name and similar structure exists in the staging database.

##### Pedon PC to NASIS?

***Required***

A checkbox that indicates that the table is included in the import from Pedon PC to NASIS.

***Note: Any changes to the list of Pedon PC to NASIS tables also requires changes to the Pedon PC application.***

When this was written, the following tables, using physical names, are set as Pedon PC to NASIS tables:

|  |  |  |
| --- | --- | --- |
| Pediagfeatures | Phdbcompliantcavity | Plotplantinventory |
| Pedon | Phdbcore | Site |
| Pefmp | Phdbcorereading | Siteaoverlap |
| Pehydricfieldindicator | Phdbscoop | Siteassoc |
| Peinfiltrationch | Phdbscoopreading | Siteassocsite |
| Peinfiltrationchdata | Phdesgnsuffix | Siteassocsoi |
| Peinfiltrationfh | Phfeatcolor | Siteassoctext |
| Peinfiltrationfhdata | Phfeatures | Sitebedrock |
| Peinfiltrationsummary | Phfmp | Siteecositehistory |
| pepenetrationresistance | Phfrags | Siteerosionacc |
| Perestrictions | Phhuarts | Sitegeomordesc |
| Pesoilstability | Phlabresults | Sitemuoverlap |
| Petaxhistfmmin | Phmottles | Siteobs |
| Petaxhistmoistcl | Phorizon | Siteobstext |
| Petaxhistory | Phpores | Sitepm |
| Petext | Phpvsf | Sitesoilmoist |
| Petxhistfmother | Phpvsfcolor | Sitesoiltemp |
| Phcemagent | Phrdxfeatures | Sitesurffrags |
| Phcolor | Phredoxfcolor | Sitetext |
| Phconccolor | Phtext | Transect |
| Phconcs | Phtexture | Transectestcomposition |
| Phcracks | Phtexturemod | Transecttext |
| Phdb |  |  |

### Import Pedon Diagnostic Message SQL Statement

A SQL statement used in the construction of an import Pedon diagnostic message for the corresponding table in a Pedon database. The statement returns the User ID and Rec ID of the Site, Transect or Site Association associated with the corresponding table in a Pedon database.

Some examples of the Import Pedon Diagnostic Message SQL Statements from NASIS 7.3.3 are shown in the following table.

|  |  |
| --- | --- |
| Table physical name | Import Pedon Diagnostic Message SQL Statement |
| Pedon | SELECT pedon.upedonid, pedon.peiid FROM pedon WHERE pedon.peiid=[Rec ID]; |
| Pefmp | SELECT pedon.upedonid, pedon.peiid FROM pedon INNER JOIN pefmp ON pedon.peiid = pefmp.peiidref WHERE pefmp.pefmpiid=[Rec ID]; |
| Site | SELECT site.usiteid, site.siteiid FROM site WHERE site.siteiid=[Rec ID]; |
| Siteaoverlap | SELECT site.usiteid, site.siteiid FROM site INNER JOIN siteaoverlap ON site.siteiid = siteaoverlap.siteiidref WHERE siteaoverlap.sareaoviid=[Rec ID]; |
| Siteassoc | SELECT siteassoc.usiteassocid, siteassoc.siteassociid FROM siteassoc WHERE siteassoc.siteassociid=[Rec ID]; |
| Transect | SELECT transect.utransectid, transect.tsectiid FROM transect WHERE transect.tsectiid=[Rec ID]; |
| transectestcomposition | SELECT transect.utransectid, transect.tsectiid FROM transect INNER JOIN transectestcomposition ON transect.tsectiid = transectestcomposition.tsectiidref WHERE transectestcomposition.tsectestcompiid=[Rec ID]; |
| Vegplot | SELECT site.usiteid, site.siteiid FROM (site INNER JOIN siteobs ON site.siteiid = siteobs.siteiidref) INNER JOIN vegplot ON siteobs.siteobsiid = vegplot.siteobsiidref WHERE vegplot.vegplotiid=[Rec ID]; |

##### DAG Level

##### Required

A table's level in the in the set of directed acyclic graphs that are defined for the corresponding data model. A root table corresponds to level 0. The DAG level is calculated using the relationship master data, so all indexes and relationship data must be entered before you can calculate the DAG level. The DAG calculations can be found in the Calculations/Validations Explorer Pane in the folder ‘System Table’. Use the following process to generate the proper DAG level data:

1. Select Calculations
2. Select System Table
3. Run calculation – DAG Level – 1. Delete Current DAG Levels
4. Run calculation – DAG Level – 2. Assign New DAG - The first time,

run this several times in a row, at least 5 times.

1. Select Reports under Soil Metadata Repository
2. Run both reports: ‘NREPO – DAG Level Missing’ and ‘NREPO – DAG Level Wrong’. Continue to run step 4 till these reports show no records.

You must calculate the DAG levels for all databases, the DAG level is what is used to determine the order for deleting the tables in the “NREPO -SQL Script – DropTables” report script.

##### Table Id

##### Required

An integer value that uniquely identifies a table in the NASIS 6 Style Metadata for all versions of a data model. The table ID must be unique within a specific data model. Using the Table ID allows the same ID to be used in the NASIS 6 Style Metadata for all versions of a data model. For example, the ‘area’ table ID of 256 has been used in all releases of NASIS starting with NASIS 5.1.

The Table IDs can be manually entered. This a very bad idea. To ensure that all Table IDs are unique, run the calculation named ‘Update Table ID where ID is null’ in the ‘System Table folder’ to populate null values. Running the calculation on the selected set ensures that all null Table IDs get populated.

##### Description

##### Optional/Required

No description available. While not strictly required by the system, it must be populated for future workers on the data model. While not strictly required, I've always required a table description before adding it to the data model.

##### SOL Advanced Options

##### Optional

To capture any SQL Server advance database/table options in the NREPO data.

#### SYSTEM TABLE TABLE HISTORY

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

## Attribute Related

The NASIS Repository makes a distinction between an “attribute” and a “column”. A column must be associated with one and only one attribute, but an attribute may be associated with one or more columns, in one or more tables. Attributes and tables are not directly related:

Attribute -> Column -> Table

Soil data includes both point data and what we refer to as “aggregated” data. Point data is data at a particular point on planet Earth. Pedon data is point data. Veg data is point data. Data about map units is considered “aggregated” data in that it is based in part on data from multiple points. Therefore, the data for many attributes is reported either as a range of values, or a range of values with a corresponding “expected value”. We refer to such an attribute as a “modal attribute”. Other terms include “high, low” or “high, low, representative value”, also known as “HL” and “HLRV”.

The same attribute is sometimes related to many columns. This allows us to record attributes at the attribute level that we don’t want to have to define for every column instance of that attribute.

In our data model repository, for a column associated with a HL or HLRV attribute, we record a single column, to cut down on the amount of duplicate data entry that would otherwise be required.

##### Any process that creates the scripts required to create a database instance of a data model programmatically expands columns associated with a modal attribute into multiple columns. The same expansion occurs whenever reports or metadata are created.

#### ATTRIBUTE TABLE

Attribute Logical Name

***Required***

The attribute logical name may contain a maximum of 30 characters. It may not contain any spaces. It is unique within a system.

##### Attribute Physical Name

***Required***

The attribute logical name may contain a maximum of 33 characters. It may not contain any spaces. It is unique within a system.

A column name should be as connotative as possible. Don’t choose a shorter name over a longer name is the expense of clarity. A column name should accurately reflect what a column records.

For a NASIS database only (NASIS the application as opposed to NASIS the project/staff), a column name may contain a maximum of 80 characters, but only when the corresponding column is modal (low/high

| Column Name | Purpose |
| --- | --- |
| sandtotal\_l | Records the low end of the range of total sand. |
| sandtotal\_ls | Indicates if the low end of the range of total sand was calculated or manually entered. |
| sandtotal\_r | Records the representative value of total sand. |
| sandtotal\_rs | Indicates if the representative value of total sand was calculated or manually entered. |
| sandtotal\_h | Records the high end of the range of total sand. |
| sandtotal\_hs | Indicates if the high end of the range of total sand was calculated or manually entered. |

or low/rv/high) and/or calculated. In NASIS, an extension is added to the name of a modal column, and for a calculated column, a companion column with its own extension, ‘\_s’ is created to indicate the source of the corresponding value. For example, in NASIS, attribute "sandtotal" is a low/rv/high column that may also be automatically calculated. Therefore, this attribute is represented by a total of six columns.

##### Attribute Label

##### Optional/Required

This column will not be included in 7.4.1.

***Logical Data Type***

***Required***

## Specify the logical data type; current options are:

|  |
| --- |
| Logical Data Types |
| Binary |
| Boolean |
| Calculation |
| Choice |
| Date/Time |
| Edit Setup |
| SQL Geometry |
| ESRI Geometry Line – no longer being used |
| ESRI Geometry Point – no longer being used |
| ESRI Geometry Polygon – no longer being used |
| Evaluation |
| File Reference |
| Float |
| GUID |
| Integer |
| Logical Data Types |
| Money |
| Narrative Text |
| Property |
| Query |
| Report |
| Rule |
| String |
| Hyperlink |
| Unknown |
| XML |

### Process for adding/changing logical data types.

Just adding a new logical data type to the domain\_detail table or making changes to the domain\_detail data is not sufficient to actually see these changes in NASIS because NASIS is metadata driven and the list of logical data types are coming from the MetadataDomainDetail table, not the domain\_detail table. There are two options for showing changes to the logical data types in NASIS: 1)Release a new version of NASIS which includes the domain\_detail changes or 2)Make the changes to the logical data type in the MetadataDomainDetail table on the NASIS server database, wait a day, then reinitialize your local NASIS client database. There is a nightly BCP process at NITC (DISC) that downloads all the metadata tables from the NASIS server database, which is used when a NASIS client data is reinitialized. The Domain\_ID value for logical data type is 737.

##### Field Size

##### For varchar and character data elements this is required, this is the size allocated for the data element. Usually it is the size of the Column Label. It needs to be populated for every attribute with a text-like Logical Data Type. Can be overridden with the column field size override column.

For attributes that have attribute logical data type = string, you can specify the length of the string. You would not specify a field size for columns that have a physical data type of varchar(max).

##### Attribute Precision

##### Optional

For attributes that have attribute logical data type = float, you can set the decimal precision, and for attributes that have column physical data type = datetime2, you can set the date precision. This is done in the Shorter Date Time Interval Qualifier.

##### Attribute Unit of Measure

##### Optional

Where appropriate, the unit of measured is determined by selecting the appropriate unit of measure from a drop down menu.

##### Domain Group Name

##### Optional

##### Domain Name

##### Optional

An integer number that uniquely identifies a domain in a particular database. When a domain group name is selected from the drop-down choice list there will also be the list of domains for that group.

Where appropriate, the Domain is determined by selecting the appropriate domain from a drop down menu. The logical data type must be set to “choice” in order to select a domain.

For attributes that have attribute logical data type = date/time, the attribute shorter date time interval qualifier is used to derive Date Time Precision in the NASIS 6 Style Metadata.

|  |  |  |  |
| --- | --- | --- | --- |
| Year | Day | Minute | Fraction |
| Month | Hour | Second |  |

##### **Attribute Longer Date Time Interval Qualifier (NASIS)**

##### Optional

No description available.

This is no longer used.

##### **Minimum**

##### Optional

The minimum value allowed. This value is required only where appropriate. A value must obviously be within the range of the corresponding data type. A floating point value should not be expressed where precision exceeds the corresponding precision value. It is permissible to provide a minimum without a corresponding maximum.

##### **Maximum**

##### Optional

The maximum value allowed. This value is required only where appropriate. A value must obviously be within the range of the corresponding data type. A floating point value should not be expressed where precision exceeds the corresponding precision value. It is permissible to provide a minimum without a corresponding maximum.

##### Case Sensitive?

***Required***

There is a checkbox in the System Attribute grid editor, "Case Sensitive?", for indicating when an attribute should be treated as "case sensitive".

This indicates that the data in the column must be searched in a case sensitive manner. Currently the only attributes that are case sensitive are the mapunit symbol and horizon name. When generating the SQL for the database, a collation of ‘Latin1\_General\_BIN’ is used to indicate a case sensitive column.

##### Official Definition

##### Optional/Required

A narrative description of each attribute. The official definition should make the purpose of this attribute clear to all potential audiences, which may include both end-users and developers. This data is stored in the NASIS 6 Style Metadata as the column description. While not officially required I don't add an attribute without one. I can't really conceive of an attribute with no Official Definition.

It is displayed when the user clicks on the View Info choice when right clicking on a column label and when right clicking on a table in the Tables Explorer.

##### Extended Definition

##### Optional

An expanded narrative description of each attribute. The extended definition should expand on the purpose and use this attribute. This data is stored in the NASIS 6 Style Metadata as the column help text.

##### Attribute Id

##### Optional/Required

An integer value that uniquely identifies an attribute. The attribute ID must be unique within a specific data model. Using the attribute ID allows the same ID to be used for all versions of a data model. For example, the ‘area name’ attribute ID of 4542 has been used in all releases of NASIS, from NASIS 5.1 to the current NASIS data model. Must be unique within each database. This field is automatically populated. Until the data is uploaded it may have a value of -1.

The Attribute IDs can be manually entered. This a very bad idea. To ensure that all Attribute IDs are unique, run the calculation named ‘Update Attribute ID where ID is null’ in the ‘Attribute folder’ to populate null values. Running the calculation on the selected set ensures that all null Attribute IDs get populated.

#### ATTRIBUTE HISTORY TABLE

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### TABLE COLLECTION HISTORY

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### TABLE COLUMN TABLE

##### Default Sequence

***Required***

No description available.

**Attribute Logical Name**

***Required***

Must be in the Attribute table.

**Attribute Physical Name**

***Required***

Must be in the Attribute table.

##### Attribute Label

##### Optional/Required

Is no longer available in 7.4.1

##### Attribute Logical Data Type

***Required***

Must be in the Attribute table.

##### Attribute Field Size

##### Optional/Required

Can only be edited in the Attribute table. Only populated when attribute data type is a string type***.***

Column Label

***Required***

A column label may contain a maximum of 80 characters. A column label may contain any valid printable character. Invalid characters are non-alphanumeric symbols, like %, >, etc. The column label must not start with a number. Do not include an underline connecting words.

##### ***Physical Data Type***

***Required***

##### Indicates the corresponding column's physical data type. The logical data type property is entered for the attribute. The logical data type and the physical data types should be compatible, for instance it would not make sense to select ‘datetime’ and the logical data type and then select ‘integer’ for the physical data type.

### Process for adding/changing physical data types.

Adding a new physical data type to the domaindetail table or making changes to the domaindetail data is not sufficient to see changes in NASIS because NASIS is metadata driven and the list physical data types are coming from the MetadataDomainDetail table. There are two options for showing changes to the physical data types in NASIS: 1)Release a new version of NASIS which includes the domaindetail changes or 2)Make the changes to the physical data type in the MetadataDomainDetail table on the NASIS server database, wait a day, then reinitialize your local NASIS client database. There is a nightly BCP process at DISC that downloads all of the metadata tables from the NASIS server database, which is used when a NASIS client data is reinitialized.

**NOTE:** New physical data types cannot be added to the NASIS database without making application changes.

***Column Display Size***

##### Optional/Required

The default display width of a column in spreadsheet view and reports, in characters.

***Type of Default Value***

##### Optional

The type of data to be used as the default value for a table column. Below is the current list of default values that can be selected.

|  |  |
| --- | --- |
| Choice Sequence | Default Type |
| 1 | current date/time |
| 2 | Literal |
| 3 | nasis group ID |
| 4 | nasis site ID |
| 5 | nasis user ID |
| 6 | Identity |
| 7 | zero length string |
| 8 | propagate from hierarchy parent |
| 9 | non-quoted literal |
| 10 | Propagated |

***Literal Default Value***

##### Optional

When literal is selected as the default type, enter the literal default value here.

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Column | Default Type | Default Value |
| Area | Obterm | Literal | 0 |
| Attribute | attcasesensitive | Literal | 0 |
| Calculation | Dataafuse | Literal | 0 |
| Chaashto | Rvindicator | Literal | 0 |
| Chtexturegrp | Stratextsflag | Literal | 0 |
| Columnlookup | Branch | Literal | 0 |
| Pedon | Pedrecorigin | Literal | NASIS |
| soilseriesmlrasusing | mlraareatypeiidref | Literal | 5 |
| soilseriesstatesusing | stateareatypeiidref | Literal | 3 |

##### Set Default on Row Change?

***Required***

A checkbox indicating if the corresponding column should be set to its default value when the corresponding record is updated.

This is a checkbox for indicating which columns in a NASIS table should have their default values automatically updated whenever any change is made to an instance of the corresponding table. The most common columns with this setting are typically the ID of the NASIS user who last modified an instance of that table, and the date and time when that change occurred.

Examples:

|  |  |
| --- | --- |
| Table Name | Column Name |
| area | recwlupdated |
| area | recuseriidref |
| areatext | recwlupdated |
| areatext | recuseriidref |
| areatype | recwlupdated |
| areatype | recuseriidref |

##### Set Default on Object Change?

***Required***

This is a checkbox for indicating which columns in a NASIS Data Object’s root table should have their default values automatically updated whenever any change is made to an instance of the corresponding NASIS Data Object. The most common columns with this setting are typically the ID of the NASIS user who last modified an instance of any table in that NASIS Data Object, and the date and time when that change occurred.

Examples:

|  |  |
| --- | --- |
| Table Name | Column Name |
| areatype | Objwlupdated |
| areatype | Objuseriidref |
| calculation | Objwlupdated |
| calculation | Objuseriidref |
| datamapunit | Objwlupdated |
| datamapunit | Objuseriidref |

##### Aggregation

***Required***

An integer number that uniquely identifies an attribute's ordinality (aggregation). Some attributes are recorded as a single representative value and some attributes are recorded as a high, low and representative value. An attribute's aggregation indicates how many physical columns a logical attribute resolves to.

|  |
| --- |
| **Aggregation Choices:** |
| High, Low |
| High, Low, Representative Value |
| Representative Value |
| List |
| List, Representative Value |
| None |

When the Attribute Logical Type is Choice then List or List, Representative Value are the aggregation choices.

**Not Null?**

***Required***

A checkbox in the Column Table grid editor, "Not Null?", indicating that non-null value for the corresponding column is required.

##### Calculable?

***Required***

A checkbox in the Column Table grid editor, "Calculable?", indicating that a column's value can be derived by a "calculation" script. At the time that this was written, the value in all but 8 of the 76 non-obsolete calculable columns could also be entered directly by a user. The non-obsolete calculable columns that cannot also be entered directly by a user are shown on the following page.

chtexturegrp.texture pedon.taxonkind

component.taxclname pedon.taxonname

ecologicalsite.ecositenm petaxhistory.taxclname

pedon.taxclname phorizon.texture

If the “Protected” checkbox is not checked, it means the data can also be entered directly by a user. If the “Protected” checkbox is checked, it prevents the user from directly entering the data. Columns that are set to “Calculable” have another column added to the database with a suffix of “\_s”. This column is used to indicate the source of the data, was it “calculated” or “manually entered”.

For calculated columns, the logical data type must also be set to “calculable”.

##### Visible?

***Required***

A checkbox in the Column Table grid editor, "Visible?", for indicating that a column should be visible in its corresponding table in the NASIS interface.

A checkbox for indicating which columns should be visible in either the NASIS grid editor or one of the special NASIS editors. There are two common reasons for not making a column visible: 1) to not show Foreign key iids, those that end in ‘iidref’ and 2) to hide an obsolete column from the application.

Sometimes we leave an obsolete column in a table to preserve its final value but make that column non-visible.

##### Protected?

***Required***

A checkbox in the Column Table form that indicates which columns should be prevented from being changed by end users. The following column columns are usually protected: 1) table record iids, those that end with “iid”, 2) foreign key iids, those that end with “iidref”, and 3) record and object when last updated columns.

##### Unique?

***Required***

A checkbox in the Column Table form that indicates if the corresponding column must be unique.

##### Business Column?

***Required?***

A checkbox indicating if the column contains soil business related data.

##### Sort Sequence

##### Optional/Required

No description available.

##### Sort Ascending

##### Optional

No description available.

##### Sort Type

##### Optional

No description available.

##### Alignment

##### Optional/Required

Describes how the display of data in a grid cell is aligned horizontally. This is non-editable field and is filled using a calculation.

##### In Replication Select List?

***Required***

A checkbox in the Column Table form that indicates if the corresponding column should be included in the corresponding table's replication selection list. This is used to determine which columns are displayed in the Local Database Setup module.

##### Include in Pedon PC?

***Required***

A checkbox in the Column Table form that the column is in the data model for Pedon PC.

##### FGDC?

***Required***

A checkbox that indicates whether or not this attribute is considered part of the Federal Geographic Data Committee soil metadata standard.

##### Set Server Default Type to Identify?

***Required***

A checkbox that indicates that the column should be defined as an identity in the server database.

##### Spatial Data Type?

***Required***

There is a checkbox in the Column Table grid editor, "Spatial Data Type?", indicating when a column corresponds to a SQL Server spatial data type. For such a column, the corresponding physical data type should be "Geometry". When "Spatial Data Type?" is checked, text box control "SRID?" should also be populated. For a spatial data type column, the SQL scripts used to create an instance of the corresponding database include a related check constraint.

##### SRID

##### Optional

The spatial reference ID is a number that identifies the datum and projection for spatial data.

##### File Content Column

##### Optional

the Column ID (MetadataTableColumn.ColumnlD) of the column used to store the contents of a file. This is an attribute of the corresponding column. The logical data type of the corresponding column must be "File Reference". A File Reference type column will display a file name in a table editor grid and will also include buttons labeled Open or Save. Open is used to import the contents of a file into the column identified by this attribute. Save is used to copy the imported file contents to a file whose name is stored in the File Reference column.

##### Override Field Size

##### Optional

Use this column to override the attribute field size. This allows the same attribute to have different field lengths, in different tables. This is primarily for varchar and char columns.

##### Help Text

##### Optional

No description available.

##### Column Id

##### Optional/Required

An integer value that uniquely identifies a column in a data model.

The Column IDs can be manually entered. This a very bad idea. To ensure that all Column IDs are unique, run the calculation named ‘Update Tablecolumn ID where ID IS NULL’ in the ‘Table Column folder’ to populate null values. Running the calculation on the selected set ensures that all null Column IDs get populated.

#### TABLE COLUMN TABLE HISTORY

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### INDEX MASTER TABLE

##### Constraint or Index Name

##### Optional/Required

The name that is used to physically implement an index in a database management system.

##### Unique Index?

***Required***

A checkbox in the System Index Master grid editor, "Unique Constraint", indicating that an index should be a unique index. "Unique" should be selected for any index that corresponds to either a primary key or an alternate unique constraint. Again, why I labeled this control "Unique Constraint" is beyond me. There should probably be a "Unique Constraint" checkbox in lieu of this control.

##### Foreign Key?

***Required***

A checkbox in the System Index Master grid editor, "Foreign Key?", indicating that the columns in an index represent a foreign key. The status of this

checkbox has nothing to do with how the corresponding index is crated. This designation lets a person know when an index corresponds to a foreign key, and this designation is leveraged by one or more data model QA queries.

##### Primary Index?

***Required***

A checkbox that indicates if the corresponding index is the primary index for the corresponding table.

##### Clustered Index

***Required***

A checkbox in the System Index Master grid editor, "Clustered?", indicating that an index should be clustered. In general, each table should have one clustered index. Typically, for a non-root table, a table's foreign key for its primary parent table is clustered. Susan McGlasson, our OBA, often tells me which index for a table should be clustered, based on performance considerations.

##### Index Deferable?

***Required***

A checkbox in the System Index Master grid editor, "Deferrable?", indicating that a unique constraint is "deferrable". This designation allows records that would violate a unique constraint to be pasted, so that a user can manually resolve any duplicates after a set of records has been pasted, but before the results of a paste operation can be saved. Obviously, all columns in a deferrable unique constraint must be user editable.

##### Performance Index?

***Required***

### Specifying an Index as a Performance Index

There is a checkbox in the System Index Master grid editor, "Performance?", indicating that an index is a "performance index". This checkbox exists merely as a reminder why the corresponding index exists to begin with. A performance index is almost always a duplicate (non-unique) index that is not also a foreign key.

##### Spatial?

***Required***

### Specifying that an Index is a Spatial Index

There is a checkbox in the System Index Master grid editor, "Spatial Index?", indicating that an index is a spatial index. This designation must be explicit because the process for generating the SQL to create a spatial index is different from the process for generating the SQL to create all other indexes. This designation controls which process creates the corresponding index.

##### Number of Columns

##### Optional

For the associated index, how many columns are part of the index.

##### Index Column Name

##### Optional

No description.

##### Index Description

##### Optional

For the associated index, the index description.

##### Index Id

##### Required

If null, there is a calculation that is run during QA that will populate the Index Id field.

An integer value that uniquely identifies an index. The index ID must be unique within each database.

The Index IDs can be manually entered. This a very bad idea. To ensure that all Index IDs are unique, run the calculation named ‘1-Update Indexmaster ID where ID IS NULL’ in the ‘Index Master folder’ to populate null values. Running the calculation on the selected set ensures that all null Index IDs get populated.

##### SQL Advanced Options

##### Optional

##### To capture any SQL Server advance database/table options in the NREPO data.

#### INDEX DETAIL TABLE

***Index Column Sequence***

***Required***

Specifies the sequence of a column in a table index.

##### Column Label

##### Optional/Required

The column label.

##### Column Physical Name

##### Optional/Required

The column (attribute) physical name.

***Include Clause Column?***

***Required***

Indicates that the column is to be added to the index in the include clause.

##### Not Null Column Clause?

***Required***

Indicates that the column is to be added to the index in the not null clause.

#### INDEX DETAIL HISTORY

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### COLUMN LOOKUP TABLE

It is hard to describe all the possibilities. The basic idea is just to identify which columns will be displayed for a foreign key. But the columns can come from the parent table or tables higher in the hierarchy, so the relationship must be specified for each table. There can be placeholder tables, meaning a table that doesn't have any display columns but is used to establish a relationship to a higher table. There can also be display columns from more than one parent path, which is what the branch attribute is used for.

Get a report to run to show all Column Lookups – use this as a baseline: NREPO-QA-Mod-Dep- G1 Column Lookups - Show All

##### Display Sequence

***Required***

The left to right sequence in which lookup columns in a related table should be displayed.

##### Branch

***Required***

An integer value used to distinguish on branch of a column lookup from another. Both branches must share the same relationship or set of relationships from the foreign key table to that table's drill down parent table, in the case of a drill down relationship, or to that table's immediate lookup table in the case of a lookup relationship.

##### Left Table Name

##### Optional

##### Left Index Name

##### Optional

##### Relationship Name

##### Optional

##### FKC Name

##### Optional

##### Display Table Name

##### Optional

##### Column Label

##### Optional

##### Display Column Name

##### Optional

The label that should be displayed for a column in a lookup choice list and when that column is displayed in lieu of a foreign key.

##### Column Label Override

##### Optional

***Display Only in Choice List?***

***Required***

A checkbox that indicates whether or not the corresponding column should be displayed in a choice list but not be displayed in lieu of the corresponding foreign key in a table once a selection has been made. This switch can be used to include an obsolete indicator in a choice list but exclude that column from being displayed in the related table itself.

#### COLUMN LOOKUP HISTORY TABLE

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### RELATIONSHIP MASTER TABLE

##### Right Table

##### Optional

##### Right Constraint or Index Name

##### Optional

##### Right Columns

##### Optional

***Relationship Name***

***Required***

A name given to a relationship between two tables in a relational database. If there is more than one relationship between the same two tables, the name of each of those relationships must be unique., otherwise use ‘default’ as the relationship name.

The relationship name is used for the "join by name" expression in a NASIS query, so the names should be meaningful. The NASIS query generator converts "join by name" into actual join conditions for the named relationship. If there is only one possible join between two tables, the name "default" is usually used.

***Fail on Delete?***

***Required***

A checkbox that indicates if the delete rule for the corresponding relationship is "fail". If the delete rule is not "fail", it is assumed to be "cascade".

***Cardinality***

***Required***

Indicates whether the relationship between the left table and right table is one to one or one to many. For a one to one relationship, a record in the left table is related to zero or one record in the right table. For a one to many relationship, a record in the left table is related to zero or more records in the right table. Cardinality does not indicate whether or not the relationship is mandatory.

The allowable choices are:

One and Only One

One or More

Zero or More

Zero or One

This is mostly a documentation issue, but cardinality is verified by a data model QA queries.

***Mandatory?***

***Required***

A checkbox that indicates if in order for a record to exist in the right table of a relationship, a corresponding record must exist in the left table of that relationship, i.e. mandatory = "yes". In other words, when mandatory is "no", a record may exist in the right table of a relationship without having a corresponding record in the left table of that relationship.

As far as I can tell this just means that the foreign key can be null.

### Specifying that a Relationship is Mandatory

There is a checkbox in the Relationship Master grid editor, "Mandatory?", for indicating when a relationship is mandatory. A mandatory relationship is when a record cannot be added to a table that is a child in a relationship with another table, without having a record in the corresponding parent table. This is mostly a documentation issue, but this specification is also leveraged by one or more data model QA queries.

***In Hierarchy?***

***Required***

A checkbox that when set, the two tables involved in the corresponding relationship are considered to be part of the same hierarchy. A hierarchy is a graph with a single root node where no branches ever converge. In a data model, a table may be a member of one and only one hierarchy. For the NASIS data model, MetadataTableCollection records all hierarchies that are defined. Other data models, like the one for the Staging Server, includes hierarchies but no table collections.

The drill-down operation in the table editor follows the in-hierarchy relationships. For example, the Legend Mapunit Area Overlap table is shown as a child of Legend Area Overlap and not Legend Mapunit, because the relationship between Legend Mapunit and Legend Mapunit Area Overlap is not in-hierarchy. It just provides a way of resolving multiple paths to the same table.

### Specifying that a Relationship is an In Hierarchy Relationship (NASIS Specific)

There is a checkbox in the Relationship Master grid editor, "In Hierarchy?", for indicating that the two tables identified by this relationship are part of the NASIS data model's directed acyclic graph (DAG). This attribute is used by the function that verifies if the tables involved in all relationships where this attribute is set conform to a directed acyclic graph. There are a number of data model QA queries related to this attribute.

***Favorite Child?***

***Required***

A checkbox that indicates if the right table in a relationship corresponds to the left table's "favorite child". When visiting the "children" of a table, the "favorite child" is the default table that will be visited when no explicit child table is specified.

### Specifying which Table is the "Favorite Child" for a Child Table with Siblings (NASIS Specific)

There is a checkbox in the Relationship Master grid editor, "Favorite Child?", for indicating which child table for a given parent table, is the "favorite child". A table cannot have more than one favorite child. In the NASIS interface, this determines which child table tab is the leftmost tab for the corresponding parent table. All tabs to the right of the favorite child table tab are in alphabetical order.

***Traverse?***

***Required***

A checkbox that indicates whether or not Load Related and Find Related are enabled for the corresponding relationship.

### Specifying that a Relationship is Traversable, and related Relationship Naming Considerations (NASIS Specific)

There is a checkbox in the Relationship Master grid editor, "Traverse?", for indicating if a relationship is "traversable". In NASIS, this means that the "Find Related" and "Load Related" functions are available for the two tables involved in that relationship.

Every relationship in the NASIS database has a name. When there is only one relationship between a given pair of tables, the d fault name of that relationship is "default".

If there is more than one relationship between the same two tables, each of those relationships must have a name other than "default", and one and only one of those relationship can be traversed via "Find Related" or "Load Related".

For example, each NASIS Data Object root table typically has two relationships with table "nasisuser". One of those relationships corresponds to the NASIS user who last updated any table in that NASIS Data Object instance. For this case, the relationship name is always "object last updated by". The other relationship corresponds to the NASIS User who last updated that NASIS Data Object root table record instance. For this case, the relationship name is always "record last updated by".

***Allow Paste?***

***Required***

### Specifying when a Record in a Table can be inserted by Pasting a Record from a Related Table (NASIS Specific)

There is a checkbox in the Relationship Master grid editor, "Paste?", for indicating when a record can be inserted into the parent table in a relationship by pasting a record instance from the child table involved in that relationship.

For example, "Paste?" is set for the relationship between tables "mapunit" and "lmapunit" (legend mapunit). One can insert a new record in table "lmapunit" by copying and pasting a record from table "mapunit".

***Constraint Deferable?***

***Required***

A checkbox that indicates if the corresponding constraint can be deferred on the client database, i.e. it is not created. Like deferrable UCs, we can also defer FKs in the local database because sometimes the parent record has not been downloaded.

***Foreign Key Constraint Name***

***Required***

The foreign key constraint name associated with the corresponding relationship.

***Relationship Id***

##### Optional/Required

An integer value that uniquely identifies a relationship between two tables in the NASIS-style metadata.

The RelationshipIDs can be manually entered. This a very bad idea. To ensure that all RelationshipIDs are unique, run the calculation named ‘1-Update Relationshipmaster ID where ID IS NULL’ in the ‘Relationship Master folder’ to populate null values. Running the calculation on the selected set ensures that all null RelationshipIDs get populated.

#### RELATIONSHIP MASTER HISTORY TABLE

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### DOMAIN GROUP

***Domain Group Name***

***Required***

For NASIS always use the Domain Group named: Current NASIS/SSURGO Domains

***Domain group description***

##### Optional

A narrative text description of a domain group. For NASIS the description should read: Current domains used with NASIS and corresponding SSURGO databases

#### DOMAIN TABLE

For NASIS, a domain is defined as a finite set of acceptable character strings. Although we do have domains where all choices represent either a valid integer or floating point value, we haven’t ever implemented a truly numeric domain. Some of our domains use what appear to be floating point numbers, they in fact are classes.

### 

### Domain Integrity

How we implement domain integrity varies from application to application. Here are some of the approaches we have used.

1. All domains are stored in a common table that an application uses to constrain choices during data entry. This is how the NASIS application implements domain integrity. The application is enforcing domain integrity, but a highly generalized approach is used.

2. A domain is implemented as a related "lookup" table, and referential integrity enforces domain integrity. There is a lot of this in the Soil Data Mart database. The proliferation of lookup tables can be annoying. I would also add it is very confusing to enter.

### Domain Documentation in the Data Model Repository

A common structure is used to document all domains in our data model repository. This scheme was originally developed for NASIS. Not all attributes that are defined may be necessary in all cases, but most domains can be documented using this scheme.

For a domain as a whole, we record the following:

***Domain Name***

***Required***

A domain name should provide a good indication of what a domain represents. For a domain that is associated with only one logical attribute, the domain name is often similar or exactly the same as the corresponding attribute logical name.

***Ordering***

***Required***

This attribute indicates if a domain is explicitly ordered or sorted ascending on Shorter String (see immediately below). These are the only two options currently supported in our data model repository.

***Ordered?***

***Required***

A checkbox indicating if the members of a domain can be logically ordered. When a domain can be logically ordered, that order must be specified using "Member sequence" (see immediately below), and the lowest value must always correspond to sequence number 1. In NASIS, when this value is set, the less than and greater than operators can be used in a query that includes the corresponding attribute. For Soil Data Viewer, when this value is set, a user may be allowed to change the default tie-break rule at runtime, when aggregation is performed.

### Specifying that a Domain is Logically Ordered

There is a checkbox in the System Choice List grid editor, "Ordered?", for indicating that a domain is logically ordered. An ordered domain is a domain where less than and greater than comparisons can be made between members of that domain. In the Soil Data Viewer application, for an attribute associated with a logically ordered domain, aggregation method "Minimum or Maximum" is allowed.

***Display Label?***

***Required***

When Shorter String (see immediately below) is used for data entry and displaying Shorter String alone in a choice list is not sufficient for the user to be able to make an informed selection, setting this value indicates that both Shorter String and Longer String should be displayed in any choice list for the corresponding domain. For some domains, the values in Shorter String are relatively short cryptic codes that require further explanation.

### Specifying that both the Shorter and Longer Version of a Choice should be Displayed in a Domain's Choice List (NASIS Specific)

There is a checkbox in the System Choice List grid editor, "Display Label?", for indicating that both the shorter and longer version of a choice should be displayed in a domain's choice list. The shorter version of a choice is referred to as "choice" or "choice value" and is typically in all lower case. The longer version of a choice is referred to as "choice label" and is typically in mixed case. The mixed case version of a choice is typically preferred in soil reports.

#### *Customizable?*

***Required***

A checkbox for indicating whether or not a particular domain is customizable by the NASIS user.

### 

### Specifying that a Domain is Customizable

There is a checkbox in the System Choice List grid editor, "Customizable", for indicating that a domain should be customizable, which means that a **NASIS** user can indicate which choices in a domain should and should not be displayed in a domain's choice list. **(NASIS** Specific)

#### *Description*

##### Optional/Required

A description of what the domain is describing. While not strictly required, I require a domain description before adding it to the data model.

***Domain Id***

##### Optional/Required

An integer value that uniquely identifies a domain in the Domain Group. In NASIS that Domain Group is named 'Current NASIS/SSURGO Domains'.

The Domain IDs can be manually entered. This a very bad idea. To ensure that all Domain IDs are unique, run the calculation named ‘Update Domain ID where ID IS NULL’ in the ‘Domain folder’ to populate null values. Running the calculation on the selected set ensures that all null Domain IDs get populated.

#### DOMAIN HISTORY TABLE

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### DOMAIN DETAIL TABLE

**Member sequence**

***Required***

At the current time, domain members must be either explicitly sequenced, or sort ascending on "shorter string". Members are sequenced, beginning at one.

**Member ID**

***Required***

Member ID is an integer value that unambiguously identifies a member of a domain. These values should be assigned sequentially, beginning with one. In NASIS, this is the value that is actually stored in the database in order to record a domain selection.

**Data Entry Text**

***Required***

Shorter string is an alphanumeric string containing 128 or fewer characters. For NASIS, this is the string that must be used for data entry. Since everything in NASIS is case sensitive, the string is usually in all lower case. Within a domain, this value must be unique.

***Label Text***

***Required***

Longer string is an alphanumeric string containing 254 or fewer characters. This tends to be a longer, more connotative, mixed case string. In a report, a domain selection is typically represented by this value. Within a domain, this value must be unique.

***Alternate Text Entry***

##### Optional/Required

It usually represents a code. Enter this value (if there is one) instead of what is entered in the Data Entry Text field.

***Description****.*

##### Optional/Required

When what a domain choice corresponds to cannot be inferred from either Shorter String or Longer String, a corresponding narrative text description is required. You can always choose to provide a description, for any reason. It is OK to provide a description for some members but not all members.

If there is no need to have both a shorter and longer version of an alphanumeric string, the same value can be recorded in both fields.

Should the occasion arise where the value used for data entry is not the shorter string, the data model repository will have to be updated to support this option.

***Obsolete?***

A checkbox indicating if the choice is used or is no longer used.

#### DOMAIN DETAIL HISTORY TABLE

***Date***

Optional/Required

No description available. Not required by either system settings or policy, but it only makes sense to record the date of any action in the History table. It sure would have helped me out.

***Notes***

Optional/Required

Notes describing decisions, issues, or other history related to the record. Not required by either system settings or policy, but it only makes sense to record the changes made to the records in the History table. It sure would have helped me out.

#### UNIT OF MEASURE TABLE

It is an Object and a Dynamic Domain. This table records the various units of measure in which values of attributes in a particular database are recorded. The unit of measure table is used by all data models.

***Symbol***

***Required***

The symbol for the unit of measure. It is all lower case.

***Units***

***Required***

The actual unit of measure. It is all lower case.

***SI?***

***Required***

A checkbox that indicates if the Unit of Measure is an SI unit of measure.

***Application***

***Required***

Despite what the md/em UOM describes as the application (mad dogs per english man application) this is supposed to be where the UOM is used. Such as kg: Added for sampling soil in the field. Will also be used to weigh plant materials for Ecological Site Inventory projects. Currently, only two of the UOM do not have anything in the field.

***Notes***

##### Optional/Required

Notes describing decisions, issues, or other history related to the record. While not strictly required please record notes about why this UOM was added to the table. Most of these fields are blank but should be populated.

***Obsolete?***

***Required***

A checkbox that indicates if a the UOM is obsolete.

***Unit of Measure Id***

***Required***

An integer value that uniquely identifies a unit of measure in a data model, not just within a table.

I don't really understand this because it is also the UOM position in the table.

## There are a few General Principles when working in the NASIS Repo.

Objects you will be working with when working in the NASIS Repo are:

* SYSTEM
* DOMAIN GROUP

System child tables you will be working with are:

* Table Collection
  + System Table
    - Table Column Table
      * Table Column History
    - Column Lookup Table
      * Column Lookup History
  + Index Master Table
    - Index Detail
      * Index Detail History
    - Index Master History
  + Relationship Master Table
    - Relationship Master History
  + System Table History
  + Table Collection History
* Attribute Table
  + Attribute History

Domain Group you will be working with.

* Domain
  + Domain Detail Table
  + Domain Detail History
  + Domain History

When I refer to the NASIS SYSTEM all the actions described are exactly the actions done in the Staging Server and the Soil Data Warehouse. There may be some actions that are different, they will be identified. The Soil Data Mart and the SSURGO date models will have some actions that are the same, some that are slightly different, and some that are not included in the Staging and SDW data models.

Since many attributes and columns are duplicated in other Systems it is a good idea to download the NASIS, Staging Server, SDW, SDM, and SSURGO into your local database and add them to your selected set.

If the column or attribute you are working on exists in other Systems it is a good idea to know which systems the column/attribute is used in. Use Find Related to do this.

NOTE: You will often find that you want to copy a record (attribute, column, their respective history) from one System to another. NASIS won' let you do this.

I like to copy a Table History record to multiple parent records in either the same parent table or other parent tables where the History is often generic enough that it can be used in other records of the same table or other parent tables in the same System. You will want to copy the history record and paste it into other records in the same table or other parent tables. That works like you'd expect.

Since the NASIS, Staging Server, and Soil Data Warehouse data models are for all intents and purposes duplicates you will want/need to make the same changes in the other systems. Including the History table records. So, you will copy the History table record and open another system and navigate to the table where you want to do the same action and record the same history. You won't be able to, bummer.

Basic rule is you can't copy in one system and paste in another system. This is a simple rule, but you will find that you have forgotten it when working in multiple systems.

Concerning Domain Groups.

All Systems must have one or more Domain Groups (unless, of course, there are no choice type columns) associated with the System. In very rare cases there may be more than one Domain Group used in a System. At the time this was written, there is only one Domain Group used in the NASIS SYSTEM. This Domain Group is named "Current NASIS/SSURGO Domains". Note, that when a System is downloaded into your local database it automatically brings along the default Domain Group; i.e. "Current NASIS/SSURGO Domains".

## Originating/Contact Person, Pending Action, Pending Status Conventions

The last four columns prior to the last set of Not Editable (grayed out) columns are where you can record information about who originated all the child tables and their child tables, etc., and the last change to any of the child tables the person to contact with questions about their metadata information and their child tables and the pending action or status of the last change to the system.

Not all tables will have these four columns. The tables with a red double strikethrough ~~through the name~~ do not have these four columns. The parent SYSTEM and the parent DOMAIN GROUP also do not have these four columns.

SYSTEM

* Table Collection
  + System Table
    - Table Column Table
      * Table Column History
    - Column Lookup Table
      * Column Lookup History
  + Index Master Table
    - Index Detail
      * Index Detail History
    - Index Master History
  + Relationship Master Table
    - Relationship Master History
  + System Table History
  + Table Collection History
* Attribute Table
  + Attribute History

DOMAIN GROUP

* Domain
  + Domain Detail Table
    - Domain Detail History
  + Domain History

The Originating Person column is generally the person who started the change for that table/attribute/column/etc. It has gotten to the point that this column has become less than useful because many of the folks in the column are no longer with us.

The Contact Person column is usually the Data Modeler at the time the add or change or delete was requested. I've kept that naming convention up.

The Pending Action is one of three types. Add, Change, or Delete.

Add means something new has been added to the data model, such as a new attribute, a new column, or a new table.

Change means that anything in the record where the Pending Action is change has been changed. Change could be anything from correcting a misspelling in the Help Text column or the Official Definition columns to a new Domain Main or Column Label. Basically, any editable field in the record, except for the four columns we are discussing here.

Delete means that the table record, attribute record, column record, index record, etc is going to be dropped from the data model. Delete Pending Action is really just a place holder to let me know that this record will not be in the next data model. One of the last changes to the data model being created is to actually delete the records with a Pending Action of Delete

The vast majoring of all Pending Actions are changes. All Pending Actions will be set to null in the next data model version.

A Pending Status is one of three types. Approved, Pending, Rejected.

Right of the bat, I've never used Rejected. I've never even seen Rejected in this column. So, that leaves Approved or Pending.

The vast majority of the values found/used in Pending Status is, well, pending.

As far as I know, I'm the only one that has ever used Approved. I use Approved when the Pending Action is Delete and only to let me know that whatever is being deleted is not used in SSURGO, for the most part. Approved is transient. It only stays around until I'm convinced whatever it is in reference to can be deleted won't affect any other kind of record.

For example, from just about the beginning of NASIS, there was an attribute named critical\_shear\_stress (aka TauC). It was added because at one time it was assumed that WEPP would need it. So, it was added to the NASIS/Staging/SDW data models. Several years ago it was decided that critical\_shear\_stress was not going to be used in WEPP, and so it was never implemented in NASIS beyond the stage of adding the attribute record and its corresponding column record. It was never added to the Soil Data Mart or the SSURGO data models. So, early on after I became the NASIS data modeler I made the Pending Action Delete and the Pending Status Pending. After a few years of making sure it wasn't ever going to be used and as we got closer to implementing a data model I changed the Pending Status to Approved and just recently I physically deleted critical\_shear\_stress record from the NASIS/Staging/SDW 7.4.1 data model.

As we get closer to actually implementing a new SSURGO data model, I will make more of the pending deletes approved and finally deleted from the NASIS/Staging/SDW/SDM data models.

As mentioned above, Change is the vast majority of all Pending Actions; in addition, Change and a Pending Status of Pending is the most widely seen of any of the records except those that have a null Pending Action and a null Pending Status.

Just before we send the data model to our DBA in FC we delete all those records with the Pending Action of Deleted and Pending Status of Pending or Approved. After the deletes are processed we set all Pending Actions to null and all Pending Statuses to null. This is so we can start with a blank slate with the following data model version.

Adding columns to the data model.

For those columns with only one value (i.e. no choices), the general principle is to first add the attribute the column is based on. At the same time, enter the Originating Person, Contact Person, Pending Action, and Pending status.

Columns with a choice list follow a different path. In general, if the column is a choice type column; first create the new Domain in the Domain table in the Domain Group, if a domain does not already exist. If a domain does exist, go to the next step. Now go back to the Attribute table. Then create the attribute for that column, if an attribute doesn't already exist in the System. NOTE: When adding an Attribute or a Column you MUST be In the table you want to add the attribute/column to. This attribute will be linked to the column to be added in a later step. Now, enter the domain to be used when populating this new attribute from the drop down list in the Domain Name column.

Now you can create the column you want to add. The first action you need to do is to select the attribute, from the drop down list, you want linked to the new column. Finish the record. Don't forget to add the Originating Person, the Contact Person, the Pending Action, and the Pending Status.

In general, to delete an attribute/column, this process is reversed. Delete the Domain, Delete the Column, then Delete the Attribute. Of course, it is not as simple as it looks.

In the Table Column table:

Deleting a column is straight forward. First write down what attribute the column is linked to using the Find Related icon. Select the column's record and click on the Delete Icon in the editor grid. Done.

If you also want to delete the attribute that was linked to the column you just deleted you need to open the attribute table, usually by clicking on the Attribute tab. Go to the attribute table, again you must be in the table you want to delete the column from,

One of the conventions followed in working with the NASIS Metatdata Repository, aka NASIS Repo, is that all changes to a column or attribute must be completed and recorded before doing any deletion operation. Meaning, the Pending Action column of the affected System child table is set to Delete and the Pending Status column of the same record must be set to Approved.

Adding columns to the data model.

For those columns with only one value (i.e. no choices), the general principle is to first add the attribute the column is based on. At the same time, enter the Originating Person, Contact Person, Pending Action, and Pending status.

Columns with a choice list follow a different path. In general, if the column is a choice type column; first create the new Domain in the Domain table in the Domain Group, if a domain does not already exist. If a domain does exist, go to the next step. Now go back to the Attribute table. Then create the attribute for that column, if an attribute doesn't already exist in the System. NOTE: When adding an Attribute or a Column you MUST be In the table you want to add the attribute/column to. This attribute will be linked to the column to be added in a later step. Now, enter the domain to be used when populating this new attribute from the drop down list in the Domain Name column.

Now you can create the column you want to add. The first action you need to do is to select the attribute, from the drop down list, you want linked to the new column. Finish the record. Don't forget to add the Originating Person, the Contact Person, the Pending Action, and the Pending Status.

In general, to delete an attribute/column, this process is reversed. Delete the Domain, Delete the Column, then Delete the Attribute. Of course, it is not as simple as it looks.

In the Attribute table:

Deleting a column is straight forward. First write down what attribute the column is linked to using the Find Related icon. Select the column's record and click on the Delete Icon in the editor grid. Done.

If you also want to delete the attribute that was linked to the column you just deleted you need to open the attribute table, usually by clicking on the Attribute tab. Go to the attribute table, again you must be in the table you want to delete the column from.

One of the conventions followed in working with the NASIS Metatdata Repository, aka NASIS Repo, is that all changes to a column or attribute must be completed and recorded before doing any deletion operation. Meaning, the Pending Action column of the affected System child table is set to Delete and the Pending Status column of the same record must be set to Approved.

## How to delete a System

Make sure the System you want to delete is the only System in your selected set and that it is checked out.

NOTE: This process is to be used only to delete a test Systems. DO NOT USE to delete official Systems.

It takes two general steps. First, delete all the records in all the child tables of the System\_table. Second, delete the System, itself.

Deleting the system data isn't as straightforward as selecting a table and clicking the Delete icon. There are six steps that you need to take, in the order listed below, to delete the system data.

1. Open the Relationship Master table.

Select All

Click the Delete Icon

Upload Changes

1. Open the Column Lookup table.

Select All

Click the Delete Icon

Upload Changes

1. Open the Index Master table.

Select All

Click the Delete Icon

Upload Changes

1. Open the Table Column table.

Select All

Click the Delete Icon

Upload Changes

1. Open the Attribute table.

Select All

Click the Delete Icon

Upload Changes

1. Open the System Table table.

Select All

Click the Delete Icon

Upload Changes

Deleting the System itself is easy.

1. Open the System table

Select the System

Click the Delete Icon

Upload the changes

The System is no longer in the National Database

## Steps for deleting an attribute without a domain.

You may want to delete an attribute from the System. You need to find all the columns that the attribute is linked to because the column(s) must be deleted first.

1. To delete an attribute, you need to find all the columns that it is linked to because the column(s) must be deleted first.
2. All the attributes you want to delete need to have the Pending Action column set to Delete and the Pending Status column set to Approved.
3. All the columns you want to delete need to have the Pending Action column set to Delete and the Pending Status set to Approved.
4. I know this seems like a waste of time. But, you were supposed to have identified all the attributes/columns that you want to delete in the entire system before starting the actual deletion of the attribute/column.
5. First, understand that the Attribute table is unique to a System.
6. The system must be checked out
7. Open the System Table table
8. Open the Table Column table
9. Open the Attribute table from the Table Explorer or click on the Attribute tab of the System.
10. Do a Find Related and select Table Column as the target table
11. Click on the attribute you want to delete. You will get an error if every column linked to that attribute isn't deleted first. To get around this, the only System in your selected set is the system you want the attribute to be deleted from. To delete attributes and columns from other systems, you must go through the same process with each of the other systems, one-by-one.
12. When deleting an attribute or column from a table, make comments in the system\_table\_history table of the attributes/columns deleted because this is the only place such a record will exist after the attribute or column is actually deleted from the data model. This will help future data modelers get a better understanding of what was done and why.

## Steps for deleting an attribute with a domain.

You may want to delete an attribute with a domain from the System. You need to find all the columns that the attribute is linked to because the column(s) must be deleted first.

*Attributes must be deleted before Domains can be deleted. You will recognize that this is the reverse of the way to add an Attribute. You don't need to worry about the Domain deletion part, because you probably won't be allowed to the Domain anyway. More on that later.*

1. To delete an attribute you need to find all the columns that it is linked to because the column(s) must be deleted first.
2. All the attributes you want to delete need to have the Pending Action column set to Delete and the Pending Status column set to Approved.
3. All the columns you want to delete need to have the Pending Action column set to Delete and the Pending Status set to Approved.
4. I know this seems like a waste of time. But, you were supposed to have identified all the attributes/columns that you want to delete in the entire system before starting the actual deletion of the attribute/column.
5. First, understand that the Attribute table is unique to a System.
6. The system must be checked out
7. Open the System Table table
8. Open the Table Column table
9. Click on the Attribute tab of the System Table
10. Click on the attribute you want to delete
11. Do a Find Related and select Table Column as the target table
12. NOTE: Find Related only works on the data in your selected set.
13. If more than one column is linked to the attribute you won't be allowed to delete the attribute.

You will get an error saying that the attribute is in use. You need to find each column the attribute is linked to and delete that column either from the system you are working in or from other systems linked to that attribute. You can do one of two things to get around this error. 1) remove the other Systems from your local database (uploading and checking in as necessary). Or, 2) Remove the column from all the systems in your local database. This means loading them into your selected set, checking them out, deleted the columns, recording a history note in the Table Column table containing that column. Uploading the changes and checking them in.

1. When deleting an attribute or column from a table, make comments in the system\_table\_history table of the attributes/columns deleted because this is the only place such a record will exist after the attribute or column is actually deleted from the data model. This will help future data modelers get a better understanding of what was done and why.

The last thing you do is to try to delete the domain used. I say try because you won't be able to. Since the Current NASIS/SSURGO Domains Domain Group is the default Domain Group associated with every system associated with NASIS. You will be able to delete the row in the checked out Domain Master, but you will get an Upload Conflict when you upload changes.

This is because the Domain Group is in use by other Systems, even if they are not in your selected set or even in your local database. All they need to be is in existence. So, basically, for all intents and purposes once you add a domain to the Current NASIS/SSURGO Domains domain group it is in there forever.

## Deleting Domains

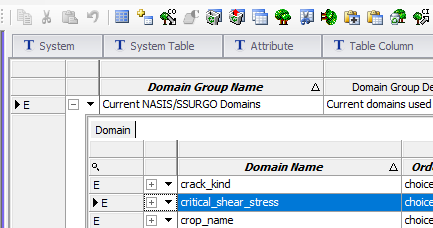
Remember that Domains are the first thing that must be created when creating an attribute (data element) that uses a choice list – or where the value is not continuous. And, thus are the last to be removed when deleting a column. There are several attributes where the values look like floats or integers but are in fact discontinuous. A famous example is soil\_erodability\_factor\_rf. With values like .02, .05, .10, and .15 it looks like a float with precision of 2. In reality, these are class values and are choices rather than what it looks like; continuous. Then there are those domains that appear to be continuous, such as t\_factor with values of 1, 2, 3, 4, or 5 that really are continuous.

For those attributes that are tagged as being obsolete and are to be deleted, many have domains. Such as the many Wildlife Habitat attributes. These attributes also exist in SSURGO so we can't just delete them. We have to wait until an updated SSURGO data model is ready and then delete the Wildlife Habitat attributes and their respective columns. Since all of these have a choice list, named Wildlife\_Rating and is linked to the Wildlife Habitat attributes.

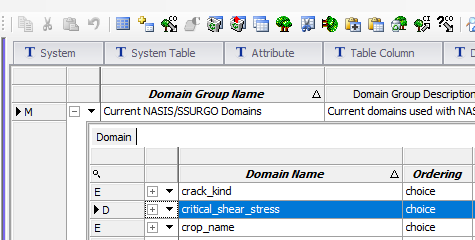
Since, according the data element creating process and its close cousin, deleting a data element domains are the last thing deleted. When deleting a domain, it is necessary to identify those attributes that it is linked to. As an example, let's see what happens when trying to delete the domain linked to the attribute named critical\_shear\_stress.

As pointed out above, NASIS/Staging/SDW data models are for all intents and purposes identical you need to delete the critical\_shear\_stress attribute and column from all three data models. Since we are working with data model version 7.4.1 imagine that all three of these data models version 7.4.1 have been added to your local database and then to your selected set, and have been checked out so they can be edited.

Imagine attribute critical\_shear\_stress (aka TauC) is going to be deleted. It is a choice list type of data element and its domain is named "critical\_shear\_stress". Imagine that the critical\_shear\_stress column, which has the column label of TauC, has been deleted. Then the attribute critical\_shear\_stress is deleted. Now it is time to delete the domain with the name of critical\_shear\_stress. Highlighting that record in the Domain Master table

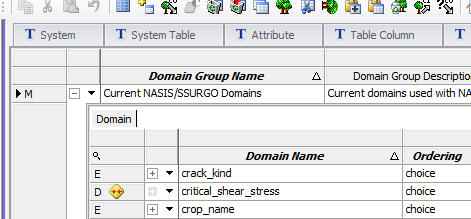


And then, clicking the Delete (empty trashcan) icon



Then uploading the change (along with all other changes that have been made)

You would get the following:



And there would be the following message in the Status Messages pane:

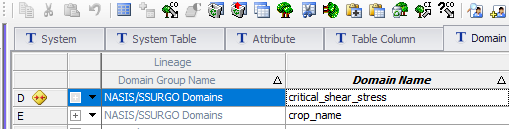
Upload completed with conflicts.

Now, to see what the conflict is you open the Upload Conflicts tab and see:

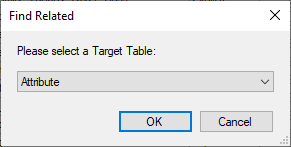
Explorer Table Record Conflict Type

Tables Domain 12339 Cannot delete. Record in attribute table refers to this row

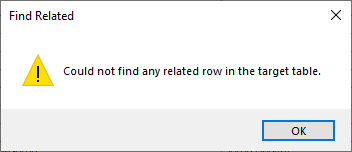
Clicking 12339 hyperlink transports you to:



Then if you did a 'Find Relate' operation you get:



Since the only choices here are 'Attribute', 'Domain Group', and 'NASIS User' accept the default; 'Attribute' and click the OK button and get:



By now, you are probably confused. Because, you know for a fact that you deleted the critical\_shear\_stress attribute and column from all three 7.4.1 data models.

What you need to remember is, that like almost all soil scientists, the soil scientists involved in the data modeling of NASIS never throw anything away.

If you were to query for 'NASIS%' in the !UTIL – LOAD System by System Version query which is in the Soil Data Repository query folder, you would see about 30 data models beginning with "NASIS", if you were to re-run the query using 'Staging%' and again using 'SDW%' you would see additional 32 SDW versions and 11 additional Staging Server data model versions.

Remember that critical\_shear\_stress is in NASIS, Staging, and SDW. So it is also in about 29 other SDW data models and 10 other Staging Server data models.

And remember that the Domain Group used by NASIS is also used by all other NASIS, Staging, and SDW data models.

Thus, in order to really delete the critical\_shear\_stress domain, you would have to go into each of the other 40 some data models and remove critical\_shear\_stress attributes and columns.

Only after all of that can you really get to delete the critical\_shear\_stress domain. Most of the other data model versions are official data model versions that were used in earlier NASIS databases (including all three data models). Meaning you would be going back changing official databases. Changing official and by now historical documents is wrong.

What does this boil down to? Essentially, domains can **NEVER** be deleted from NASIS.

# Q/A of the NASIS Data Model Process

There are a series of calculations that need to be run that populate columns the user doesn't populate during the editing of a data model.

These calculations are located in the Soil Metadata Repository folder.

There are a great many Q/A reports that need to be run and analyzed and responded to before the FC DBA can be notified that we are ready to pass off the new data model for their part in the creation and deployment of a new data model.

These Reports are all located in the Soil Metadata Repository folder.

## Calculations

The calculations must be run in the following order.

Do these calculations by System with only one System in the selected set.

### Attribute folder calculation

UPDATE ATTRIBUTE ID WHERE ID IS NULL

Populates Attribute ID column when it is NULL.

### Domain folder Calculation

UPDATE DOMAIN ID WHERE ID IS NULL

Populates Domain ID column when it is NULL.

### Index Master Id folder Calculations

1-UPDATE Indexmaster ID WHERE ID IS NULL

Populates Indexmaster ID column when it is NULL.

2-Generate Column: constraintindexname

Generates the Column named: constraintorindexname for those

columns that were missing.

3-Update indexcolumn & indexcolunames

Updates indexcolnum (Number of Columns) and Index Column

Names.

1-UPDATE Indexmaster ID WHERE ID IS NULL

Populates Indexmaster ID column when it is NULL

### Relationship Master Calculations

1-UPDATE Relationshipmaster ID WHERE ID IS NULL

Populates the relationshipmaster ID column when it is NULL

2-Generate Column: foreignkeyconstraintname

Generates the Column named: foreignkeyconstraintname for

those columns that they were missing

### System Table calculations

DAG Level – 1. Delete Current DAG Levels

Deletes current DAG Level

DAG Level - 2. Assign New DAG

Generates DAG Level

Follow instructions given in this calculation

Update Table ID where ID is null

Updates Table ID where it is null

### Table Column Calculations

Assign Alignment

Populates Alignment column where Alignment is null

Update Tablecolumn ID where ID IS NULL

Populates Column ID where it is null

### Unit of Measure Calculation

Update Unit of Measure ID where ID IS NULL

Populates Unit of Measure Id when it is null

## Reports