

NASIS Project

Database/Data Model

Change Control Guide

June 2019

[Introduction 4](#_Toc10706605)

[Database/Data Model Change Control Guidelines 4](#_Toc10706606)

[Requesting Database/Data Model Changes 5](#_Toc10706607)

[General Guidelines 5](#_Toc10706608)

[Requesting Changes other than a New Data Model or Changes to an Existing Data Model 6](#_Toc10706609)

[Requesting a New Data Model or Changes to an Existing Data Model 6](#_Toc10706610)

[Data Model Specific Guidelines 7](#_Toc10706611)

[Required Information 7](#_Toc10706612)

[Optional Information 7](#_Toc10706613)

[Table Specific Guidelines 7](#_Toc10706614)

[Required Information 7](#_Toc10706615)

[Optional Information 8](#_Toc10706616)

[What information is established for you? 8](#_Toc10706617)

[What kinds of information related to this database object cannot be documented in our data model repository at this time? 9](#_Toc10706618)

[Column Specific Guidelines 9](#_Toc10706619)

[Required Information 9](#_Toc10706620)

[Optional Information 11](#_Toc10706621)

[What kinds of information related to this database object cannot be documented in our data model repository at this time? 11](#_Toc10706622)

[Current Database/Data Model Conventions 11](#_Toc10706623)

[General Conventions 11](#_Toc10706624)

[Nomenclature Conventions 12](#_Toc10706625)

[General Conventions 12](#_Toc10706626)

[Databases/Data Models 13](#_Toc10706627)

[Tables 13](#_Toc10706628)

[Table Physical Name 13](#_Toc10706629)

[Table Label 13](#_Toc10706630)

[Columns 14](#_Toc10706631)

[Column Physical Name 14](#_Toc10706632)

[Column Label 14](#_Toc10706633)

[Defaults 14](#_Toc10706634)

[Primary Key Constraints 15](#_Toc10706635)

[Unique Constraints other than a Primary Key Constraint 15](#_Toc10706636)

[Duplicate Indexes 15](#_Toc10706637)

[Foreign Key Constraints 16](#_Toc10706638)

[Domains 16](#_Toc10706639)

[Domain Integrity 16](#_Toc10706640)

[Domain Documentation in the Data Model Repository 17](#_Toc10706641)

[What is the data model repository and how is it used? 18](#_Toc10706642)

[Data Model Versioning 18](#_Toc10706643)

[NASIS Report Scripts to Create Objects in a New Database 19](#_Toc10706644)

[What categories of database related information cannot currently be documented in our data model repository? 20](#_Toc10706645)

# Introduction

We continue to strive to improve our process for managing database evolution. The last documentation was written in 2007 by Dorn Egley who is retired from NRCS. This document is one of several that attempt to update the formal written guidelines. This document now enumerates the guidelines and conventions that pertain to logical database design and database change control.

# Database/Data Model Change Control Guidelines

For the following discussion, let’s define three different kinds of databases, "production", "sanctioned development" and "private".

A production database is one that is currently being used by a production information system.

A "sanctioned development" database is a sanctioned instance of a particular database, used for development, where "sanctioned" means that the only people who can make changes to such a database are members of the NASIS project’s Database Administration Team. A sanctioned development database is typically shared by multiple developers, and sometimes multiple subprojects.

A "private" database is an instance of a particular database that is exclusively used by a particular developer.

This document refers to the NASIS project’s Database Administration Team, where "NASIS" includes all soil related information systems, e.g. the NASIS application itself, Staging Server, Soil Data Warehouse, Soil Data Mart, Soil Data Access, Web Soil Survey, etc. At the time this document was originally written, that team included Susan McGlasson, Paul Price and Dorn Egley. At this time, 3 Jun 2019, Paul and Dorn are no longer with NRCS.

The guidelines that follow pertain only to production and sanctioned development databases. You are free to do whatever you want with a private database. If you want to create a private database, you can do so by running the SQL create scripts for the appropriate version of the database in question. See section "NASIS Report Scripts to Create Objects in a New Database" for further details.

**Guideline:** Each ongoing project (NASIS, SDM, WSS, etc.) must designate a database coordinator for each database that is referenced by any code under the auspices of that project. That person should be the sole point of contact between that project’s developers and the NASIS project’s Database Administration Team. That person needs to be aware of all database related changes for each database for which they are designated to be the coordinator.

**Guideline:** The only people authorized to create or make a change to a production database are members of the NASIS project’s Database Administration Team. Any such request should come from the project’s coordinator for that database and must be added to the COLAB tracker for the corresponding data model.

**Guideline:** The only people authorized to create a new instance of a sanctioned development database are members of the NASIS project’s Database Administration Team. Any such request should come from the project’s coordinator for that database and must be added to the COLAB tracker for the corresponding data model.

**Guideline:** The only people authorized to make a change to a sanctioned development database are members of the NASIS project’s Database Administration Team. Any such request should come from the project’s coordinator for that database and must be added to the COLAB tracker for the corresponding data model.

**Guideline:** The only people authorized to create a new data model or change an existing data model are members of the NASIS project’s Database Administration Team. Any such request should come from the project’s coordinator for that database and must be added to the COLAB tracker for the corresponding data model. Any request for a new data model or a change to an existing data model must include the information documented in the section titled "Requesting Database/Data Model Changes".

**Guideline:** Sometime prior to the release of a version of an information system or application, the NASIS project’s Database Administration Team will verify that each database instance referenced by that information system or application is completely in sync with its corresponding data model in the data model repository. Exactly when this is done is negotiable, but this should obviously be done sometime prior to beta testing.

# Requesting Database/Data Model Changes

## General Guidelines

For a variety of reasons, we have decided to manage database/data model change control independent of application change control. To do this we have established a COLAB project for this purpose, and the name of that project is "NRCS-Soils-DB-CC".

We have also decided to track changes by data model rather than by database instance. To do this we have established a separate tracker for each data model. We have also established a separate document hierarchy for data model.

Change requests can be divided into two categories:

1. Change requests related to a database instance.

2. Change requests related to a specific data model.

Both types of requests are made by adding a tracker item for the corresponding data model. For a database change request, tracker field "Request Type" should be set to "Database". For a data model change request, tracker field "Request Type" should be set to "Data Model".

All database/data model change control tracker items are automatically sent to each member of the NASIS project’s Database Administration Team.

In the case where you want a change made to a specific data model, and then want that change to immediately be reflected in one or more database instances, the data model change request should be logged separately from the database change request(s).

## Requesting Changes other than a New Data Model or Changes to an Existing Data Model

Here are examples of this kind of change request:

You want a new instance of a database.

You want an existing instance of a database updated to reflect recent data model changes.

You want the test data in a particular sanctioned development database instance to be updated.

You want to verify that an instance of a database is in sync with the current data model for that database.

For such a request you need to remember to provide all pertinent cluster or server names, database/data model names, data model version and any other information needed by NASIS project’s Database Administration Team.

*For this type of change request, tracker field "Request Type" should be set to "Database".*

## Requesting a New Data Model or Changes to an Existing Data Model

For a new data model, the NASIS project’s Database Administration Team will have to create a new tracker before any change requests can be logged. For a request for an entirely new data model, please send an e-mail that includes all required information to [George.Teachman@usda.gov](mailto:ailto:ailto:15m@yR@FTERg!t2019)

For an existing data model, the change request must be entered into the existing tracker for that data model.

*For this type of change request, tracker field "Request Type" should be set to "Data Model".*

### Data Model Specific Guidelines

#### Required Information

You must explain the purpose of a new database/data model, and why you think a new database/data model is required, as opposed to adding tables to one or more existing data models/databases.

#### Optional Information

You may propose a name for the new database/data model.

### Table Specific Guidelines

#### Required Information

You are required to provide the following information for each table.

##### Table Description

You must provide a narrative description of each table. This description should make the purpose of this table clear to all potential audiences, which may include both end-users and future developers.

##### Table Physical Name

You must propose the physical name of a new table. See the section titled "Nomenclature Conventions" for further details.

##### All Required Columns and Supporting Information

The information that is required for each column is discussed in the section titled "Column Specific Guidelines".

##### All Unique Constraints

For each table, all unique constraints must be enumerated, and for each unique constraint, you must provide the physical name of every column participating in that constraint. The absence of any unique constraint will have to be justified.

##### All Relationships to other Tables

For each relationship this table has with any other table, the following information must be provided:

1. The physical name of the related table.

2. The physical name of the column or columns on which the two tables are joined. Keep in mind that our convention is to use the same column name for any pair of columns on which two tables are joined.

*Note that for any relationship, the table in which the foreign key resides is considered to be the "child" or "dependent" table.*

3. The "on delete" rule. The delete rule indicates what should happen when an attempt is made to delete a record in the non-dependent table that has one or more corresponding records in the dependent table. Allowable options are "fail" (no action) or "cascade". Option "clear" (set to null) is not currently supported.

##### All Duplicate Indexes Required for Performance

For each table, all duplicate indexes required for performance must be enumerated, and for each duplicate index, you must provide the physical name of every column participating in that index.

#### Optional Information

##### Table Label

You may propose a table label. A table label is typically a variant of the table physical name, where most, if not all abbreviations and acronyms are completely spelled out, with intervening spaces. See the section titled "Nomenclature Conventions" for further details. In NASIS, table labels appear in the user interface. Table labels are included in our metadata reports, and we sometimes use table labels in lieu of table physical names in a data model diagram.

#### What information is established for you?

Which unique constraint should serve as the primary key will be determined by the NASIS project's Database Administration Team.

All constraint and index names will be provided for you. This includes primary key constraint names, unique constraint names, index names and foreign key constraint names.

#### What kinds of information related to this database object cannot be documented in our data model repository at this time?

At the current time, we cannot document multicolumn check constraints in the data model repository.

At the current time, table permissions are not documented in the data model repository.

At the current time, triggers cannot be documented in the data model repository.

### Column Specific Guidelines

#### Required Information

You are required to provide the following information for each column.

##### Column Description

You must provide a narrative description of each column. This description should make the purpose of this column clear to all potential audiences, which may include both end-users and future developers.

##### Column Sequence

This value corresponds to a column’s logical sequence in its corresponding table. In general, we try to keep logical and physical sequence in sync.

For a relational table, column order isn’t supposed to matter. That notwithstanding, we do tend to group columns by some scheme that makes sense to us, and that scheme can vary from table to table. Sometimes our rationale is pretty thin.

##### Column Physical Name

You must propose the physical name of a new column. See the section titled "Nomenclature Conventions" for further details.

##### Physical Data Type

Any valid SQL Server data type may be specified. All of the required parameters for the corresponding data type must also be provided.

##### Field Size

The value is required only when it is required as part of the SQL Server data type specification.

##### Precision

This value is required only for floating point attributes. This value corresponds to the number of decimal places to the right of the decimal point. The minimum allowable value is zero.

##### Null/Not Null

You must indicate whether or not a column is required.

##### Minimum Allowable Value/Maximum Allowable Value

These values are required only when appropriate. Any value must obviously be within the range of the corresponding data type. A floating point value should not be expressed with precision that exceeds the corresponding precision value. It is permissible to provide a minimum without a corresponding maximum, and vice versa.

##### Domain Constraints

If a new column’s values need to be restricted to a fixed set of ASCII values, you must provide the corresponding domain. See the section titled "**Error! Reference source not found.**" for further details.

##### Default Type/Value

This information is required only when appropriate. Default type must be one of the following:

|  |  |
| --- | --- |
| **Default Type** | **Comment** |
| Current Date/Time |  |
| Literal |  |
| NASIS Group ID | This is something that only pertains to the NASIS data model. |
| NASIS Site ID | This is something that only pertains to the NASIS data model. |
| NASIS User ID | This is something that only pertains to the NASIS data model. |
| Propagate from Hierarchy Parent | This is something that only pertains to the NASIS data model. |
| Sequence |  |
| Zero Length String |  |

For a literal default, the literal default value must be provided. For a sequence default, we will assume that the sequence should start at one and increment by one, unless you indicate otherwise.

##### Units of Measure

This value is required only for a column whose values are meaningless without a corresponding unit of measure.

##### Column Case Sensitivity Requirement

You must indicate if a column needs to be treated as case sensitive. At the time this was written, our only case sensitive attribute among all of our data models was map unit symbol.

#### Optional Information

##### Column Label

You may propose a column label. A column label is typically a variant of the column physical name, where most, if not all abbreviations and acronyms are completely spelled out, with intervening spaces. See the section titled "Nomenclature Conventions" for further details. In NASIS, column labels appear in the user interface. Column labels are also included in our metadata reports.

#### What kinds of information related to this database object cannot be documented in our data model repository at this time?

At the current time, we cannot document column specific constraints other than minimum and/or maximum allowable value. We cannot current accommodate pattern matching validation strings or expression check constraints.

At the current time, we cannot accommodate a column whose value is derived by an expression.

# Current Database/Data Model Conventions

## General Conventions

In general, each data model should be third normal form, although we don’t go through any formal process to determine this. Exceptions need to be justified.

Table and column names should provide a good indication of what that table or column records. Names should be well thought out.

Within a data model, column names must be logically unique. In other words, there should not be two columns in different tables with the same name where any other attribute of that column varies (data type, field size, precision, minimum, maximum, domain, description, etc.).

A column shouldn’t be too generically defined, .e.g. "SequenceNumber". A column name that requires one definition in one context and a slightly different definition in another context should be separated into two different columns.

In general, a Boolean attribute should be implemented as data type "Bit". In SSURGO, it should be a small int to ensure the negative value of 0 is allowed.

In general, a primary key constraint must be defined for each table. A table's primary key should be a single column surrogate key, a serial integer value. The name of the primary key column should be based on the name of the corresponding table. Business oriented unique constraints (natural keys) should never serve as a primary key.

In general, a unique constraint or duplicate index, as appropriate, should be defined for each foreign key in a table.

In general, a clustered constraint or index must be defined for each table. For a table that corresponds to a feature class, ESRI adds its own clustered index, so we don’t get to specify which constraint or index for that table should be clustered. For a table that contains one or more foreign keys, a constraint or index corresponding to one of those foreign keys should be clustered. Which foreign key is selected should be based on how data in that table is most commonly accessed.

In general, all relationships should be exclusively based on surrogate keys. The update of a non-key field in a table should never require the update of a migrated instance of that field in a related table. The name of a foreign key column should be the same as the name of the corresponding primary key column.

Barring a legitimate reason to the contrary, all referential integrity should be enforced at that database engine level.

## Nomenclature Conventions

### General Conventions

In general, the only characters allowed in a name are letters, upper and lower case, digits, spaces and underscores. A name should begin with a letter. In general, the following ASCII characters should not be used in Column Names: !,@,#,$,%,^,&,\*,<,>,|,\. The reason is that these characters cause problems when they exist in Column Names in NASIS Forms. Spaces are only permitted in the name of something that doesn’t correspond to a database object, e.g. table and column labels. Underscores are only permitted in logical names, domain names, constraint names and index names.

To make names more readable, our current convention is to use a variant of camel casing. The first letter of a word or truncated word should be capitalized, and the remainder should be in lower case. In general, acronyms and abbreviations should be in all upper case. When name length is an issue, certain words can be truncated, abbreviated, or even represented by a single capital letter.

Exceptions to these conventions will be noted where appropriate.

For the NASIS database, we originally defined both a physical name and a logical name for each table and column. The NASIS database was originally implemented in Informix, and at that time Informix allowed a maximum of 18 characters in a physical name, and many of our physical names were pretty cryptic. In our newer SQL Server databases we no longer make a distinction between logical and physical names, but this convention will likely be retained for NASIS, even when it is implemented in SQL Server because in queries, or scripts based on the NASIS CVIR language, either logical or physical names can be used, and changing either of these names would break a lot of existing queries, calculations, validations and reports.

### Databases/Data Models

A database name/ data model system name may contain a maximum of 128 characters. A database name may not be a SQL Server reserved keyword. A data model (system) version may contain a maximum of 30 characters and may not be a SQL Server reserved keyword.

### Tables

#### Table Physical Name

A table name may contain a maximum of 33 characters. A table name may not contain any spaces or underscores. An exception is made for system tables name\_l (locks), name\_d (downloads), and name\_x (deletes). A table name may not be a SQL Server reserved keyword.

A table name should be as connotative as possible. Don’t choose a shorter name over a longer name at the expense of clarity. A table name should accurately reflect what a table records. In general, table names should be singular rather than plural, e.g. "mapunit" rather than "mapunits".

#### Table Label

A table label may contain a maximum of 80 characters. A table label may contain any valid printable character. Table label conventions are the same as noted in the General Conventions. It is important to note the first character must be a letter.

### Columns

#### Column Physical Name

In NASIS, there is no column logical or physical name. They are the its respective attribute physical name and logical name.

In NASIS, an extension is added to the physical name of a modal column, and for a calculated column, a companion column with its own extension 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:

| **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. |

#### Column Label

A column label may contain a maximum of 80 characters. Column label conventions are the same as noted in the General Conventions. It is important to note the first character must be a letter. It was recently discovered, when we added the NASIS Forms functionality, that in addition to no number as the first character there are several special characters that must not be used in a column name. These include, !, @, #, $, %, ^, &,\*, <,>,/,\, and |.

### Defaults

A default name may contain a maximum of 128 characters. All default names must be of the form:

DF\_tablephysicalname\_tablecolumnname

where "tablephysicalname" and "tablecolumnname" are the physical names of the corresponding table and column.

*All defaults must be named, but all default names are automatically generated.*

### Primary Key Constraints

A primary key constraint name may contain a maximum of 128 characters. All primary key constraint names must be of the form:

PK\_tablephysicalname

where "tablephysicalname" is the physical name of the corresponding table.

*All primary key constraint names are automatically generated.*

### Unique Constraints other than a Primary Key Constraint

A unique constraint name many contain a maximum of 128 characters. All unique constraint names must be of the form:

UC\_tablephysicalname\_nasisdatamodelrepositoryindexkey

where "tablephysicalname" is the name of the corresponding table, and "nasisdatamodelrepositoryindexkey" is the non-varying key ([Index Master].[idxoddkey]) of this constraint in the NASIS data model repository.

*All unique constraint names are automatically generated.*

### Duplicate Indexes

A duplicate index name may contain a maximum of 128 characters. All duplicate index names must be of the form:

DI\_tablephysicalname\_nasisdatamodelrepositoryindexkey

where "tablephysicalname" is the name of the corresponding table, and "nasisdatamodelrepositoryindexkey" is the non-varying key ([Index Master].[idxoddkey]) of this index in the NASIS data model repository.

*All duplicate index names are automatically generated.*

### Foreign Key Constraints

A foreign key constraint name may contain a maximum of 128 characters. All foreign key constraint names must be of the form:

FK\_referencingtablephysicalname\_referencedtablephysicalname\_nasisdatamodelrepositoryforeignkeyconstraintkey

where "referencingtablephysicalname" is the name of table in which the common join column(s) corresponds to a foreign key, "referencedtablephysicalname" is the name of the table for which a unique constraint must be defined for the common join column(s), and "nasisdatamodelrepositoryforeignkeyconstraintkey" is the non-varying key ([Index Dependencies].[depoddkey]) of this foreign key constraint in the NASIS data model repository.

*All foreign key constraint names are automatically generated.*

## Domains

A domain name may contain a maximum of 60 characters. A domain name may contain underscores but may not contain any spaces.

A Domain Detail Data Entry Text may contain up to a maximum of 80 characters.

A Domain Detail Label Text may contain a maximum of 248 characters.

For our project, 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. The integer or floating point value represent classes and not a continuous range.

### 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.

One of the schemes we haven’t yet pursued is the implementation of domain integrity using the capabilities of SQL.

### 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:

1. Domain ID. Domain ID is an integer value that unambiguously identifies a domain.

2. Domain name. 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.

3. Ordering. 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.

4. Ordered? When this Boolean value is set, it indicates that 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.

5. Display label? 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.

For each member of a domain, we record the following:

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

2. Member ID. 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.

3. Data Entry Text. 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.

4. Label Text. 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.

5. Description. When what a domain choice corresponds to cannot be inferred from either Data Entry Text or Label Text, a corresponding narrative text description is required. You can always choose to provide a description, for any reason. It is rarely OK to provide a description for some members and is not OK for all members.

If an application has 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.

# What is the data model repository and how is it used?

## Data Model Versioning

For most data models, the corresponding version number is represented by two or three integer values separated by periods. Such as NASIS 7.3.3 or SSURGO 3.0. All but the last integer value corresponds to the major version, and the last integer value corresponds to the minor version. We strive to have the first number of the NASIS version (7.x.x) correspond to the first number of the NASIS Client version (7.0.3.11713).

For a database/data model that is used by only one information system, the major version is typically the same as the version of the corresponding information system. For a database/data model that is used by multiple information systems, such as the Soil Data Mart database/data model, major version is independent of any information system that accesses that database. For such a database, when to increment major version can be a pretty arbitrary decision.

Over time we have been migrating to a scheme where any change to a data model typically results in a new minor version. We try to avoid multiple minor versions for a given day, but it is not uncommon to have multiple minor versions for a given week.

## NASIS Report Scripts to Create Objects in a New Database

When we generate SQL scripts to create objects in an existing database, there are always five separate scripts, each in their own file. These reports are found in the Soil Metadata Repository Report folder.

|  |  |
| --- | --- |
| **Report Name** | **Purpose** |
| NREPO - SQL Script - CreateTables | This script creates all tables and columns. |
| NREPO - SQL Script - CreatePrimaryKeyConstraints | This script creates all primary key constraints. |
| NREPO - SQL Script - CreateUniqueConstraints | This script creates all unique constraints that are not also a primary key constraint. |
| NREPO - SQL Script - CreateDuplicateIndexes | This script creates all duplicate indexes. |
| NREPO - SQL Script - CreateForeignKeyConstraints | This script creates all foreign key constraints. |

CreateTables has to be run first, and CreatePrimaryKeyConstraints has to be run before CreateForeignKeyConstraints, but apart from that, order of execution doesn’t matter.

The output from these reports are the SQL scripts that allow the DBA to recreate the database instance.

When viewed using SQL Management Studio table names start with "dbo."

We can also create scripts for a database with or without explicit file groups. Our current capabilities allow us to specify the following file groups:

1. A file group for all tables.

2. A file group for unique constraints and indexes, although a clustered unique constraint or duplicate index will always be created in the same file group as the corresponding table.

3. A file group for the values of columns whose data type is "Text".

Any further file group granularity must be specified manually.

The scripts for each version of a data model will be zipped together and stored in COLAB under a documents folder named "SQL Create Scripts", for the corresponding data model. The zip file name will be of the form:

<data\_model\_name>\_<data\_model\_version>\_<file\_group\_convention>\_SQLCreateScripts.zip

where <data\_model\_version> will be two or more groups of digits separated by periods and <file\_group\_convention> will either be "NFG" (No File Groups) or "EFG" (Explicit File Groups). Version numbers will be left padded with zeros as necessary so that the files will be correctly sorted by version and file group convention.

For example:

sdm\_2.08\_NFG\_SQLCreateScripts.zip

corresponds to version 2.8 of the data model for the Soil Data Mart database, where no explicit file groups are specified.

Permanently saving all versions of a data model in COLAB will allow a developer to create a "private" database for any version of a data model, without having to ask for scripts to be generated.

Not all column constraints are included in the CreateTables report. This level of detail is included in the Metadata Tables. These constraints are added to the database when a user starts NASIS. The reports beginning with NREPO – Table – Metadata are used to create the metadata tables. This is something that the DBA will do once we give our data model version to them. The Metadata tables are not visible in the NASIS Repo. They are only visible when using SQL Management Studio.

## What categories of database related information cannot currently be documented in our data model repository?

These categories include, but are not necessarily limited to the following:

* Database object permissions
* SQL Server logins
* Stored Procedures and Functions
* Triggers
* Columns whose values are derived by an expression
* Multicolumn check constraints
* Expression check constraints
* SQL Server File Groups and Files
* Views