

# Appendix B Query Tools, Rules, Samples

---

## Query-Writing Tools

Available to you are several tools for writing effective queries. Be sure you have these on hand when you begin.

- **Database Structure Diagrams**  
Located in the “References” section of NASIS Online Help and packaged with *NASIS Getting Started*. Used to identify relationships among tables and which tables form a contiguous table path through the database structure.
- **Data Dictionary Definitions**  
Located in online help and used to identify data types. Can be accessed by using the Help button on the NASIS toolbar or the online help Index tab. See “Identifying Data Types,” on page B.5.
- **Lesson: Writing Custom Queries**  
Located in the *Getting Started* manual as “Chapter 17 Writing Custom Queries.”
- **Help Topics**  
Located in online help (1) “Using NASIS: Creating new queries” and “...Editing queries” and (2) “References: Query rules” and “...Query samples.”
- **Query-Writing Methodology**  
Located below. A general approach to writing queries.
- **Data Elements and Data Types Listing**  
Located in online help “References.” A comprehensive list of data elements and data types organized by table. Also available as a .PDF file for printing with *Adobe Acrobat Reader*.
- **Data Types and Comparison Operators Diagram**  
Located in this appendix and in the “References” of online help.

## Query-Writing Methodology

Following is a set of questions with sample answers and actions that guide you through the query-writing process. Remember to also rely on the other tools described above.

1. **Open the Query table.**  
Select NASIS View menu, Queries, then Query
2. **Ask yourself: What am I looking for?**  
soils with bedrock
3. **Ask yourself: What data element is that**  
Restriction\_kind

4. **Ask yourself: What table is the data element in?**  
component restriction
5. **Ask yourself: What data type is the data element?**  
an unordered code
6. **Ask yourself: What comparison operator can I use with this data type?**  
equals (=), is not equal to (!=), IS NULL, IS NOT NULL, IN ( ), NOT
7. **Ask yourself: What comparison operator will meet the conditions?**  
equals (=)
8. **Ask yourself: Do I want to use this query multiple times changing the values for just a few fields? For example, do I want to create a query that gives me the same information separately for three different restriction\_kinds?**  
yes, consider parameter substitution. Refer to details on page B.4.
9. **Ask yourself: Does the data type allow for parameter substitution?**  
yes, unordered codes allow for parameter substitution. Refer to table on page B.6.
10. **Type the WHERE clause into the Select Condition box.**  
WHERE component\_restriction.restriction\_kind = ? (The ? allows for runtime parameter substitution. The query will stop and prompt for the restriction\_kind.)
11. **Ask yourself: What tables are in my WHERE clause? Type those tables in the FROM clause in a line above the WHERE clause.**  
FROM component\_restriction  
WHERE component\_restriction.restriction\_kind = ?
12. **Ask yourself: Exactly what records do I want in my selected set? (In other words, what table do I want to target?)**  
Do I want to see only the component restrictions that meet the specified condition?

*If Yes—*

Component\_restriction is the default target table, so the query is complete. Proceed to step 13.

*If No—*

Continue.

Do I want to see all restrictions that meet the specified condition for each component?

*If Yes—*

Add the component table to the FROM clause as the default target table.

Type a JOIN statement that includes all tables necessary to form a contiguous path through the database structure.

The query is complete. Proceed to step 13.

```
FROM component, component_restriction
WHERE component_restriction.restriction_kind = ? AND
JOIN component TO component_restriction
```

*If No—*

Continue.

Do I want to see all components and all restrictions that meet this condition for each data mapunit?

*If Yes—*

Add data mapunit table to the FROM clause as the default target table.

Type a JOIN statement that includes all tables necessary to form a contiguous path through the database structure.

Query is complete. Proceed to step 13.

```
FROM data_mapunit, component, component_restriction
WHERE component_restriction.restriction_kind = ? AND
JOIN component TO component_restriction AND
JOIN data_mapunit TO component
```

*If No—*

Reconsider which records you want in your selected set.

**13. Verify that the query conforms to query language rules.**

Click the Verify button.

**14. After successful verification, accept the query as written and close the query editor.**

Click the Apply button.

**15. In the Query table, return to the Query Description field, click the Zoom button, then type a description of the query's purpose and how to run it (which table to target). (Following is an example description.)**

Use this query to select all components and all restrictions for each data mapunit that meets this condition. Run with target table set to data mapunit.

**Note:** The query is complete.

## Query Rules

### General query rules

- Queries work against the permanent database and pull data into the selected set. A query does not re-query the selected set.
- A query has ownership the same way data mapunits, legends, and area types and other objects have ownership. Like other owned objects, queries must be loaded into the selected set to be edited.
- You can create, save, and share queries.
- The query name must be unique among queries owned by your database.
- A query description should contain the intention of the query (e.g. "Use to select soils with restrictive features within certain depths") and how to run it (e.g. "Run this query once with target table set as data mapunit and once with target table set as legend.")
- A query has two basic parts: FROM clause and WHERE clause. These make up the Select Conditions.

### Select conditions rules

- The FROM clause lists all the tables used in the WHERE clause.
- In the FROM clause, multiple tables are separated with a comma.
- Tables in the FROM clause need not be in a certain order, except that the default target table must be first in the list.
- The FROM clause includes all possible target tables.
- The WHERE clause specifies the conditions to be placed on data elements.
- Conditions compare the values of data elements against other data elements, against constant values, or against parameters.
- When multiple tables exist in the FROM clause, define the relationship between tables by adding JOIN statements to the WHERE clause.
- Each JOIN statement is separated with an AND operator.
- JOIN statements must form a contiguous table path through the data structure.
- Table and data element logical names or physical names can be used when defining query conditions.
- **Note:** If your query results seem to ignore the parameters you specified and you do not get the results you expect, verify that your target table is set to the right table.

### Parameter substitution and comparison operator rules

- The use of operators is limited by data types. See Table B-1 Data Types and Comparison Operators on page B.6.
- Character strings in a condition must be in quotes.
- Wildcards: \* and ? are used with MATCHES and IMATCHES; % and \_ are used with LIKE.
- The question mark character (?) in the select condition query defines a parameter for the query. The value for the parameter is entered when the query is run.
- If the question mark character (?) is immediately followed by a string in quotes, the string is used as a label for the parameter (in the NASIS Query Parameters box) when the query is run.
- To make the query case insensitive, use the IMATCHES operator.
- To use a parameter with the IN operator, place the question mark character (?) within the parentheses (for example, <element> IN (?)).
- To use a parameter with the BETWEEN operator, use two question marks (for example, <element> BETWEEN ? AND ?).

## Identifying Data Types

Data types are defined in the Official Data Dictionary. Identifying a data type of a data element is a critical step in writing queries. You can identify them in one of three ways: The help topic “Data elements and data types” found in the online help “References,” the Help button on the NASIS toolbar, or the online help Index tab.

### Data Types help topic

1. Activate the NASIS online help system.
2. In the Help Topics dialog, click on **References** and click **Open**.
3. Click **Data elements and data types** and click **Display**.

**Note:** The Data elements and data types PDF file (with original formatting) can be printed. Follow the instructions on the topic.

### Help button

1. On the NASIS screen with the particular table open and the data element displayed, click the **Help** button, then click the data element name.
2. At the bottom of the Data Element Explanation, click **Data Dictionary**. The official definition appears.

**Note:** If the data element is a link field, you will see a “Display Data Element Name.” Using the online help Index function, search for the Display Name.

3. On the Official Data Dictionary screen, refer to *Data Type*.

### Index function

If the Query Editor is open and you have already begun writing a query, use the Index function to get data types.

1. With the Query Editor open, use the table and column choice lists to insert the data element name into the Select Conditions box. (The logical name appears.)
2. Within the online help system, use the Index function to search for the logical name of the data element (table.element string).
3. At the bottom of the Data Element Explanation, click **Data Dictionary**.
4. On the Official Data Dictionary screen, refer to *Data Type*.

**Data Types** **Comparison Operators**

	=	!=	>	<	>=	<=	IS NULL	IS NOT NULL	LIKE " "	MATCHES " " / MISMATCHES " "	BETWEEN AND	IN( )
Character												
Variable character (string)												
Text (narrative text)	II I	III	III	III	III	III			IV	IV	III	III
Decimal									IV	IV		
Float									IV	IV		
Smallfloat									IV	IV		
Integer									IV	IV		
Smallint									IV	IV		
Money									IV	IV		
Serial									IV	IV		
Date									IV	IV		
Datetime									IV	IV		
Boolean									IV	IV		
Ordered code (choice)									IV	IV		
Unordered code (choice)			II	II	II	II			IV	IV	II	
Evaluation	II I	III	III	III	III	III			IV	IV	III	III
Property	II I	III	III	III	III	III			IV	IV	III	III
Query	II I	III	III	III	III	III			IV	IV	III	III
Rule	II I	III	III	III	III	III			IV	IV	III	III

**Notes:** date and datetime values must be entered in the correct format or an SQL error will result (see Query #4 on page B.10).

NOT, AND, and OR operators are used to combine two conditions; they are not related to data type.

- blank Allowed
- II Allowed by query program, but results may not be meaningful.
- III Allowed by query program, but will result in SQL error when query is executed.
- IV Not allowed

**Table B-1. Data Types and Comparison Operators**

## Sample Query #1

### Purpose:

To select components (or data mapunits) that have bedrock within 75 centimeters of the surface.

### Background:

You start building this query by recognizing that restrictive features, such as bedrock, are in the Component Restrictions table and that depth to the top of the restrictive feature is recorded as a low, high, and representative (\_r) value in centimeters.

### Select conditions:

```
FROM component_restrictions, component, data_mapunit
WHERE component_restrictions.restriction_kind = ? AND
component_restrictions.restriction_depth_to_top_r < ? AND
JOIN component TO component_restrictions AND
JOIN component TO data_mapunit
```

### Explanation of select conditions:

The WHERE conditions include the component\_restrictions table, thus this table must be in the FROM clause. However, you also want to be able to choose either the Component or Data Mapunit table as a target table, so these tables are also included in the FROM clause. Because there is more than one table in the FROM clause, the tables must be joined with a JOIN statement.

The first question mark character (?) will be replaced by a choice list of restriction kinds in the NASIS Query Parameter dialog box when this query is run from the Select Manager. You can choose “bedrock(lithic)” or any other restriction kind when the query is run. The second question mark will be replaced by a blank field where you can enter the depth to bedrock.

**Note:** This sample query is called “Restrictive Features” in the Pangaea database.

## Sample Query #2

### Purpose:

To select all mapunits used in an MLRA.

### Background:

You start building this query by recognizing that the occurrence of a mapunit in a MLRA is recorded in the Legend Area Overlap and Mapunit Area Overlap tables. Although Area Type Name and Area Symbol appear as columns in the Legend Area Overlap table, these columns actually refer to the Area Type and Area tables.

### Select conditions:

```
FROM mapunit, mapunit_area_overlap, legend_area_overlap, area, area_type
WHERE area_type.area_type_name = "MLRA" AND
area.area_symbol MATCHES ? "MLRA Symbol" AND
JOIN area_type TO legend_area_overlap AND
JOIN area TO legend_area_overlap AND
JOIN legend_area_overlap TO mapunit_area_overlap AND
JOIN mapunit_area_overlap TO mapunit
```

### Explanation of select conditions:

The WHERE conditions include the area\_type and area tables. Because these tables are in the WHERE clause, they must be included in the FROM clause. However, you want to select mapunits that exist in an MLRA so the focus of our query is the mapunit table and this table is listed first in the FROM clause (to designate it as the default target table). Other tables in the FROM clause are required to complete a contiguous join between Area Type and Mapunit. Notice that the join path is through the overlap tables because the occurrence of a mapunit in a MLRA is recorded in these tables.

The question mark character (?) will be replaced by a blank field in the NASIS Query Parameters dialog box when this query is run from the Select Manager. In the blank field, you can enter the area symbol for the MLRA when the query is run. The statement "MLRA Symbol" is used for clarity and will be substituted as a label in the NASIS Query Parameter dialog box when the query is run from the Select Manager.

**Note:** This sample query is called "Mapunits in MLRA" in the Pangaea database.

## Sample Query #3

### Purpose:

To select active mapunits that have one or more hydric components and the data for those components.

### Background:

You start building this query by recognizing that the hydric interpretation is in the Component Interpretation table, thus this table is essential to the query. You also recognize that an area can have more than one legend and that all mapunits ever used in a survey legend are retained in the mapunit table, thus you must provide query conditions that load only the legend and mapunits that you need.

### Select conditions:

```
FROM component_interp, area, correlation, data_mapunit, legend, mapunit,
component
WHERE component_interp.interpretation_kind = "hydric soil rating" AND
component_interp.interpretation_rating = "yes" AND
area.area_symbol MATCHES ? AND
legend.soil-survey_area_status = ? AND
mapunit.mapunit_status != "additional" AND
correlation.representative_dmu = 1 AND
JOIN area TO legend AND
JOIN legend TO mapunit AND
JOIN mapunit TO correlation AND
JOIN correlation TO data_mapunit AND
JOIN data_mapunit TO component AND
JOIN component TO component_interp
```

### Explanation of select conditions:

The query will need to be run twice—once with target table set to mapunit and again with the target table set to component\_interp—because data for the report are in both the legend object and data mapunit object. The WHERE conditions include the Component Interpretation, Area, Legend, Mapunit, and Correlation tables, thus these tables must be in the FROM clause. Other tables in the FROM clause are required to complete a contiguous join between the tables. The comparison operator MATCHES is used where the data element is character or variable character. The comparison operators = (equals) or != (not equals) are used where the data element is a code.

**Note:** This sample query is called “Data for hydric soil report” in the Pangaea database.

## Sample Query #4

### Purpose:

To select data mapunits not updated since date (?).

### Background:

Run this query to find any data mapunits that were last updated before the date you specify. The required format is as follows: 1996-01-14 06:12:01, which means January 14, 1996 at 6:12 a.m. and 1 second.

### Select conditions:

```
FROM data_mapunit  
WHERE data_mapunit_when_last_updated < ?
```

### Explanation of select conditions:

This query illustrates the use of the data type "Datetime." To enter a date and time for comparison to a date time column, the full date and time must be entered as in the example.