

Basic Syntactical Items

This chapter describes basic syntactical items, which are not explained any further within the individual SQL statement descriptions.

This chapter covers the following topics:

- Constants
 - Names
 - Parameters
 - include-columns
 - Natural Formats and SQL Data Types
-

Constants

The constants used in the syntactical descriptions of the Natural SQL statements are:

- *constant*
- *integer*

These items are described below.

<i>constant</i>	The item <i>constant</i> always refers to a Natural constant.
<i>integer</i>	The item <i>integer</i> always represents an integer constant.

Note:

If the character for decimal point notation (session parameter DC) is set to a comma (,), any specified numeric constant must not be followed directly by a comma, but must be separated from it by a blank character; otherwise an error or wrong results occur.

Invalid Syntax:	Valid Syntax:
VALUES (1,'A') leads to a syntax error	VALUES (1 , 'A')
VALUES (1,2,3) leads to wrong results	VALUES (1 ,2 ,3)

Names

The names used in the syntactical descriptions of the Natural SQL statements are:

- *authorization-identifier*
- *dsm-name*

- *view-name*
- *column-name*
- *location-name*
- *table-name*
- *correlation-name*

These items are described below.

<i>authorization-identifier</i>	The item <i>authorization-identifier</i> , which is also called creator name, is used to qualify database tables and views. See also below.
<i>dgm-name</i>	The item <i>dgm-name</i> always refers to the name of a Natural data definition module (DDM) as created with the Natural utility SYSDDM.
<i>view-name</i>	The item <i>view-name</i> always refers to the name of a Natural view as defined in the DEFINE DATA statement.
<i>column-name</i>	The item <i>column-name</i> always refers to the name of a physical database column.
<i>location-name</i>	The item <i>location-name</i> always denotes the location of the table. Specification of location-name is optional and belongs to the SQL Extended Set.

table-name	<p>Syntax:</p> <pre>[[location-name.]authorization-identifier.]ddm-name</pre> <p>The item <i>table-name</i> in this section is used to reference both SQL base tables and SQL viewed tables.</p> <p>ddm-name</p> <p>A Natural data definition module (DDM) must have been created for a table to be used. The name of such a DDM must be the same as the corresponding database table name or view name.</p> <p>location-name</p> <p>This optional item specifies the location of the table to be accessed.</p> <p>authorization-identifier</p> <p>There are two ways of specifying the <i>authorization-identifier</i> of a database table or view.</p> <p>One way corresponds to the standard SQL syntax, in which the <i>authorization-identifier</i> is separated from the table name by a period. Using this form, the name of the DDM must be the same as the name of the database table without the <i>authorization-identifier</i>.</p> <p>Example:</p> <pre>DEFINE DATA LOCAL 01 PERS VIEW OF PERSONNEL 02 NAME 02 AGE END-DEFINE SELECT * INTO VIEW PERS FROM SQL.PERSONNEL ...</pre> <p>Alternatively, you can define the <i>authorization-identifier</i> as part of the DDM name. The DDM name then consists of the <i>authorization-identifier</i> and the database table name separated by a hyphen (-). The hyphen between the <i>authorization-identifier</i> and the table name is converted internally into a period.</p> <p>Note:</p> <p>This form of DDM name can also be used with a FIND or READ statement, because it conforms to the DDM naming conventions applicable to these statements.</p> <p>Example:</p> <pre>DEFINE DATA LOCAL 01 PERS VIEW OF SQL-PERSONNEL 02 NAME 02 AGE END-DEFINE SELECT * INTO VIEW PERS FROM SQL-PERSONNEL ...</pre> <p>If the <i>authorization-identifier</i> has been specified neither explicitly nor within the DDM name, it is determined by the SQL database system.</p> <p>In addition to being used in SELECT statements, table names can also be specified in DELETE, INSERT and UPDATE statements.</p> <p>Examples:</p> <pre>... DELETE FROM SQL.PERSONNEL WHERE AGE IS NULL INSERT INTO SQL.PERSONNEL (NAME,AGE) VALUES ('ADKINSON',35) UPDATE SQL.PERSONNEL SET SALARY = SALARY * 1.1 WHERE AGE > 30 ...</pre>
-------------------	---

correlation-name	<p>The item <i>correlation-name</i> represents an alias name for a <i>table-name</i>. It can be used to qualify column names; it also serves to implicitly qualify fields in a Natural view when used with the INTO clause of the SELECT statement.</p> <p>Example:</p> <pre> DEFINE DATA LOCAL 01 PERS-NAME (A20) 01 EMPL-NAME (A20) 01 AGE (I2) END-DEFINE ... SELECT X.NAME , Y.NAME , X.AGE INTO PERS-NAME , EMPL-NAME , AGE FROM SQL-PERSONNEL X , SQL-EMPLOYEES Y WHERE X.AGE = Y.AGE END-SELECT ... </pre> <p>Although in most cases the use of <i>correlation-names</i> is not necessary, they may help to make the statement clearer.</p>
-------------------------	---

Parameters

parameter

<code>[:<i>sql-type</i>:] <i>host-variable</i> [INDICATOR [:] <i>host-variable</i>] [LINDICATOR [:] <i>host-variable</i>]</code>
--

The syntax items are described below:

<i>sql-type</i>	<p>An <i>sql-type</i> specifies the SQL data type of the <i>host-variable</i> when it is used for DB2 access. The specification of <i>sql-type</i> is optional as most SQL data types are implicitly assigned to Natural host-variables. However for some Natural host-variables the SQL data type can not be associated implicitly.</p> <p><i>sql-type</i> belongs to the SQL Extended Set.</p> <p>If an <i>sql-type</i> is specified, it has to be surrounded by colons (:).</p> <p>Valid sql-types are:</p> <table border="1"> <thead> <tr> <th>sql-type</th> <th>Natural format</th> <th>DB2 SQL data type</th> </tr> </thead> <tbody> <tr> <td>BLOBFILE</td> <td>group(I4,I4,I4,A255)</td> <td>BLOB file reference(916/917)</td> </tr> <tr> <td>CLOBFILE</td> <td>group(I4,I4,I4,A255)</td> <td>CLOB file reference(920/921)</td> </tr> <tr> <td>DBCLOBFILE</td> <td>group(I4,I4,I4,A255)</td> <td>DBCLOB file reference(924/925)</td> </tr> <tr> <td>BLOBLOC</td> <td>(I4)</td> <td>BLOB locator960/961)</td> </tr> <tr> <td>CLOBLOC</td> <td>(I4)</td> <td>CLOB locator(964/965)</td> </tr> <tr> <td>DBCLOBLOC</td> <td>(I4)</td> <td>DBCLOB locator(968/969)</td> </tr> </tbody> </table> <p>See also <i>Natural Formats and SQL Data Types</i>.</p>	sql-type	Natural format	DB2 SQL data type	BLOBFILE	group(I4,I4,I4,A255)	BLOB file reference(916/917)	CLOBFILE	group(I4,I4,I4,A255)	CLOB file reference(920/921)	DBCLOBFILE	group(I4,I4,I4,A255)	DBCLOB file reference(924/925)	BLOBLOC	(I4)	BLOB locator960/961)	CLOBLOC	(I4)	CLOB locator(964/965)	DBCLOBLOC	(I4)	DBCLOB locator(968/969)
sql-type	Natural format	DB2 SQL data type																				
BLOBFILE	group(I4,I4,I4,A255)	BLOB file reference(916/917)																				
CLOBFILE	group(I4,I4,I4,A255)	CLOB file reference(920/921)																				
DBCLOBFILE	group(I4,I4,I4,A255)	DBCLOB file reference(924/925)																				
BLOBLOC	(I4)	BLOB locator960/961)																				
CLOBLOC	(I4)	CLOB locator(964/965)																				
DBCLOBLOC	(I4)	DBCLOB locator(968/969)																				
<i>host-variable</i>	<p>A <i>host-variable</i> is a Natural user-defined variable (no system variable) which is referenced in an SQL statement. It can be defined either as an individual field or as part of a Natural view. Under certain circumstances (multifetch into Natural arrays, MERGE statement), it is also possible to specify fixed one-dimensional array ranges such as <i>array(*)</i> or <i>array(1:5)</i>. See also compiler option DB2ARRY.</p> <p>When defined as a receiving field (for example, in the INTO clause), a <i>host-variable</i> identifies a variable to which a value is assigned by the database system.</p> <p>When defined as a sending field (for example, in the WHERE clause), a <i>host-variable</i> specifies a value to be passed from the program to the database system.</p> <p>See also <i>Natural Formats and SQL Data Types</i>.</p>																					

[:]	<p>Colon:</p> <p>To comply with SQL standards, a host-variable can also be prefixed by a colon (:). When used with flexible SQL, host-variables must be qualified by colons.</p> <p>Example:</p> <pre>SELECT NAME INTO :#NAME FROM PERSONNEL WHERE AGE = :VALUE</pre> <p>The colon is always required if the variable name is identical to an SQL reserved word. In a context in which either a host-variable or a column can be referenced, the use of a name without a colon is interpreted as a reference to a column.</p>
INDICATOR	<p>INDICATOR Clause:</p> <p>The INDICATOR clause is an optional feature to distinguish between a "null" value (that is, no value at all) and the actual values 0 or "blank".</p> <p>When specified with a receiving <i>host-variable</i> (target field), the INDICATOR <i>host-variable</i> (null indicator field) serves to find out whether a column to be retrieved is "null".</p> <p>Example:</p> <pre>DEFINE DATA LOCAL 1 NAME (A20) 1 NAMEIND (I2) END-DEFINE SELECT * INTO NAME INDICATOR NAMEIND ... </pre> <p>In this example, NAME represents the receiving <i>host-variable</i> and NAMEIND the null indicator field.</p> <p>If a null indicator field has been specified and the column to be retrieved is null, the value of the null indicator field is negative and the target field is set to 0 or "blank" depending on its data type. Otherwise, the value of the null indicator field is greater than or equal to 0.</p> <p>When specified with a sending <i>host-variable</i> (source field), the null indicator field is used to designate a null value for this field.</p> <p>Example:</p> <pre>DEFINE DATA LOCAL 1 NAME (A20) 1 NAMEIND (I2) UPDATE ... SET NAME = :NAME INDICATOR :NAMEIND WHERE ... </pre> <p>In this example, :NAME represents the sending <i>host-variable</i> and :NAMEIND the null indicator field. By entering a negative value as input for the null indicator field, a null value is assigned to a database column.</p> <p>An INDICATOR <i>host-variable</i> is of format/length I2.</p>

LINDICATOR	<p>LINDICATOR Clause:</p> <p>The LINDICATOR clause is an optional feature which is used to support columns of varying lengths, for example, VARCHAR or LONG VARCHAR type.</p> <p>When specified with a receiving <i>host-variable</i> (target field), the LINDICATOR <i>host-variable</i> (length indicator field) contains the number of characters actually returned by the database into the target field. The target field is always padded with blanks.</p> <p>If the VARCHAR or LONG VARCHAR column contains more characters than fit in the target field, the length indicator field is set to the length actually returned (that is, the length of the target field) and the null indicator field (if specified) is set to the total length of this column.</p> <p>Example</p> <pre>DEFINE DATA LOCAL 1 ADDRESSLIND (I2) 1 ADDRESS (A50/1:6) END-DEFINE SELECT * INTO :ADDRESS(*) LINDICATOR :ADDRESSLIND ...</pre> <p>In this example, :ADDRESS(*) represents the target field which receives the first 300 bytes (if available) of the addressed VARCHAR or LONG VARCHAR column, and :ADDRESSLIND represents the length indicator field which contains the number of characters actually returned.</p> <p>When specified with a sending <i>host-variable</i> (source field), the length indicator field specifies the number of characters of the source field which are to be passed to the database.</p> <p>Example:</p> <pre>DEFINE DATA LOCAL 1 NAMELIND (I2) 1 NAME (A20) 1 AGE (I2) END-DEFINE MOVE 4 TO NAMELIND MOVE 'ABC%' TO NAME SELECT AGE INTO :AGE WHERE NAME LIKE :NAME LINDICATOR :NAMELIND ...</pre> <p>A LINDICATOR <i>host-variable</i> is of format/length I2 or I4. For performance reasons, it should be specified immediately before the corresponding target or source field; otherwise, the field is copied to the temporary storage at runtime.</p> <p>If the LINDICATOR field is defined as an I2 field, the SQL data type VARCHAR is used for sending or receiving the corresponding column. If the LINDICATOR <i>host-variable</i> is specified as I4, a large object data type (CLOB/BLOB) is used.</p> <p>If the field is defined as DYNAMIC, the column is read in an internal loop up to its real length. The LINDICATOR field and the Natural system variable *LENGTH are set to this length. In case of a fixed length field, the column is read up to the defined length. In both cases, the field is written up to the value defined in the LINDICATOR field.</p> <p>Let a fixed length field be defined with a LINDICATOR field specified as I2. If the VARCHAR column contains more characters than fit into this fixed length field, the length indicator field is set to the length actually returned and the null indicator field (if specified) is set to the total length of this column (retrieval). This is not possible for fixed length fields greater than or equal to 32 KB (length does not fit into null indicator field).</p>
-------------------	--

include-columns

This clause belongs to the SQL Extended Set. It is available in the statements DELETE, INSERT, MERGE and UPDATE.

INCLUDE (<i>column-name data-type,...</i>)

Syntax Element	Description
INCLUDE	INCLUDE introduces a list of columns that is to be included in the result table of a DELETE, INSERT, MERGE or UPDATE statement. INCLUDE can only be specified when a DELETE, INSERT, MERGE or UPDATE statement is nested in the FROM clause of a SELECT statement.
<i>column-name</i>	Specifies the name of a column of the result table of the MERGE statement that is not the same name as another include column or a column in the target table.
<i>data-type</i>	Specifies the data type of the include column. See below.

data-type

{ <i>built-in-type</i> }
{ <i>distinct-type</i> }

Syntax Element	Description
<i>built-in-type</i>	Specifies a built-in data type. See the IBM DB2 for z/OS documentation for a description of built-in types.
<i>distinct-type</i>	Specifies a distinct type.

Natural Formats and SQL Data Types

The Natural format of a host-variable is converted to an SQL data type according to the following table:

Natural Format/Length	SQL Data Type
An , A DYNAMIC	CHAR (n) , VARCHAR(n), CLOB(n)
B2	SMALLINT
B4	INT
F4	REAL
F8	DOUBLE PRECISION
I2	SMALLINT
I4	INT
$Nnn.m$	NUMERIC ($nn+m,m$)
$Pnn.m$	NUMERIC ($nn+m,m$)
T, A8	TIME
D, A10	DATE
A26	TIMESTAMP
Gn ; for view fields only	GRAPHIC (n)
Un , U DYNAMIC	GRAPHIC (n) ,VARGRAPHIC(n), DBCLOB(n) CCSID 1200
Bn , B DYNAMIC	BINARY(n), VARBINARY(n), BLOB(n)
P19.0	BIGINT
F8	DECFLOAT(n)
A DYNAMIC, B DYNAMIC, U DYNAMIC	XML
Group structure(I4,I4,I4,A255) prefixed with :BLOBFILE:	BLOB-file-reference
Group structure(I4,I4,I4,A255) prefixed with :CLOBFILE:	CLOB-file-reference
Group structure(I4,I4,I4,A255) prefixed with :DBCLOBFILE:	DBCLOB-file-reference
I4 prefixed with :BLOBLOC:	BLOB-locator
I4 prefixed with :CLOBLOC:	CLOB-locator
I4 prefixed with :DBCLOBLOC:	DBCLOB-locator

Natural does not check whether the converted SQL data type is compatible to the database column. Except for fields of format N, no data conversion is done.

In addition, the following extensions to standard Natural formats are available with Natural SQL:

- A one-dimensional array of format A can be used to support alphanumeric columns longer than 253 bytes. This array must be defined beginning with index 1 and can only be referenced by using an asterisk (*) as the index. The corresponding SQL data type is CHAR(n), where n is the total number of bytes in the array.

- A special host variable indicated by the keyword `LINDICATOR` can be used to support variable-length columns. The corresponding SQL data type is `VARCHAR(n)`; see also the `LINDICATOR` clause.
- The Natural formats date (D) and time (T) can be used with Natural for DB2. They are converted to DB2 `DATE` and `TIME`.

A sending field specified as one-dimensional array without a `LINDICATOR` field is converted into the SQL data type `VARCHAR`. The length is the total number of bytes in the array, not taking into account trailing blanks.