UPDATE - SQL UPDATE - SQL

# **UPDATE - SQL**

This chapter covers the following topics:

- Function
- Syntax Description
- Examples

For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Belongs to Function Group: Database Access and Update

See also the following sections in the *Database Management System Interfaces* documentation:

- NDB UPDATE SQL in the Natural for DB2 part.
- *UPDATE* in the *Natural for SQL/DS* part.

# **Function**

The SQL UPDATE statement is used to perform an UPDATE operation on either rows in a table without using a cursor ("searched" UPDATE) or columns in a row to which a cursor is positioned ("positioned" UPDATE).

# **Syntax Description**

Two different structures are possible:

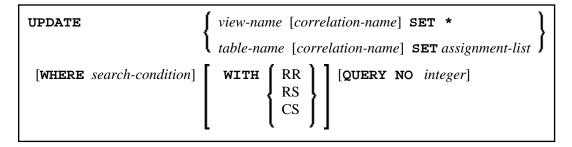
- Syntax 1 Searched UPDATE
- Syntax 2 Positioned UPDATE

### Syntax 1 - Searched UPDATE

The "searched" UPDATE statement is a stand-alone statement not related to any SELECT statement. With a single statement you can update zero, one, multiple or all rows of a table. The rows to be updated are determined by a *search-condition* that is applied to the table. Optionally, view and table names can be assigned a *correlation-name*.

#### Note:

The number of rows that have actually been updated with a "searched" UPDATE can be ascertained by using the system variable \*ROWCOUNT (see the *System Variables* documentation).



For an explanation of the symbols used in the syntax diagram, see *Syntax Symbols*.

Syntax Element Description - Syntax 1:

view-name	View Name:	
	Refers to the name of a Natural view as defined in the DEFINE DATA statement. For further information, see <i>view-name</i> (in the section <i>Basic Syntactical Items</i> ).	
SET	SET Clause:	
	If a view has been specified for updating, an asterisk (*) has to be specified in the SET clause, because all columns of the view must be updated.	
	If a table has been specified for updating, the SET clause must contain either an <i>assignment-list</i> or the name of the view which contains the columns to be updated.	
assignment-list	See Assignment List below.	
WHERE search-condition	WHERE Clause:  This clause is used to specify the selection criteria for the rows to be updated.	
	If no WHERE clause is specified, the entire table is updated.	

#### **Assignment List**

$$\left\{ \begin{array}{c} column-name = \left\{ \begin{array}{c} scalar-expression \ \mathbf{NULL} \end{array} \right\} \right\}$$

In an *assignment-list*, you can assign values to one or more columns. A value can be either a *scalar-expression* or NULL. For further information, see *Scalar Expressions*.

If the value NULL has been assigned, it means that the addressed field is to contain no value (not even the value 0 or "blank").

#### **SQL Extended Set**

The following syntax elements belong to the SQL Extended Set:

correlation-name	The item <i>correlation-name</i> represents an alias name for a <i>table-name</i> . For further information, see <i>correlation-name</i> (in the section <i>Basic Syntactical Items</i> ).	
WITH	WITH Isolation Level Clause:  This clause allows the explicit specification of the isolation level used when locating the row to be updated. For detailed information, see WITH - Isolation Level in the corresponding section NDB - SELECT - Cursor-Oriented (in the Natural for DB2 part of the Database Management System Interfaces documentation).	
QUERYNO integer	QUERYNO Clause:  This clause allows you to explicitly specify the number to be used in EXPLAIN output and trace records for this statement. The number is used as QUERYNO column in the PLAN_TABLE for the rows that contain information on this statement.	

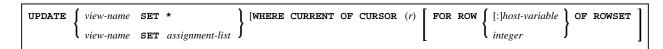
## **Syntax 2 - Positioned UPDATE**

The "positioned" UPDATE statement always refers to a cursor within a database loop. Thus, the table or view referenced by a positioned UPDATE statement must be the same as the one referenced by the corresponding SELECT statement; otherwise an error message is returned. A positioned UPDATE cannot be used with a non-cursor selection.

#### Common Set Syntax



#### Extended Set Syntax



Syntax Element Description - Syntax 2:

UPDATE - SQL Examples

view-name	View Name:
	Refers to the name of a Natural view as defined in the DEFINE DATA statement. For further information, see <i>view-name</i> (in the section <i>Basic Syntactical Items</i> ).
SET *	SET Clause:
SET assignment-list	If a view has been specified for updating, an asterisk (*) has to be specified in the SET clause, because all columns of the view must be updated.
	If a table has been specified for updating, the SET clause must contain either an <i>assignment-list</i> or the name of view which contains the columns to be updated.
WHERE CURRENT OF CURSOR	Statement Reference:
(r)	The (r) notation is used to reference the statement which was used to select the row to be updated. If no statement reference is specified, the UPDATE statement is related to the innermost active processing loop in which a database record was selected.
FOR ROW OF ROWSET	FOR ROW OF ROWSET Clause:
	This clause belongs to the SQL Extended Set.
	The optional FOR ROW OF ROWSET clause for positioned SQL UPDATE statements specifies which row of the current rowset has to be updated. It should only be specified if the UPDATE statement is related to a SELECT statement which uses rowset positioning and which has column arrays in the <i>INTO Clause</i> .
	If this clause is omitted, all rows of the current rowset are updated by the values in the <i>assignment-list</i> .
	This clause cannot be specified if <i>view-name</i> SET * is specified.

# **Examples**

- Example 1 Searched UPDATE
- Example 2 Searched UPDATE with assignment-list
- Example 3 Positioned UPDATE
- Example 4 Positioned UPDATE with assignment-list

## **Example 1 - Searched UPDATE**

```
DEFINE DATA LOCAL

1 PERS VIEW OF SQL-PERSONNEL

2 NAME

2 AGE
...
END-DEFINE
...
ASSIGN AGE = 45
ASSIGN NAME = 'SCHMIDT'
UPDATE PERS SET * WHERE NAME = 'SCHMIDT'
```

## **Example 2 - Searched UPDATE with assignment-list**

```
DEFINE DATA LOCAL

1 PERS VIEW OF SQL-PERSONNEL

2 NAME

2 AGE
...
END-DEFINE
...
UPDATE SQL-PERSONNEL SET AGE = AGE + 1 WHERE NAME = 'SCHMIDT'
```

### **Example 3 - Positioned UPDATE**

```
DEFINE DATA LOCAL

1 PERS VIEW OF SQL-PERSONNEL

2 NAME

2 AGE
...

END-DEFINE
...

SELECT * INTO PERS FROM SQL_PERSONNEL WHERE NAME = 'SCHMIDT'
COMPUTE AGE = AGE + 1
UPDATE PERS SET * WHERE CURRENT OF CURSOR
END-SELECT
```

## **Example 4 - Positioned UPDATE with assignment-list**

```
DEFINE DATA LOCAL

1 PERS VIEW OF SQL-PERSONNEL

2 NAME

2 AGE
...
END-DEFINE
...
SELECT * INTO PERS FROM SQL-PERSONNEL WHERE NAME = 'SCHMIDT'
UPDATE SQL-PERSONNEL SET AGE = AGE + 1 WHERE CURRENT OF CURSOR
END-SELECT
...
```