INSERT - SQL

Common Set Syntax:

```
INSERT INTO table-name { (*) [VALUES-clause]
                       { [(column-list)] VALUE-LIST
```

Extended Set Syntax:

```
INSERT INTO table-name { (*) [OVERRIDING USER VALUE] [VALUES-clause]
                       { [(column-list)] [OVERRIDING USER VALUE] VALUE-LIST
```

This chaptercovers the following topics:

- Function
- Syntax Description
- Example

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

Belongs to Function Group: *Database Access and Update*

Function

The SQL `INSERT` statement is used to add one or more new rows to a table.

Syntax Description

Syntax Element	Description
`INTO` *`table-name`*	**INTO Clause:** In the `INTO` clause, the table is specified into which the new rows are to be inserted. See further information on *`table-name`*.

Syntax Element	Description
column-list	**Column List:** Syntax: *column-name...* In the *column-list*, one or more *column-names* can be specified, which are to be supplied with values in the row currently inserted. If a *column-list* is specified, the sequence of the columns must match with the sequence of the values either specified in the *insert-item-list* or contained in the specified view (see below). If the *column-list* is omitted, the values in the *insert-item-list* or in the specified view are inserted according to an implicit list of all the columns in the order they exist in the table.
VALUES-clause	**Values Clause:** With the VALUES clause, you insert a *single* row into the table. See *VALUES Clause* below.

insert-item-list	**INSERT Single Row:**
	
	In the *insert-item-list*, you can specify one or more values to be assigned to the columns specified in the *column-list*. The sequence of the specified values must match the sequence of the columns.
	
	If no *column-list* is specified, the values in the *insert-item-list* are inserted according to an implicit list of all the columns in the order they exist in the table.
	
	The values to be specified in the *insert-item-list* can be *constants*, *parameters*, *special-registers* or NULL.
	
	See the section *Basic Syntactical Items* for information on *view-name*, *constant* and *parameter*. See also the information on *special-register*.
	
	If the value NULL has been assigned, this means that the addressed field is to receive no value (not even the value 0 or "blank").
	
	Example - INSERT Single Row:
	
	`...`
	`INSERT INTO SQL-PERSONNEL (NAME,AGE)`
	`  VALUES ('ADKINSON',35)`
	`...`

Syntax Element	Description
OVERRIDING USER VALUE	**OVERRIDING USER VALUE Clause:** This clause belongs to the SQL Extended Set. This clause is not currently supported. When used, it will cause a compiler error.

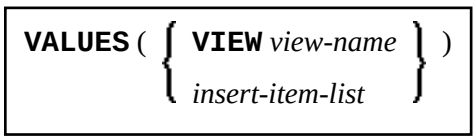
VALUES Clause

With the VALUES clause, you insert a *single* row into the table. Depending on whether an asterisk (*) or a *column-list* has been specified, the VALUES clause can take one of the following forms:

VALUES Clause with Preceding Asterisk Notation

If asterisk notation is specified, a view *must* be specified in the VALUES clause. With the field values of this view, a new row is inserted into the specified table using the field names of the view as column names of the row.

VALUES Clause with Preceding Column List

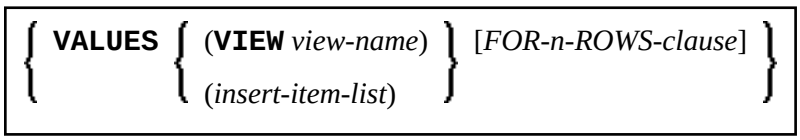


If a *column-list* is specified and a view is referenced in the VALUES Clause, the number of items specified in the column list must correspond to the number of fields defined in the view within the *VALUE-LIST*.

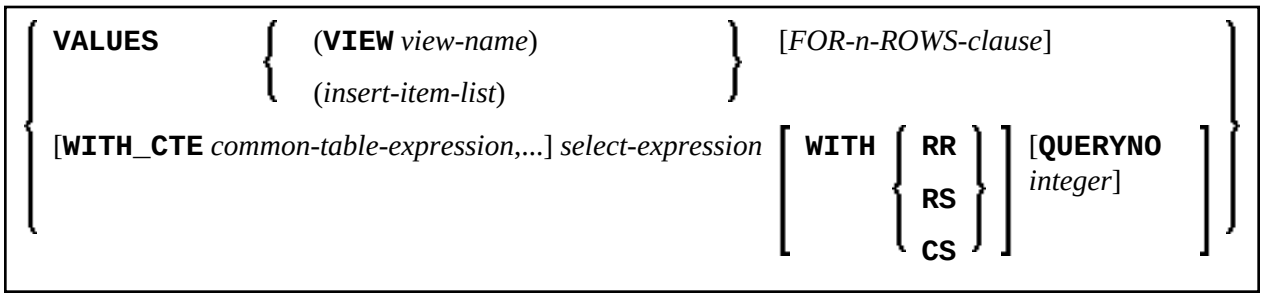
If no *column-list* is specified, the fields defined in the view are inserted according to an implicit list of all the columns in the order they exist in the specified table.

VALUE-LIST

Common Set Syntax:



Extended Set Syntax:



Syntax Description:

Syntax Element	Description
VIEW *view-name*	**View Name:** With the field values of this view, a new row is inserted into the specified table using the field names of the view as column names of the row.

insert-item-list	**INSERT Single Row:**
	
	In the *insert-item-list*, you can specify one or more values to be assigned to the columns specified in the column-list. The sequence of the specified values must match the sequence of the columns.
	
	If no *column-list* is specified, the values in the *insert-item-list* are inserted according to an implicit list of all the columns in the order they exist in the table.
	
	The values to be specified in the *insert-item-list* can be constants, parameters, special-registers or NULL.
	
	See the section *Basic Syntactical Items* for information on *view-name*, *constant* and *parameter*. See also the information on special-register.
	
	If the value NULL has been assigned, this means that the addressed field is to receive no value (not even the value 0 or blank).
	
	Example - INSERT Single Row:
	
	`...`
	`INSERT INTO SQL-PERSONNEL (NAME,AGE)`
	`VALUES ('ADKINSON',35)`
	`...`

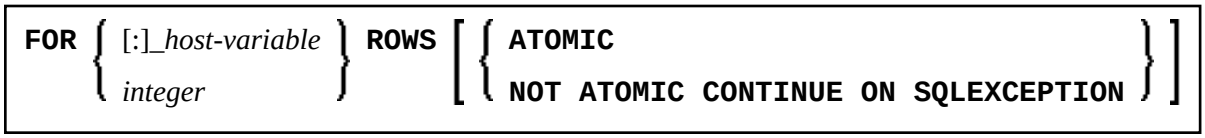
FOR-n-ROWS-clause	**FOR *n* Rows Clause:** Optional clause, see *FOR-n-ROWS-Clause* below.

Syntax Element	Description
`WITH_CTE` *`common-table-expression`*	**WITH_CTE Clause:** This clause belongs to the SQL Extended Set. This optional clause permits defining a result table which can be referenced in any `FROM` clause of the `SELECT` statement that follows. Multiple common-table-expressions can be specified following the single `WITH_CTE` keyword. Each common-table-expression can also be referenced in the `FROM` clause of subsequent common-table-expression. For more information, see *WITH CTE common-table-expression,...* in the section *SELECT - Cursor-Oriented*.

`select-expression`	**INSERT Multiple Rows:**
	
	This clause belongs to the SQL Extended Set.
	
	With a *select-expression*, you insert *multiple* rows into a table. The *select-expression* is evaluated and each row of the result table is treated as if the values in this row were specified as values in a *VALUES Clause* of a single-row `INSERT` operation.
	
	For further information, see *Select Expressions*.
	
	Example - Insert Multiple Rows:
	
	`...`
	`INSERT INTO SQL-RETIREE (NAME,AGE,SEX)`
	`  SELECT LASTNAME, AGE, SEX`
	`  FROM SQL-EMPLOYEES`
	`  WHERE AGE > 60`
	`...`
	
	Note:
	The number of rows that have actually been inserted can be ascertained by using the system variable `*ROWCOUNT`.

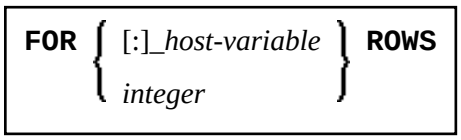
Syntax Element	Description	
`WITH RR/RS/CS`	**WITH Isolation Level Clause:** This clause belongs to the SQL Extended Set. This clause allows the explicit specification of the isolation level used when locating the rows to be inserted. It is only valid against DB2 databases. When used against other databases, it will cause runtime errors.	
	`CS`	Cursor Stability
	`RR`	Repeatable Read
	`RS`	Read Stability
`QUERYNO_`*`integer`*	**QUERYNO Clause:** This clause belongs to the SQL Extended Set. This clause is not currently supported and will be ignored.	

FOR-*n*-ROWS-Clause



This clause is composed of the following subclauses:

FOR [:] *hostvariable/integer* ROWS Clause



The specification of this clause is optional. It should only be specified, if

- compiler option `DB2ARRY` is specifed
- and multiple rows are to be inserted from arrays specified in the *insert-item-list* of the *VALUES Clause*.

If specified, [:]_*hostvariable/integer* determines the number of rows to be inserted into the DB2 table from the arrays specified in the *insert-item-list* of the *VALUES Clause* starting with the first occurrence.

The purpose of this clause is to improve the performance of programs inserting rows from Natural arrays in a loop. By using this clause, the rows contained in the arrays can be inserted by one SQL statement.

See example below.

ATOMIC Clause

```
{ ATOMIC                                  }
{ NOT ATOMIC CONTINUE ON SQLEXCEPTION     }
```

This clause specifies whether the insertion of multiple rows should be treated by DB2 as an atomic operation or not.

It should only be specified, if

- compiler option DB2ARRY is specifed
- and multiple rows are to be inserted from arrays specified in the *insert-item-list* of the *VALUES Clause*.

Syntax Description:

Syntax Element	Description
ATOMIC	Specifies that in case of any error no row is inserted into the target table. This is the default value.
NOT ATOMIC CONTINUE ON SQLEXCEPTION	Specifies that in case of errors all rows for which no error occurred are inserted while those rows for which errors occurred are discarded by DB2.

See the *DB2 SQL REFERENCE* for sqlcodes returned in such cases.

Example

```
DEFINE DATA LOCAL
01 NAME        (A20/1:10)  INIT <'ZILLER1','ZILLER2','ZILLER3','ZILLER4'
                                ,'ZILLER5','ZILLER6','ZILLER7','ZILLER8'
                                ,'ZILLER9','ZILLERA'>
01 ADDRESS     (A100/1:10) INIT <'ANGEL STREET 1','ANGEL STREET 2'
                                ,'ANGEL STREET 3','ANGEL STREET 4'
                                ,'ANGEL STREET 5','ANGEL STREET 6'
                                ,'ANGEL STREET 7','ANGEL STREET 8'
                                ,'ANGEL STREET 9','ANGEL STREET 10'>
01 DATENATD (D/1:10)  INIT <D'1954-03-27',D'1954-03-27',D'1954-03-27'
                           ,D'1954-03-27',D'1954-03-27',D'1954-03-27'
                           ,D'1954-03-27',D'1954-03-27',D'1954-03-27'
                           ,D'1954-03-27'>
01 SALARY      (P4.2/1:10) INIT <1000,2000,3000,4000,5000
                                ,6000,7000,8000,9000,9999>
01 L§ADDRESS   (I2/1:10) INIT <14,14,14,14,14,14,14,14,14,15>
01 N§ADDRESS   (I2/1:10) INIT <00,00,00,00,00,00,00,00,00,00>
01 ROWS        (I4)
01 INDEX       (I4)
01 V1 VIEW OF NAT-DEMO_ID
02 NAME
02 ADDRESS     (EM=X(20))
```

```
02 DATEOFBIRTH
02 SALARY
01 ROWCOUNT  (I4)
END-DEFINE
OPTIONS DB2ARRY=ON                  /* <-- ENABLE DB2 ARRAY
ROWCOUNT := 10
INSERT INTO NAT-DEMO_ID
       (NAME,ADDRESS,DATEOFBIRTH,SALARY)
       VALUES
       (:NAME(*),                   /* <-- ARRAY
        :ADDRESS(*)                 /* <-- ARRAY
        INDICATOR :N§ADDRESS(*)     /* <-- ARRAY
        LINDICATOR :L§ADDRESS(*),   /* <-- ARRAY DB2 VCHAR
        :DATENATD(1:10),            /* <-- ARRAY NATURAL DATES
        :SALARY(01:10)              /* <-- ARRAY NATURAL PACKED
       )
       FOR :ROWCOUNT ROWS
SELECT * INTO VIEW V1 FROM NAT-DEMO_ID WHERE NAME > 'Z'
DISPLAY V1                          /* <-- VERIFY INSERT
END-SELECT
END
```