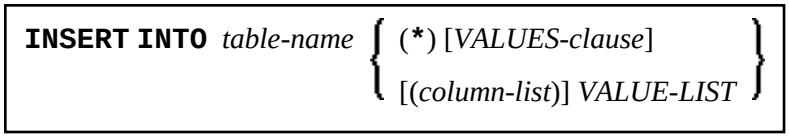
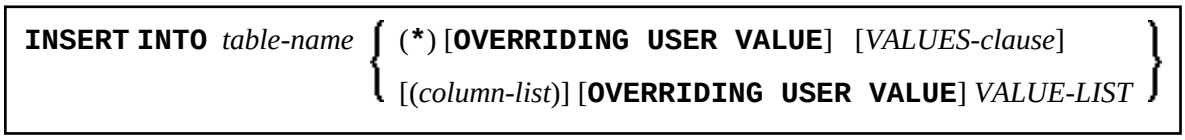
INSERT - SQL

Common Set-Syntax:



Extended Set-Syntax:



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Beispiel

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Siehe auch die folgenden Abschnitte in der *Database Management System Interfaces*-Dokumentation:

- *NDB - INSERT* im *Natural for DB2*.
- *INSERT* im *Natural for SQL/DS*.

Funktion

Das SQL-Statement `INSERT` dient dazu, einer Tabelle eine oder mehrere neue Reihen hinzuzufügen.

Syntax-Beschreibung

INTO *table-name*	**INTO-Klausel:** In der INTO-Klausel geben Sie an, welcher Tabelle Reihen hinzugefügt werden sollen. Siehe auch *table-name*.

column-list	**column-list:** Syntax: `column-name...` In der *column-list* können Sie eine oder mehrere Spalten angeben, die in der hinzugefügten Reihe Werte erhalten sollen. Die Reihenfolge der angegebenen Spalten muss der Reihenfolge der Werte entsprechen, die in der *insert-item-list* oder im angegebenen View sind (siehe unten). Wenn Sie keine *column-list* angeben, werden die in der *insert-item-list* bzw. im View angegebenen Werte entsprechend der impliziten Liste aller Spalten eingefügt, und zwar in der Reihenfolge, in der sie in der Tabelle stehen.
VALUES-clause	**Values-Klausel:** Mit dieser Klausel fügen Sie eine einzelne Reihe in die Tabelle ein. Siehe *VALUES-Klausel* weiter unten.

insert-item-list	**INSERT Single Row:**
	
	In der *insert-item-list* können Sie einen oder mehrere Werte angeben, die den in der *column-list* angegebenen Spalten zugewiesen werden sollen. Die Reihenfolge der angegebenen Werte muss der der Spalten entsprechen.
	
	Wenn keine *column-list* angegben wird, werden die Werte in der *insert-item-list* nach einer impliziten Liste mit allein Spalten in der Reihenfolge eingefügt, wie sie in der Tabelle vorkommen.
	
	Die in der *insert-item-list* anzugebenden Werten können *constants*, *parameters*, *special-registers* oder `NULL` sein.
	
	Informationen zu *view-name* siehe Abschnitt *Basic Syntactical Items*, *constant* und *parameter*. Siehe auch die Informationen zu *special-register*.
	
	Wenn der Wert `NULL` zugewiesen wurde, bedeutet dies, dass das adressierte Feld keinen Wert (auch nicht Wert 0 oder leer) erhalten soll.
	
	Beispiel - `INSERT` Single Row:
	
	`...`
	`INSERT INTO SQL-PERSONNEL (NAME,AGE)`
	`  VALUES ('ADKINSON',35)`
	`...`

OVERRIDING USER VALUE	**OVERRIDING USER VALUE-Klausel:** Diese Klausel gehört zum SQL Extended Set. Diese Klausel bewirkt, dass der Wert ignoriert wird, der in der VALUES-Klausel angegeben ist oder von einem Fullselect für eine Spalte erzeugt wurde, die als GENERATED ALWAYS definiert worden ist.

VALUES-Klausel

Mit der VALUES-Klausel fügen Sie eine einzelne Reihe in die Tabelle ein. Der VALUES-Klausel kann entweder ein Stern (*) oder eine *column-list* vorangestellt werden, und sie hat dementsprechend eine der folgenden Formen:

VALUES-Klausel mit vorangehender Stern-Notation

VALUES (**VIEW** *view-name*)

Wenn Sie Stern-Notation angeben, *müssen* Sie in der VALUES-Klausel einen View angeben. Mit den Feldwerten des Views wird dann eine neue Reihe in die Tabelle eingefügt, wobei die Feldnamen des Views als Spaltennamen der Reihe verwendet werden.

VALUES-Klausel mit vorangehender *column-list*

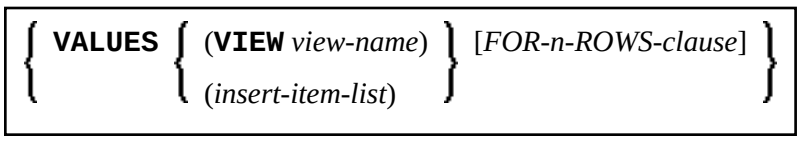
[(*column-list*)] [**OVERRIDING USER VALUE**] *VALUE-LIST*

Wenn Sie eine *column-list* angeben und in der VALUES-Klausel einen View referenzieren, muss die Anzahl der Spalten in der *column-list* der Anzahl der Felder im View innerhalb der VALUE-LIST entsprechen.

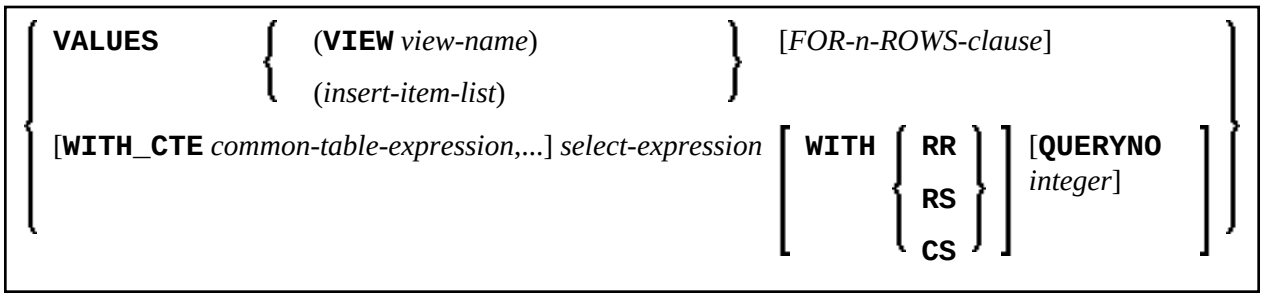
Wenn Sie keine *column-list* angeben, werden die im View angegebenen Werte entsprechend der impliziten Liste aller Spalten in der Reihenfolge, in der sie in der Tabelle stehen, eingefügt.

VALUE-LIST

Common Set-Syntax:



Extended Set-Syntax:



Syntax-Beschreibung:

VIEW *view-name*	**View-Name:** Mit den Feldwerten dieses Views wird eine neue Reihe in die angegebene Tabelle eingefügt, wobei die Feldnamen des Views als Spaltennamen der Reihe benutzt werden.

insert-item-list	**INSERT Single Row:**
	
	In der *insert-item-list* können Sie einen oder mehrere Werte angeben, die den in der *column-list* angegebenen Spalten zugewiesen werden sollen. Die Reihenfolge der angegebenen Werte muss mit der Reihenfolge der Spalten übereinstimmen.
	
	Wenn keine *column-list* angegeben wird, werden die Werte in der *insert-item-list* nach einer impliziten Liste mit allen Spalten in der Reihenfolge eingefügt, wie sie in der Tabelle vorkommen.
	
	Die in der *insert-item-list* anzugebenden Werte können Konstanten, Parameter, *special-registers* oder `NULL` sein.
	
	Informationen zu *view-name*, *constant* und *parameter* siehe *Basic Syntactical Items*. Siehe auch die Informationen zu *special-register*.
	
	Wenn der Wert `NULL` zugewiesen worden ist, bedeutet dies, dass das adressierte Feld keinen Wert erhalten soll (auch nicht den Wert 0 oder Leerzeichen).
	
	Beispiel - INSERT Single Row:
	
	`...`
	`INSERT INTO SQL-PERSONNEL (NAME,AGE)`
	`VALUES ('ADKINSON',35)`
	`...`

FOR-n-ROWS-clause	Optionale Klausel, siehe weiter unten.

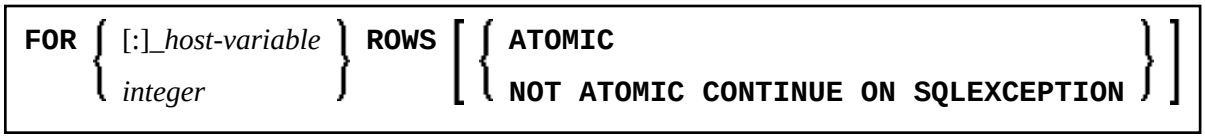
WITH_CTE ***common-table-expression***	Diese Klausel gehört zum SQL Extended Set. Diese optionale Klausel ermöglicht die Definition einer Ergebnistabelle, die in einer `FROM`-Klausel des folgenden `SELECT`-Statements referenziert werden kann. Mehrere *common-table-expressions* können nach dem einzelnen Schlüsselwort `WITH_CTE` angegeben werden. Jede *common-table-expression* kann auch in der `FROM`-Klausel der nachfolgenden *common-table-expressions* referenziert werden. Weitere Informationen siehe *SELECT - Cursor-Oriented, WITH CTE common-table-expression,...*.

select-expression	**INSERT Multiple Rows:**
	
	Diese Klausel gehört zum SQL Extended Set.
	
	Mit einem *select-expression* können Sie mehrere Reihen in eine Tabelle einfügen. Der *select-expression* wird ausgewertet und jede Reihe der Ergebnistabelle wird so behandelt, als ob die Werte in der Reihe als Werte in einer *VALUES-Klausel* einer Single-Row-Insert-Operation angegeben werden.
	
	Weitere Informationen siehe *Select Expressions*.
	
	Beispiel - INSERT Multiple Rows:
	
	`...`
	`INSERT INTO SQL-RETIREE (NAME,AGE,SEX)`
	`  SELECT LASTNAME, AGE, SEX`
	`  FROM SQL-EMPLOYEES`
	`  WHERE AGE > 60`
	`...`
	
	Anmerkung:
	Die Anzahl der tatsächlich eingefügten Reihen können Sie mit der Systemvariablen `*ROWCOUNT` überprüfen (siehe *Systemvariablen*-Dokumentation).

WITH RR/RS/CS	**WITH Isolation Level-Klausel:** Diese Klausel gehört zum SQL Extended Set. Diese Klausel ermöglicht die explizite Angabe des zum Auffinden der einzufügenden Reihen benutzten Isolation Level.

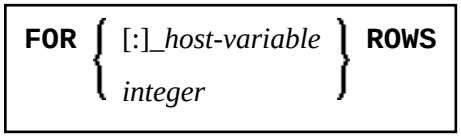
QUERYNO_*integer*	**QUERYNO-Klausel:** Diese Klausel gehört zum SQL Extended Set. Diese Klausel gibt explizit die in der EXPLAIN-Ausgabe und zur Ablaufverfolgung der Sätze für dieses Statement zu benutzende Anzahl an.

FOR-*n*-ROWS-Klausel:



Diese Klausel setzt sich aus den folgenden Subklauseln zusammen:

FOR [:] *hostvariable/integer* ROWS-Klausel:



Die Angabe dieser Klausel ist optional. Sie sollte nur angegeben werden, wenn

- die Compiler-Option DB2ARRY angegeben wird und
- mehrere Reihen von Arrays eingefügt werden sollen, die in der *insert-item-list* der VALUES-Klausel angegeben worden sind.

Wenn sie angegeben wird, legt die Option `[:] hostvariable/integer` die Anzahl der Reihen fest, die in die DB2-Tabelle eingefügt werden sollen, und zwar von den Arrays, die in der *insert-item-list* der VALUES-Klausel ab der ersten Ausprägung angegeben wurden.

Diese Klausel soll die Verarbeitungszeit der Programme verbessern, mittels derer Reihen von Natural-Arrays in einer Schleife eingefügt werden. Anhand dieser Klausel können die in den Arrays enthaltenen Reihen von einem SQL-Statement eingefügt werden.

Siehe Beispiel weiter unten.

Siehe auch den Teil *Natural for DB2* in der *Database Management System Interfaces*-Dokumentation.

ATOMIC-Klausel:

Diese Klausel gibt an, ob die Einfügung mehrerer Reihen von DB2 als eine Atomic-Operation behandelt werden sollte oder nicht.

Sie sollte nur angegeben werden, wenn

- die Compiler-Option DB2ARRY angegeben wird und
- mehrere Reihen von Arrays eingefügt werden sollen, die in der *insert-item-list* der *VALUES* -Klausel angegeben worden sind.

Syntax-Beschreibung:

ATOMIC	Gibt an, dass im Falle eines Fehlers keine Reihe in die Zieltabelle eingefügt wird. Dies ist die Voreinstellung.
NOT ATOMIC CONTINUE ON SQLEXCEPTION	Gibt an, dass im Falle von Fehlern alle Reihen eingefügt werden, für die keine Fehler aufgetreten sind, während diejenigen Reihen, für die Fehler aufgetreten sind, von DB2 entfernt werden.

In solchen Fällen zurückgegebene *sqlcodes* entnehmen sie der DB2 SQL REFERENCE.

Beispiel

```
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
```