

NDB - Natural DML Statements

This section summarizes particular points you have to consider when using Natural DML statements with DB2. Any Natural statement not mentioned in this section can be used with DB2 without restriction.

- BACKOUT TRANSACTION
 - DELETE
 - END TRANSACTION
 - FIND
 - GET
 - HISTOGRAM
 - READ
 - STORE
 - UPDATE
-

BACKOUT TRANSACTION

This statement undoes all database modifications made since the beginning of the last logical transaction. Logical transactions can start either after the beginning of a session or after the last SYNCPOINT, END TRANSACTION, or BACKOUT TRANSACTION statement.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- If this command is executed within a Natural stored procedure or Natural user-defined function (UDF), Natural for DB2 executes the underlying rollback operation. This sets the caller into a must-rollback state. If this command is executed within a Natural stored procedure or UDF for Natural error processing (implicit ROLLBACK), Natural for DB2 does not execute the underlying rollback operation, thus allowing the caller to receive the original Natural error.
- Under CICS, the BACKOUT TRANSACTION statement is translated into an EXEC CICS ROLLBACK command. However, in pseudo-conversational mode, only changes made to the database since the last terminal I/O are undone. This is due to CICS-specific transaction processing.

Note:

Be aware that with terminal input in database loops, Natural switches to conversational mode if no file server is used.

- In batch mode and under TSO, the BACKOUT TRANSACTION statement is translated into an SQL ROLLBACK command.

Note:

If running in a DSNMTV01 environment the BACKOUT TRANSACTION statement is ignored if the used PSB has been generated without the CMPAT=YES option.

- Under IMS/TM, the BACKOUT TRANSACTION statement is translated into an IMS Rollback (ROLB) command. However, only changes made to the database since the last terminal I/O are undone. This is due to IMS/TM-specific transaction processing.

As all cursors are closed when a logical unit of work ends, a BACKOUT TRANSACTION statement must not be placed within a database loop; instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own ROLLBACK command if the Natural program issues database calls, too. The calling Natural program must issue the BACKOUT TRANSACTION statement for the external program.

If a program tries to backout updates which have already been committed, for example by a terminal I/O, a corresponding Natural error message (NAT3711) is returned.

DELETE

The DELETE statement is used to delete a row from a DB2 table which has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement DELETE WHERE CURRENT OF *cursor-name*, which means that only the row which was read last can be deleted.

Example:

```
FIND EMPLOYEES WITH NAME = 'SMITH'
      AND FIRST_NAME = 'ROGER'
DELETE
```

Natural would translate the above Natural statements into SQL and assign a cursor name (for example, CURSOR1) as follows:

```
DECLARE CURSOR1 CURSOR FOR
SELECT FROM EMPLOYEES
      WHERE NAME = 'SMITH' AND FIRST_NAME = 'ROGER'
DELETE FROM EMPLOYEES
      WHERE CURRENT OF CURSOR1
```

Both the SELECT and the DELETE statement refer to the same cursor.

Natural translates a DML DELETE statement into an SQL DELETE statement in the same way it translates a FIND statement into an SQL SELECT statement.

A row read with a FIND SORTED BY cannot be deleted due to DB2 restrictions explained with the FIND statement. A row read with a READ LOGICAL cannot be deleted either.

DELETE when using the File Server

If a row rolled out to the file server is to be deleted, Natural rereads automatically the original row from the database to compare it with its image stored in the file server. If the original row has not been modified in the meantime, the DELETE operation is performed. With the next terminal I/O, the transaction is terminated, and the row is deleted from the actual database.

If the DELETE operates on a scrollable cursor, the row on the file server is marked as DELETE hole and is deleted from the base table.

However, if any modification is detected, the row will not be deleted and Natural issues the NAT3703 error message for non-scrollable cursors.

If the DELETE operates on a scrollable cursor, NDB simulates SQLCODE -224 (THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING) for DB2 compliance.

If the DELETE operates on a scrollable cursor and the row has become a hole, NDB simulates SQLCODE 0E222 (AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE).

Since a DELETE statement requires that Natural rereads a single row, a unique index must be available for the respective table. All columns which comprise the unique index must be part of the corresponding Natural view.

END TRANSACTION

This statement indicates the end of a logical transaction and releases all DB2 data locked during the transaction. All data modifications are committed and made permanent.

How the statement is translated and which command is actually issued depends on the TP-monitor environment:

- If this command is executed from a Natural stored procedure or UDF, Natural for DB2 does not execute the underlying commit operation. This allows the stored procedure or UDF to commit updates against non DB2 databases.
- Under CICS, the END TRANSACTION statement is translated into an EXEC CICS SYNCPOINT command. If the file server is used, an implicit end-of-transaction is issued after each terminal I/O. This is due to CICS-specific transaction processing in pseudo-conversational mode.
- In batch mode and under TSO, the END TRANSACTION statement is translated into an SQL COMMIT WORK command.

Note:

If running in a DSNMTV01 environment the END TRANSACTION statement is ignored if the used PSB has been generated without the CMPAT=YES option.

- Under IMS/TM, the END TRANSACTION statement is not translated into an IMS CHKP call, but is ignored. Due to IMS/TM-specific transaction processing, an implicit end-of-transaction is issued after each terminal I/O.

Except when used in combination with the SQL WITH HOLD clause (see SELECT in Natural SQL Statements), an END TRANSACTION statement must not be placed within a database loop, since all cursors are closed when a logical unit of work ends. Instead, it has to be placed outside such a loop or after the outermost loop of nested loops.

If an external program written in another standard programming language is called from a Natural program, this external program must not contain its own COMMIT command if the Natural program issues database calls, too. The calling Natural program must issue the END TRANSACTION statement for the external program.

Note:

With DB2, the END TRANSACTION statement cannot be used to store transaction data.

FIND

The FIND statement corresponds to the SQL SELECT statement.

Example:

Natural statements:

```
FIND EMPLOYEES WITH NAME = 'BLACKMORE'
      AND AGE EQ 20 THRU 40
OBTAIN PERSONNEL_ID NAME AGE
```

Equivalent SQL statement:

```
SELECT PERSONNEL_ID, NAME, AGE
FROM EMPLOYEES
WHERE NAME = 'BLACKMORE'
      AND AGE BETWEEN 20 AND 40
```

Natural internally translates a FIND statement into an SQL SELECT statement. The SELECT statement is executed by an OPEN CURSOR statement followed by a FETCH command. The FETCH command is executed repeatedly until either all records have been read or the program flow exits the FIND processing loop. A CLOSE CURSOR command ends the SELECT processing.

The WITH clause of a FIND statement is converted to the WHERE clause of the SELECT statement. The basic search criterion for a DB2 table can be specified in the same way as for an Adabas file. This implies that only database fields which are defined as descriptors can be used to construct basic search criteria and that descriptors cannot be compared with other fields of the Natural view (that is, database fields) but only with program variables or constants.

Note:

As each database field (column) of a DB2 table can be used for searching, any database field can be defined as a descriptor in a Natural DDM.

The WHERE clause of the FIND statement is evaluated by Natural **after** the rows have been selected via the WITH clause. Within the WHERE clause, non-descriptors can be used and database fields can be compared with other database fields.

Note:

DB2 does not have sub-, super-, or phonetic descriptors.

A FIND NUMBER statement is translated into a SELECT statement containing a COUNT(*) clause. The number of rows found is returned in the Natural system variable *NUMBER as described in Natural System Variables.

The FIND UNIQUE statement can be used to ensure that only one record is selected for processing. If the FIND UNIQUE statement is referenced by an UPDATE statement, a non-cursor (Searched) UPDATE operation is generated instead of a cursor-oriented (Positioned) UPDATE operation. Therefore, it can be used if you want to update a DB2 primary key. It is, however, recommended to use Natural SQL Searched UPDATE statement to update a primary key.

In static mode, the FIND NUMBER and FIND UNIQUE statements are translated into a SELECT SINGLE statement as described in the section Natural SQL Statements.

The FIND FIRST statement cannot be used. The PASSWORD, CIPHER, COUPLED and RETAIN clauses cannot be used either.

The SORTED BY clause of a FIND statement is translated into the SQL SELECT ... ORDER BY clause, which follows the search criterion. Because this produces a read-only result table, a row read with a FIND statement that contains a SORTED BY clause cannot be updated or deleted.

A limit on the depth of nested database loops can be specified at installation time. If this limit is exceeded, a Natural error message is returned.

Note for DB2 for OS/390 Version 7:

If a processing limit is specified as a constant integer number, for example, FIND (5), the limitation value will be translated into a FETCH FIRST integer ROWS ONLY clause in the generated SQL string.

FIND when using the File Server

As far as the file server is concerned, there are no programming restrictions with selection statements. It is, however, recommended to make yourself familiar with its functionality considering performance and file server space requirements.

GET

This statement is ISN-based and therefore cannot be used with DB2 tables.

HISTOGRAM

The HISTOGRAM statement returns the number of rows in a table which have the same value in a specific column. The number of rows is returned in the Natural system variable *NUMBER as described in Natural System Variables.

Example:

Natural statements:

```
HISTOGRAM EMPLOYEES FOR AGE  
OBTAIN AGE
```

Equivalent SQL statement:

```
SELECT COUNT(*), AGE FROM EMPLOYEES  
WHERE AGE > -999  
GROUP BY AGE  
ORDER BY AGE
```

Natural translates the HISTOGRAM statement into an SQL SELECT statement, which means that the control flow is similar to the flow explained for the FIND statement.

READ

The READ statement can also be used to access DB2 tables. Natural translates a READ statement into an SQL SELECT statement.

READ PHYSICAL and READ LOGICAL can be used; READ BY ISN, however, cannot be used, as there is no DB2 equivalent to Adabas ISNs. The PASSWORD and CIPHER clauses cannot be used either.

Since a READ LOGICAL statement is translated into a SELECT ... ORDER BY statement - which produces a read-only table -, a row read with a READ LOGICAL statement cannot be updated or deleted (see Example 1). The start value can only be a constant or program variable; any other field of the Natural view (that is, any database field) cannot be used.

A READ PHYSICAL statement is translated into a SELECT statement without an ORDER BY clause and can therefore be updated or deleted (see Example 2).

Example 1:

Natural statements:

```
READ PERSONNEL BY NAME
OBTAIN NAME FIRSTNAME DATEOFBIRTH
```

Equivalent SQL statement:

```
SELECT NAME, FIRSTNAME, DATEOFBIRTH FROM PERSONNEL
WHERE NAME >= ' '
ORDER BY NAME
```

Example 2:

Natural statements:

```
READ PERSONNEL PHYSICAL
OBTAIN NAME
```

Equivalent SQL statement:

```
SELECT NAME FROM PERSONNEL
```

If the READ statement contains a WHERE clause, this clause is evaluated by the Natural processor **after** the rows have been selected according to the descriptor value(s) specified in the search criterion.

NDB Features under DB2 for OS390 Version 7

Processing Limit

If a processing limit is specified as a constant integer number, for example, READ (5), in the SQL string generated, the value that defines the limitation will be translated into the clause

FETCH FIRST *integer* ROWS ONLY

Cursors for DB2 Clauses

NDB uses insensitive scrollable cursors to process the following READ statement:

READ .. [IN] [LOGICAL] VARIABLE/DYNAMIC *operand5* [SEQUENCE]

NDB uses insensitive scrollable cursors to process the READ statement below. If relating to a Positioned UPDATE or DELETE, NDB uses insensitive static cursors.

READ .. [IN] [PHYSICAL] DESCENDING/VARIABLE/DYNAMIC *operand5* [SEQUENCE]

operand5:

Value 'A' will be translated into a FETCH FIRST/NEXT DB2 access, and Value 'D' into a FETCH LAST/PRIOR DB2 access.

READ when using the File Server

As far as the file server is concerned there are no programming restrictions with selection statements. It is, however, recommended to make yourself familiar with its functionality considering performance and file server space requirements.

STORE

The STORE statement is used to add a row to a DB2 table. The STORE statement corresponds to the SQL statement INSERT.

Example:

Natural statement:

```
STORE RECORD IN EMPLOYEES
  WITH PERSONNEL_ID = '2112'
      NAME           = 'LIFESON'
      FIRST_NAME    = 'ALEX'
```

Equivalent SQL statement:

```
INSERT INTO EMPLOYEES (PERSONNEL_ID, NAME, FIRST_NAME)
VALUES ('2112', 'LIFESON', 'ALEX')
```

The PASSWORD, CIPHER and USING/GIVING NUMBER clauses cannot be used.

UPDATE

The Natural DML UPDATE statement updates a row in a DB2 table which has been read with a preceding FIND, READ, or SELECT statement. It corresponds to the SQL statement UPDATE WHERE CURRENT OF *cursor-name* (Positioned UPDATE), which means that only the row which was read last can be updated.

UPDATE when using the File Server

If a row rolled out to the file server is to be updated, Natural automatically rereads the original row from the database to compare it with its image stored in the file server. If the original row has not been modified in the meantime, the UPDATE operation is performed. With the next terminal I/O, the transaction is terminated and the row is definitely updated on the database.

If the UPDATE operates on a scrollable cursor, the row on the file server and the row in the base table are updated. If the row no longer qualifies for the search criteria of the related SELECT statement after the update, the row is marked as UPDATE hole on the file server.

However, if any modification is detected, the row will not be updated and Natural issues the NAT3703 error message for non-scrollable cursors.

If the UPDATE operates on a scrollable cursor, NDB simulates SQLCODE -224 (THE RESULT TABLE DOES NOT AGREE WITH THE BASE TABLE USING) for DB2 compliance.

If the UPDATE operates on a scrollable cursor and the row has become a hole, NDB simulates SQLCODE 0E222 (AN UPDATE OR DELETE OPERATION WAS ATTEMPTED AGAINST A HOLE).

Since an UPDATE statement requires rereading a single row by Natural, a unique index must be available for this table. All columns which comprise the unique index must be part of the corresponding Natural view.

UPDATE with FIND/READ

As explained with the FIND statement, Natural translates a FIND statement into an SQL SELECT statement. When a Natural program contains a DML UPDATE statement, this statement is translated into an SQL UPDATE statement and a FOR UPDATE OF clause is added to the SELECT statement.

Example:

```
FIND EMPLOYEES WITH SALARY < 5000
  ASSIGN SALARY = 6000
  UPDATE
```

Natural would translate the above Natural statements into SQL and assign a cursor name (for example, CURSOR1) as follows:

```
DECLARE CURSOR1 CURSOR FOR
SELECT SALARY FROM EMPLOYEES WHERE SALARY < 5000
  FOR UPDATE OF SALARY
UPDATE EMPLOYEES SET SALARY = 6000
  WHERE CURRENT OF CURSOR1
```

Both the SELECT and the UPDATE statement refer to the same cursor.

Due to DB2 logic, a column (field) can only be updated if it is contained in the FOR UPDATE OF clause; otherwise updating this column (field) is rejected. Natural includes automatically all columns (fields) into the FOR UPDATE OF clause which have been modified anywhere in the Natural program or which are input fields as part of a Natural map.

However, a DB2 column is not updated if the column (field) is marked as "not updateable" in the Natural DDM. Such columns (fields) are removed from the FOR UPDATE OF list without any warning or error message. The columns (fields) contained in the FOR UPDATE OF list can be checked with the LISTSQL command.

The Adabas short name in the Natural DDM determines whether a column (field) can be updated.

The following table shows the ranges that apply:

Short-Name Range	Type of Field
AA - N9	non-key field that can be updated
Aa - Nz	non-key field that can be updated
OA - O9	primary key field
PA - P9	ascending key field that can be updated
QA - Q9	descending key field that can be updated
RA - X9	non-key field that cannot be updated
Ra - Xz	non-key field that cannot be updated
YA - Y9	ascending key field that cannot be updated
ZA - Z9	descending key field that cannot be updated
1A - 9Z	non-key field that cannot be updated
1a - 9z	non-key field that cannot be updated

Be aware that a primary key field is never part of a FOR UPDATE OF list. A primary key field can only be updated by using a non-cursor UPDATE operation (see also UPDATE in the section Natural SQL Statements).

A row read with a FIND statement that contains a SORTED BY clause cannot be updated (due to DB2 limitations as explained with the FIND statement). A row read with a READ LOGICAL cannot be updated either (as explained with the READ statement).

If a column is to be updated which is redefined as an array, it is strongly recommended to update the whole column and not individual occurrences; otherwise, results are not predictable. To do so, in reporting mode you can use the OBTAIN statement (as described in the Natural Statements documentation), which must be applied to all field occurrences in the column to be updated. In structured mode, however, all these occurrences must be defined in the corresponding Natural view.

The data locked by an UPDATE statement are released when an END TRANSACTION (COMMIT WORK) or BACKOUT TRANSACTION (ROLLBACK WORK) statement is executed by the program.

Note:

If a length indicator field or NULL indicator field is updated in a Natural program without updating the field (column) it refers to, the update of the column is not generated for DB2 and thus no updating takes place.

UPDATE with SELECT

In general, the DML UPDATE statement can be used in both structured and reporting mode. However, after a SELECT statement, only the syntax defined for Natural structured mode is allowed:

```
UPDATE [ RECORD ] [ IN ] [ STATEMENT ] [( r )]
```

This is due to the fact that in combination with the SELECT statement, the DML UPDATE statement is only allowed in the special case of:

```
...
SELECT ...
  INTO VIEW view-name
  ...
```

Thus, only a whole Natural view can be updated; individual columns (fields) cannot.

Example:

```
DEFINE DATA LOCAL
01 PERS VIEW OF SQL-PERSONNEL
  02 NAME
  02 AGE
END-DEFINE

SELECT *
  INTO VIEW PERS
  FROM SQL-PERSONNEL
  WHERE NAME LIKE 'S%'

  IF NAME = 'SMITH'
    ADD 1 TO AGE
  UPDATE
  END-IF

END-SELECT
...
```

In combination with the DML UPDATE statement, any other form of the SELECT statement is rejected and an error message is returned.

In all other respects, the DML UPDATE statement can be used with the SELECT statement in the same way as with the Natural FIND statement described earlier in this section and in the Natural Statements documentation.