DBS_UPDATE

DB_UPDATE and DBS_UPDATE actions

Function

Modification of the given existing row (rows) in the database.

Declaration

```
 DB_UPDATE handleIdent_Int, rowIdent, retCodeIdent_Int [WHERE
strExpression_Str [BINDIN varIdent1, varIdent2, ... ]] [ORAHINT
hintIdent_Str]

 DB_UPDATE handleIdent_Int, rowIdent, retCodeIdent_Int [WHERE
strExpression_Str [BINDIN structRowIdent]] [ORAHINT hintIdent_Str]

 DBS_UPDATE dbObjIdent, rowIdent, retCodeIdent_Int [WHERE
strExpression_Str [BINDIN varIdent1, varIdent2, ... ]] [TRANS
transHandle_Int] [ORAHINT hintIdent_Str]

 DBS_UPDATE dbObjIdent, rowIdent, retCodeIdent_Int [WHERE
strExpression_Str [BINDIN structRowIdent]] [TRANS transHandle_Int]
[ORAHINT hintIdent_Str]
```

or

```
DB_UPDATE handleIdent_Int, structIdent, retCodeIdent_Int [ORAHINT
hintIdent_Str]

DBS_UPDATE dbObjIdent, structIdent, retCodeIdent_Int [TRANS
transHandle_Int] [ORAHINT hintIdent_Str]
```

Parameters

handleIdent_Int	in	Identifier of *Int* type (handle) of the connection with the database (DB_CONNECT).
dbObjIdent	in	Reference to an object of Database table type.
rowIdent	in	One structure row identifier (a row to update).
structIdent	in	Identifier of entire structure (the rows to update).
retCodeIdent_Int	out	Return value of *Int* type - action success.
strExpression_Str	in	Expression of *String* type, which identifies modified rows. If the expression is parameterized, the keyword **BINDIN** and the values of parameters (*structRowIdent* or *varIdent1*, *varIdent2*, ...) are mandatory.
varIdent1, varIdent2, ...	in	List of objects, constants or local variables, which will specify the values of parameters of parameterized SQL expression *strExpression_Str*.
structRowIdent	in	Reference to a row of local variable of *Record* type or to a row of structured variable. The row's values will specify the values of parameters of parameterized SQL expression *strExpression_Str*.
transHadle_Int	in	Identifier of the *Connection* to the database.
hintIdent_Str	in	Expression of *String* type that defines Oracle SQL hint. It is used as an instruction for the performance optimizer of SQL command. The value is used without the opening and terminating characters /*+ <*orahint*> */. The example is mentioned here.

Return code

The value of the parameter *transHandle_Int*. See the table of error codes. It is possible to get extended error information.

Description

The database must be opened with the access _DB_MODIFY. Analogous to the action DB_READ (DBS_READ), modified rows may be determined by two ways:

1. Setting the key items in a structure row (`rowIdent`) or in the entire structure (`structIdent`). The setting must be performed before the action **DB_UPDATE**. The respective items must be marked as the key during the definition of an object of a Table type. Then in the database, a row which key items values are equal to the key items of `rowIdent` is found and is modified. The row or structure to be updated must be of a proper type of structure. The values of all key items in `rowIdent` must be valid, otherwise the update will fail, end with an error, and % GetLastExtErrorCode() will return error 667.
2. The value of the expression `strExpression_Str` which represents **WHERE** clause for the SQL command **UPDATE** performing a database change. In this case, a key item value is not used.

If the `strExpression_Str` expression is not specified (or is empty) or the key is not defined when defining a Table type object, the action will return the error _ERR_TRANS_ERROR (11) and the variable _ERR_NR_TRANS_EX will have the value BAD_CONDITION.

Note: we recommend using the version without the `strExpression_Str` expression - when using the expression, Oracle temporarily locks the updated table as well as all tables into which the updated table has foreign indexes. This can cause a blocked session if another transaction in progress has updated any rows in these tables. The blocking will last until the ongoing transaction is completed.

The advantage of the action **DBS_UPDATE** during the work with a table is the possibility to leave out its opening and closing (simpler notation).

For D2000 v5.00: A disadvantage of the action **DBS_UPDATE** is in speed. Each **DBS_UPDATE** call results in necessity to open and close the database in DBManager - it can be a time-consuming operation and it is a comparatively nonstandard method in term of databases.
The need to open and close the database may be eliminated in the scope of transaction processing so that the command is followed by the parameter

```
TRANS
```

.

For D2000 v6.00 and higher: DBManager optimization (connection recycling, predefined connections) causes, that the action **DBS_UPDATE** is executes as quick as the action **DB_UPDATE** and moreover there is saved a time required for execution of the action **DB_CONNECT** to open the database.

The need to open and close the database may be eliminated in the scope of transaction processing so that the command is followed by the parameter

```
TRANS
```

.

Example

Work with a database table (actions DB_ ...).

Related topics

DB_CONNECT
DB_DELETE
DB_DISCONNECT
DB_INSERT
DB_INSUPD
DB_READ
DB_READ_BLOB
DB_UPDATE_BLOB

DB_TRANS_OPEN
DB_TRANS_COMMIT
DB_TRANS_ROLLBACK
DB_TRANS_CLOSE

All database related actions

Related pages:

Script actions