SQL_PREPARE

SQL_PREPARE action

Function

The action prepares the execution of the SQL command **SELECT**.

Declaration

```
 SQL_PREPARE handleIdent_Int, retCodeIdent_Int, selectStringExpr BIND
_locVar1, _locVar2, ...


 SQL_PREPARE handleIdent_Int, retCodeIdent_Int, selectStringExpr BIND
_locVarRowIdent


 SQL_PREPARE handleIdent_Int, retCodeIdent_Int, selectStringExpr BIND
_locVarRecordIdent


 SQL_PREPARE handleIdent_Int, retCodeIdent_Int, selectStringExpr BINDOUT
_locVar1, _locVar2, ...


 SQL_PREPARE handleIdent_Int, retCodeIdent_Int, selectStringExpr BINDOUT
_locVarRowIdent


 SQL_PREPARE handleIdent_Int, retCodeIdent_Int, selectStringExpr BINDOUT
_locVarRecordIdent
```

Parameters

handleIdent_Int	in	Identifier - the unique number (handle) of a connection.
retCodeIdent_Int	out	Return code identifier.
selectStringExpr	in	Expression of the Text type.
_locVar1, _locVar2, ...	in	List of local variables.
_locVarRowIdent	in	Reference to a row of local variable of the *Record* type.
_locVarRecordIdent	in	Identifier of local variable of the *Record* type.

Return code

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

Description

Reading a database by the command SELECT is implemented in two or three phases.
The first (preparatory) phase is executed by the action **SQL_PREPARE**. The command SELECT, defined by a value of the expression *selectStringExpr*, is prepared (and if the keyword **BINDOUT** is not used, then also executed) in the database (specified by a value of the identifier *handleIdent_Int*). Success of the action is indicated by a value of *retCodeIdent_Int*.
The second phase is required if the keyword **BINDOUT** was used. This keyword means that the expression *selectStringExpr* used parametrization and it is necessary to use the command SQL_BINDIN to specify the input parameters of the expression *selectStringExpr* before actual execution of the SQL statement.
The last phase is the sequential reading of the rows, prepared by the command SELECT, using the action SQL_FETCH.

Values read are saved into local variables listed after the keyword **BIND** or **BINDOUT** of the action **SQL_PREPARE**. There are three possible **variants**:

1. **List of non-structured local variables.**
   Reading is executed row by row into local variables, which are listed after the keyword **BIND** or **BINDOUT**.

2. **Reference to one row of local variable of the *Record* type.**
   Reading is executed one row into one row of the local variable. The structure of data that are read must be the same as the structure of the local variable.

3. **Reference to local variable of the the *Record* type.**
   Reading is executed either one or more rows of the local variable. Its size may be changed as

necessary before the result. The structure of data that are read must be the same as the structure of the local variable.

One reading (gained by the action SQL_CONNECT) may be active just for one handle. The action SQL_PREPARE will cancel the validity of the previous action. The action SQL_FREE allows to finish a reading.

Note: by using parameterization it is possible to make the work of SQL database easier, because the preparation (compilation) of parameterized SQL query will be performed only once (by the action **SQL_PREPARE**). Consequently, the values of parameters must be specified by the action SQL_BINDIN (which will also execute the SQL command) and then the action SQL_FETCH may be called once or more times to obtain the results. Then it is possible to set new values of the parameters and re-execute the SQL command by repeating the action SQL_BINDIN and obtain the new results by one or more calls of the action SQL_FETCH.
By proper setting of the database parameters (e.g. Oracle: *session_cached_cursors*) it is possible to ensure recycling of cursors (compiled statements) between the calls of **SQL_PREPARE**.

Example

Work with a database (actions SQL_ ...)

```
 BOOL _useBinding = @TRUE ; use parameterized SQL command
 INT  _handle    ; handle to database
 INT  _retCode   ; return code
 TEXT _name      ; product name
 TEXT _type      ; product type

                 ; parameterized SQL command
 TEXT _sqlPar =  "SELECT Name, Type FROM Products WHERE ID>= #PAR# AND
ID<= #PAR#"

                 ; non-parameterized SQL command
 TEXT _sqlNpar =  "SELECT Name, Type FROM Products WHERE ID>= 1 AND ID<=
100"

 SQL_CONNECT MyDatabase, _handle, _retCode

 IF _useBinding THEN ; parameterized alternative
   SQL_PREPARE _handle, _retCode, _sqlPar BINDOUT _name, _type
   SQL_BINDIN  _handle, _retCode, 1, 100 ; read all products between 1 and
100
 ELSE ; non-parametrized alternative
   SQL_PREPARE _handle, _retCode, _sqlNpar BIND _name, _type
 ENDIF

 DO_LOOP
   SQL_FETCH _handle, _retCode
   EXIT_LOOP _retCode #  _ERR_NO_ERROR
   ; data processing goes here
 END_LOOP

 SQL_FREE _handle
 SQL_DISCONNECT _handle
```

Related topics

DB_TRANS_OPEN
DB_TRANS_COMMIT
DB_TRANS_ROLLBACK
DB_TRANS_CLOSE

SQL_CONNECT
SQL_DISCONNECT
SQL_EXEC_DIRECT
SQL_EXEC_PROC

SQL_BINDIN
SQL_FETCH
SQL_FREE

SQL_SELECT

All database related actions.

Related pages:

Script actions