SQL_SELECT

SQL_SELECT action

Function

The action executes the SQL command SELECT.

Declaration

```
SQL_SELECT dbObjIdent, retCodeIdent_Int, [maxRowsIdent_Int],
selectIdent_String [TRANS transHandle_Int]  BIND ...
```

or

```
SQL_SELECT connectString, retCodeIdent_Int, [maxRowsIdent_Int],
selectIdent_String  ON dbManIdent  BIND ...
```

The key word BIND must be followed by one of these parameters:
BIND _locVar1, _locVar2, ...
BIND _locVarRowIdent
BIND _locVarRecordIdent

Parameters

dbObjIdent	in	Reference to an object of Database or Database table type.
connectString	in	Identifier of *Text* type or a text constant containing a connect string to the database.
retCodeIdent_Int	out	Identifier of *Int* type - return code.
maxRowsIdent_Int	in	Identifier of *Int* type - number of rows read en bloc (if it is not entered the default value is 1).
selectIdent_String	in	Identifier of *String* type.
_locVar1, _locVar2, ...	in	List of local variable.
_locVarRowIdent	in	Reference to row of local variable of *Record* type.
_locVarRecordIdent	in	Identifier of local variable of *Record* type.

Return code

The value of the parameter *retCodeIdent_Int* - see the table of error codes. It is possible to get extended error information.

Description

The action SQL_SELECT enables to make SQL command SELECT as simple as possible. It combines the actions SQL_CONNECT, SQL_PREPARE, SQL_FETCH and SQL_DISCONNECT.

A database, in which the SQL command SELECT will be executed, can be defined by:

1. parameter *dbObjIdent* - represents the object of *Database* or *Table* type. User can enter the parameter (optional) *transHandle_Int* that defines a database connection (the value of parameter has been gained by previous calling the action DB_TRANS_OPEN), within it the SQL command will be executed,
   If the parameter *transHandle_Int* is not defined, SQL command will be executed on one of the predefined database connections.
2. parameter *connectString* (described on page SQL_CONNECT action). It is necessary to specify the process DB Manager that will execute the SQL command.

The command SELECT is defined by value of parameter *selectIdent_String*. According to number of columns of SELECT command should be chosen the proper variant to show its result through the obligatory key word BIND. The meaning and explanation of the variants is mentioned in action SQL_PREPARE.

An optional identifier *maxRowsIdent_Int* defines the row counts of select which will be returned. If it is not defined its value is 1. The number of the rows that can be read depends also on the variant BIND. Details are in the description of action SQL_FETCH.

Maximum number of returned rows is limited by a Database configuration parameter Maximum returned rows. If number of available rows exceeds this limit, the local structure *_locVarRecordIdent* will contain first Maximum returned rows and *retCodeIdent_Int* will be set to error code _ERR_DATABASE_ROWS_LIMIT.

Note

he values returned byt the SQL statement are inserted into local variables according to their order (1st value to the first bound variable, 2nd to the second, etc.). This also applies when inserting into a local structured variable (i.e. column names are not taken into account). Potentially dangerous are SELECTs of the "SELECT *" type - if the order of the columns in the database differs from the order of the columns in the structured variable, the values will be inserted in the wrong columns. Therefore, we recommend explicitly naming columns in SQL query (SELECT COL1, COL2, COL3 ...).

If retrieving data from a single table or view, use DBS_READ instead of SQL_SELECT.

Example

In this example, there exists the object of *Database* type with the name **gnat_test**. There is a table OBJLIST within database in which the column *id* is figure.
A procedure Demo1 represents various (not all) variants of SQL_SELECT action.
The procedures Demo2 and Demo3 have the same function. Demo2 is implemented through the SQL_SELECT action and Demo3 is implemented by SQL_CONNECT action, SQL_PREPARE action, SQL_FETCH action and SQL_DISCONNECT action.

```
 PROCEDURE Demo1

  INT  _retCod
  INT _maxId

   ; select through the object of Database type
  SQL_SELECT gnat_test, _retCode,,  "select max(id) from OBJLIST" BIND
_maxId
  IF _retCode = _ERR_NO_ERROR THEN
  ; variable _maxId contains the value
   ELSE
  ; error
  ENDIF

  ; select through the object of Table type
  SQL_SELECT DB.OBJLIST, _retCode,,  "select max(id) from OBJLIST" BIND
_maxId
  IF _retCode = _ERR_NO_ERROR  THEN
  ; variable _maxId contains the value
   ELSE
  ; error
  ENDIF

  INT _transHandle
   ; opening of the transaction through the object of Database type
  DB_TRANS_OPEN gnat_test, _transHandle, _retCode
  IF _retCode = _ERR_NO_ERROR  THEN
    ; transaction has been opened
    ; select within the transaction
    SQL_SELECT DB.OBJLIST, _retCode,, "select max(id) from OBJLIST"  TRANS
_transHandle BIND _maxId
    IF _retCode = _ERR_NO_ERROR THEN
    ; variable _maxId contains the value
    ELSE
    ; error
    ENDIF

    DB_TRANS_CLOSE _transHandle
  ELSE
    ; error
   ENDIF

  ; select with the connection
  SQL_SELECT "connection String", _retCode,, "select max(id) from OBJLIST"
ON SELF.DBM BIND _maxId
  IF _retCode = _ERR_NO_ERROR THEN
  ; variable _maxId contains the value
  ELSE
  ; error
  ENDIF

 END Demo1


 PROCEDURE Demo2(INT _maxId)
```

```
 INT _retCode

  _maxId := %SetInvalid(0)
  ; select through the object of Database type
  SQL_SELECT gnat_test, _retCode,, "select max(id) from OBJLIST" BIND
_maxId
  IF _retCode = _ERR_NO_ERROR THEN
  ; variable _maxId contains the value
  ELSE
  ; error
  ENDIF

  END Demo2


 PROCEDURE Demo3(INT _maxId)
  INT _retCode
  INT _handle
  _maxId := %SetInvalid(0)
  ;
  SQL_CONNECT gnat_test, _handle, _retCode
  IF _retCode <> _ERR_NO_ERROR THEN
   RETURN ; error
  ENDIF

  SQL_PREPARE _handle, _retCode, "select max(id) from OBJLIST" BIND _maxId
  IF _retCode <> _ERR_NO_ERROR THEN
   RETURN ; error
  ENDIF

  SQL_FETCH _handle, _retCode
  IF _retCode <> _ERR_NO_ERROR THEN
   RETURN ; error
  ENDIF

  SQL_DISCONNECT _handle

 END Demo3


 BEGIN

  CALL Demo1

  INT _max
  CALL Demo2(_max)

  CALL Demo3(_max)
 END
```

Related topics

DB_TRANS_OPEN
DB_TRANS_COMMIT
DB_TRANS_ROLLBACK
DB_TRANS_CLOSE

SQL_CONNECT
SQL_EXEC_DIRECT
SQL_EXEC_PROC

SQL_PREPARE
SQL_BINDIN
SQL_FETCH
SQL_FREE

Actions for accessing a database.

Related pages:

Script actions