

# Example: Work with a database (actions SQL\_ ...)

There is defined the object [SD.MachineProduced](#) and DSN to the database Database of required structure. The process DbManager with the name SELF. DBM is running.

Database structure:

Tables:

Table name	Table structure		
Machines	Column name	Type	Description
	ID	Integer	Unique machine identifier
	Name	Char(50)	Machine name
Products	Column name	Type	Description
	ID	Integer	Unique product identifier
	Name	Char(50)	Product name
MachineProduced	Column name	Type	Description
	ID_MACHINE	Integer	Unique machine identifier
	ID_PRODUCT	Integer	Unique product identifier

The table **Machines** contains a list of machines, where a machine is represented by an unique identifier Machine.ID and optional name Machine.Name. The table **Products** contains a list of product types, where a product is represented by an unique identifier Product.ID and optional name Product.Name. The table **MachineProduced** includes a list containing, that a machine MachineProduced.ID\_MACHINE produced a product MachineProduced.ID\_PRODUCT.

In the example, there will be filled the tables Machine and Product. Then there will be randomly generated records about performed products on machines.

Types of products and quantities produced on the machine 1 will be detected by three methods.

```
INT _handle      ; handle to database
INT _retCode     ; return code

; Procedure performs SQL command. Failure will be written in Log database
PROCEDURE ExecSql(IN TEXT _sql)
    TEXT _errorMsg
    SQL_EXEC_DIRECT _handle, _retCode, _sql
    IF _retCode # _ERR_NO_ERROR THEN
        _errorMsg := "Error SQL_EXEC_DIRECT " + _sql
        LOG _errorMsg PRIORITY _LOG_PRTY_ERROR
    ENDIF
END ExecSql

; Inserting a machine definition into table
PROCEDURE Insert_Machine(IN INT _id, IN TEXT _name)
    TEXT _sql
    _sql := "INSERT INTO Machines VALUES (" + %IToStr(_id) + ", '" + _name + "'" )"
    CALL ExecSql(_sql)
END Insert_Machine

; Inserting a product definition into table
PROCEDURE Insert_Product(IN INT _id, IN TEXT _name)
    TEXT _sql
    _sql := "INSERT INTO Products VALUES (" + %IToStr(_id) + ", '" + _name + "'" )"
    CALL ExecSql(_sql)
END Insert_Product

; Filling the table Machines
PROCEDURE Fill_Machines
    INT _id
    TEXT _name
    TEXT _sql
```

```

_sql := "DELETE FROM MACHINES"
CALL ExecSql(_sql)

_id := 1
_name := "Machine 1"
CALL Insert_Machine(_id, _name)

_id := 2
_name := "Machine 2"
CALL Insert_Machine(_id, _name)

_id := 3
_name := "Machine 3"
CALL Insert_Machine(_id, _name)

_id := 4
_name := "Machine 4"
CALL Insert_Machine(_id, _name)

_id := 5
_name := "Machine 5"
CALL Insert_Machine(_id, _name)
END Fill_Machines

; Filling the table Products
PROCEDURE Fill_Products
  INT _id
  TEXT _name
  TEXT _sql

  _sql := "DELETE FROM PRODUCTS"
  CALL ExecSql(_sql)

  _id := 1
  _name := "Product 1"
  CALL Insert_Product(_id, _name)

  _id := 2
  _name := "Product 2"
  CALL Insert_Product(_id, _name)

  _id := 3
  _name := "Product 3"
  CALL Insert_Product(_id, _name)

  _id := 4
  _name := "Product 4"
  CALL Insert_Product(_id, _name)

  _id := 5
  _name := "Product 5"
  CALL Insert_Product(_id, _name)
END Fill_Products

; Inserting a record that Machine produced a product
PROCEDURE MachineProduced(IN INT _idMachine, IN INT _idProduct)
  TEXT _sql
  _sql := "INSERT INTO MachineProduced VALUES (" + %IToStr(_idMachine) + ", " + %IToStr(_idProduct) + ")"
  CALL ExecSql(_sql)
END MachineProduced

; Procedure accidentally generates records that Machine produced a product
PROCEDURE Fill_MachineProduced
  INT _idMachine
  INT _idProduct
  INT _idx
  TEXT _sql

  _sql := "DELETE FROM MachineProduced"

```

```

CALL ExecSql(_sql)

_idx := 1
DO_LOOP
  EXIT_LOOP _idx = 100
  _idMachine := %Rnd() * 4.0 + 1.0
  _idProduct := %Rnd() * 4.0 + 1.0
  CALL MachineProduced(_idMachine, _idProduct)
  _idx := _idx + 1
END_LOOP
END Fill_MachineProduced

; List of products and their quantity for "Machine 1" using SQL_PREPARE version 1 or 4
PROCEDURE Machine_1_LocVarList
  BOOL _useBinding = @TRUE ; use binding or not
  TEXT _sql
  TEXT _errorMsg
  TEXT _product
  INT _quantity

  IF _useBinding THEN ; alternative with binding (version 4)

    _sql := "SELECT Products.Name, Count(*) FROM Products, Machines, MachineProduced WHERE Machines.Name =
#PAR# AND "
    _sql := _sql + "Machines.ID=MachineProduced.ID_MACHINE AND Products.ID=MachineProduced.ID_PRODUCT GROUP BY
Products.Name"
    SQL_PREPARE _handle, _retCode, _sql BINDOUT _product, _quantity
    SQL_BINDIN _handle, _retCode, "Machine 1" ; set value of input parameter

  ELSE ; non-binding alternative (version 1)

    _sql := "SELECT Products.Name, Count(*) FROM Products, Machines, MachineProduced WHERE Machines.Name =
'Machine 1' AND "
    _sql := _sql + "Machines.ID=MachineProduced.ID_MACHINE AND Products.ID=MachineProduced.ID_PRODUCT GROUP BY
Products.Name"
    SQL_PREPARE _handle, _retCode, _sql BIND _product, _quantity

  ENDIF

  IF _retCode # _ERR_NO_ERROR THEN
    _errorMsg := "Error SQL_PREPARE " + _sql
    LOG _errorMsg PRIORITY _LOG_PRTY_ERROR
    RETURN
  ENDIF

  DO_LOOP
    SQL_FETCH _handle, _retCode ; reading 1 row of the result of SQL command SELECT into the variables
_product a _quantity
    EXIT_LOOP _retCode # _ERR_NO_ERROR
  END_LOOP

END Machine_1_LocVarList

; List of products and their quantity for "Machine 1" using SQL_PREPARE version 2 or 5
PROCEDURE Machine_1_LocRowIdent
  BOOL _useBinding = @TRUE ; use binding or not
  TEXT _par = "Machine 1" ; parameter value
  TEXT _sql
  TEXT _errorMsg
  RECORD NOALIAS (SD.MachineProduced) _produced

  IF _useBinding THEN ; alternative with binding (version 5)

    _sql := "SELECT Products.Name, Count(*) FROM Products, Machines, MachineProduced WHERE Machines.Name =
#PAR# AND "
    _sql := _sql + "Machines.ID=MachineProduced.ID_MACHINE AND Products.ID=MachineProduced.ID_PRODUCT GROUP BY
Products.Name"
    SQL_PREPARE _handle, _retCode, _sql BINDOUT _produced[1]
    SQL_BINDIN _handle, _retCode, _par ; set value of input parameter

```

```

ELSE ; non-binding alternative (version 2)

    _sql := "SELECT Products.Name, Count(*) FROM Products, Machines, MachineProduced WHERE Machines.Name =
'Machine 1' AND "
    _sql := _sql + "Machines.ID=MachineProduced.ID_MACHINE AND Products.ID=MachineProduced.ID_PRODUCT GROUP BY
Products.Name"
    SQL_PREPARE _handle, _retCode, _sql BIND _produced[1]

ENDIF
IF _retCode # _ERR_NO_ERROR THEN
    _errorMsg := "Error SQL_PREPARE " + _sql
    LOG _errorMsg PRIORITY _LOG_PRTY_ERROR
    RETURN
ENDIF

DO_LOOP
    SQL_FETCH _handle, _retCode ; reading 1 row of the result of SQL command SELECT the row nr.1 of the
variable _produced
    EXIT_LOOP _retCode # _ERR_NO_ERROR
END_LOOP
END Machine_1_LocRowIdent

; List of products and their quantity for "Machine 1" using SQL_PREPARE version 3 and 6
PROCEDURE Machine_1_LocRecIdent
    BOOL _useBinding = @TRUE ; use binding or not
    TEXT _par = "Machine 1" ; parameter value
    TEXT _sql
    TEXT _errorMsg
    INT _maxRows
    RECORD NOALIAS (SD.MachineProduced) _produced

    IF _useBinding THEN ; alternative with binding (version 6)

        _sql := "SELECT Products.Name, Count(*) FROM Products, Machines, MachineProduced WHERE Machines.Name =
#PAR# AND "
        _sql := _sql + "Machines.ID=MachineProduced.ID_MACHINE AND Products.ID=MachineProduced.ID_PRODUCT GROUP BY
Products.Name"
        SQL_PREPARE _handle, _retCode, _sql BINDOUT _produced
        SQL_BINDIN _handle, _retCode, _par ; set value of input parameter

    ELSE ; non-binding alternative (version 3)

        _sql := "SELECT Products.Name, Count(*) FROM Products, Machines, MachineProduced WHERE Machines.Name =
'Machine 1' AND "
        _sql := _sql + "Machines.ID=MachineProduced.ID_MACHINE AND Products.ID=MachineProduced.ID_PRODUCT GROUP BY
Products.Name"
        SQL_PREPARE _handle, _retCode, _sql BIND _produced

    ENDIF

    IF _retCode # _ERR_NO_ERROR THEN
        _errorMsg := "Error SQL_PREPARE " + _sql
        LOG _errorMsg PRIORITY _LOG_PRTY_ERROR
        RETURN
    ENDIF

    _maxRows := 10
    DO_LOOP
        SQL_FETCH _handle, _retCode, _maxRows ; reading at most 10 rows of the result of SQL command SELECT
into the variable _produced
        EXIT_LOOP _retCode # _ERR_NO_ERROR
    END_LOOP
END Machine_1_LocRecIdent

BEGIN
; Connecting to database
SQL_CONNECT "UID=dba;PWD=sql;DSN=Database", _handle, _retCode ON SELF.DBM
IF _retCode # _ERR_NO_ERROR THEN
    LOG "Error during connect to database" PRIORITY _LOG_PRTY_ERROR
END

```

```
ENDIF
; Filling the table Machines
CALL Fill_Machines
```

```
; Filling the table Products
CALL Fill_Products
```

```
; Filling the table MachineProduced
CALL Fill_MachineProduced

; Detecting types of products produced by "Machine 1"
CALL Machine_1_LocVarList

; Detecting types of products produced by "Machine 1"
CALL Machine_1_LocRowIdent

; Detecting types of products produced by "Machine 1"
CALL Machine_1_LocRecIdent
END
```

## Note

- Terminating a script by the action [END](#) (or any optional way) automatically closes all connections to the database.



### Related pages:

[Script actions](#)