Data Visualization System
Technical Documentation

The Study on Water Resources Management in the Hashemite Kingdom of Jordan

FINAL REPORT VOLUME VIII

SUPPORTING REPORT
FOR
PART-A “WATER RESOURCES MANAGEMENT MASTER PLAN”

Chapter 11 Visualization of Water Resources Management Master Plan

Table of Contents

Data Visualization System Technical Documentation

Table of contents page

1. Installation

1.1 Requirements

System requirements:

The program runs on a windows PC with the operating systems Win NT, Windows 95/98/2000

The MS Office products MS Access and MS Excel have to be installed. The Office version has to be 97 or higher.

Recommended hardware:

Processor speed:	>= 200 MHz
Ram:	>= 64 Mbyte
Network card:	>= 10 Mbit

Required Software:

Operating systems: Win NT, Windows 95/98/2000

MS Office (MS Access and MS Excel) 97 or 2000

Oracle ODBC Driver for the Oracle WIS at MOWI

1.2 Installation procedure

ODBC driver for ORACLE WIS

To enable the DVS for online access to the ORACLE WIS database, the ORACLE ODBC driver has to be installed on the local computer. The recommended database alias name is “**WIS**”, which is the default for the DVS installation. If you use a different alias name, refer to chapter 2.1. The installation of the driver has to be done by the ORACLE administrator. The driver might be installed already on your computer (See system panel/ODBC or system panel/32-bit ODBC).

DVS application & MapObjects

The DVS is a VBA programmed MS Access application programmed using Access 97. It includes MapObjects LT components and therefor needs a special system set-up. The set-up program copies not only the DVS application files to the target folder, it also distributes and registers the needed MapObjects LT libraries and components. See chapter 1.3 for details.

Use the Windows Explorer to locate the set-up program on the network’s path

N:\WIS\Modules\Install\DVS\First Install

Double-click on the Setup.exe file

It is recommended to use the default settings during the installation process, as installation under a path different from C:\Program Files will require some changes in the configuration file DVS.cfg (see chapter 2.1). You can use a folder name different from the name DVS97, for instance DVS.

After the copying process ended you will be asked to restart your system, if you installed the DVS the first time (to make the registry changes active).

After restarting (if it was necessary) you will find a new folder DVS (or DVS97) in your Windows start-up bar. According to the MS Access system on your computer you should start either DVS97 or DVS2000.

1.3 Files distributed with the application

File Name	Location	Description
dvs_description.doc	C:\Program Files\DVS	User Manual
dvs.cfg	C:\Program Files\DVS	Configuration file
dvs.ico	C:\Program Files\DVS	DVS icon
dvs_data.mdb	C:\Program Files\DVS	Offline database
DVS2000.mdb	C:\Program Files\DVS	DVS application for Access 2000
DVS97.mdb	C:\Program Files\DVS	DVS application for Access 97
Folder dvs_xls	C:\Program Files\DVS\	This folder contains the JICA Excel sheets for offline access
Folder spatialData	C:\Program Files\DVS\	This folder contains the spatial data (ArcView shapefiles) for offline access
ShapeLT20.dll*	Shares directory C:\Program Files\Common Files\ESRI	These files contain the functions and components for the GIS viewer of the DVS. They are part of the ESRI MapObjects LT 2.0 package and have to be distributed in order to make the MapObjects components available for a destination computer as a runtime version.
AIImageLT20.dll*		
libtiff.dll		
MOLT20.ocx*		
Pe.dll		
Sg.dll		
AFLT20.dll		
Dao3032.dll		
Ctl3d95.dll	Windows system directory C:\Windows\System (95/98/ME) C:\WINNT\System32 (NT/2000)	The MapObjects LT is based on the Microsoft OLE automation architecture. Several libraries have to be installed on the target computer. These files update the system in order to enable the operating system understanding the OLE language.
Ctl3dNT.dll		
Msvcirt.dll		
Msvcp60.dll		
Msvcrt.dll		
Msvcrt40.dll		
Oleaut32.dll		
Olepro32.dll*		
RegSvrt32.exe		
Stdole2.tlb*		
Mfc42.dll*		

***** File is registered in the Windows registry

2. Customizing the application

2.1 The configuration file DVS.cfg

The configuration file stores the local setting for your DVS copy, mainly the database ODBS alias and the path names for the used shape files and tables. To change any of the parameters (necessary only for installations different from the standard installation) you can either open the Access form F_Config or you enter the values directly with a text editor such as WordPad or Editor.

The following parameters are part of the configuration file:

```
#ShapePathOnline   n:\wis\spatial-data\vector data\shape files\
```

This path points to the common part of the network’s spatial data directory

```
#ShapePathOffline  C:\Program Files\DVS\spatialData\
```

This path points to the common part of the local spatial data directory (the data is copied to the computer during the first setup)

```
#ExcelPathOnline   N:\WIS\Modules\DVS\dvs_xls\
```

This path points to the Excel files (JICA) stored on the network.

```
#ExcelPathOffline  C:\Program Files\DVS\dvs_xls\
```

This path points to the Excel files (JICA) stored on the local machine.

```
#DBOwner           JORDAN
```

ORACLE table Owner. If this is changed to “TEST”, you will receive the figures based on the TEST-STP.

```
#DSNOnline         WIS
```

ORACLE WIS ODBC database Alias. If you do not know the alias name (by default WIS), see the system panel/ODBC or system panel/32-bit ODBC or ask an ORACLE or Windows administrator.

```
#DBNameOffline     C:\Program Files\DVS\dvs_data.mdb
```

Name and location of the MS Access database for working offline.

2.2 Updating the database for offline work

STP Data

The STP data is stored locally in the MS Access database DVS_Data, located in the main DVS directory (by default C:\Program Files\DVS). You can open this database and see the list of ORACLE tables which were imported to the database. You update a table by deleting it and importing the data again from ORACLE (File/Import Data etc...)

Spatial Data

The spatial data is stored locally in the sub-directory **SpatialData** in the DVS main path. Here you find the same directory structure you find on the network. You update a shapefile by copying its 3 components (*.shp, *.dbf, *.shx) to the according directory. That way, you overwrite the old version with a newer one.

3. Design of the application

3.1 General

The application consists of forms, tables, queries, macros and modules. These elements will be discussed more detailed in the next chapters. Here only some general remarks concerning them will follow.

The forms are the interface to the user, here he selects his field of interest and the geographical location and enters the constraints of his query. In many cases the same form is used for what might look like different actions to the user, but in fact only the labeling varies while the processing is the same. To achieve this flexibility there have been set up several tables steering for example which charts can be produced for a certain resource/demand, which external tables need to be accessed for generating a certain chart or which Excel files are the basis of the chart, which shape files may be used for this special selection.

In addition to the tables which contain a kind of configuration information for the forms there are some tables containing just a descriptive text for codes used in the Oracle tables that are not described in lookup tables, like translating “y” and “Y” to “Yes”. A last group of tables hold the contents of a selection and are of a rather temporary type. Most of them would be deleted automatically when they are no longer needed.

There are some predefined queries to facilitate the handling of cross tables which are needed for the processing of water transfer data. All other queries within the application are generated only temporarily according to the selections and constraints the user has entered.

Macros and modules support the execution of the program. They contain program code that normally is not unique to one form.

3.2 Forms

The Forms are listed in the sequence in which they normally would be accessed.

NWMP

The applications starts with the form NWMP where the user can decide which program related to the National Water Master Plan he wants to activate, the preprocessing modules or the digital visualization of the data. But up to now only the digital visualization can be run, the other option is meant for a future implementation.

F_connect

When the digital visualization has been chosen the user is asked to decide if he wants to retrieve data from the central Oracle database or from a local Access database. Depending on this choice also the shape files and the Excel files will be retrieved either from the network or from a local disk drive. For the connection to the Oracle database an user-id and a password is required which has to be established by the database administrator.

In this form also a general test will be performed if the paths of the shape files, the Excel files and – if it had been chosen – the local database can be accessed. If any of these is not available the form for editing the configuration data will open.

F_Config

The form for editing the configuration data opens automatically when a necessary path information or the directory is missing. This also happens if the configuration file is missing completely. In this case a new empty file will be generated.

Here all parameters of the configuration file DVS.CFG can be entered. The content of DVS.CFG will be replaced by the new entries. The parameters are described in detail in chapter 2.1.

SelectDT

When the data source has been chosen the user can select the data type (type of demand or resource) to be visualized. The available data types are stored in the internal table TDataType which allows a relatively easy adaptation of the displayed text and adding of further data types. However, modifications of the content of this table should be done with great caution, the codes should not at all be changed!

Depending on the code of the data type (the first character being R or D) it will be displayed in the column of the resources or the demands. At this stage of the program for resources and demands there can be five entries each.

SelectChart

Depending on the selected data type this form displays a list of the available chart types. These also are retrieved from an internal table, T_DT_CT, which contains the code of the data type, a code for the chart type and various information related to the process of generating the chart. With the exception of the chart titles none of the existing entries should be changed, otherwise the forms following might not deliver proper results!

With the selection of the chart type also the required external tables can be defined. All names of the (Oracle or Access) tables that are used are stored with an unique short code in the internal table TTableNames, without the information of the table owner which is provided by the configuration file DVS.CFG. The conjunction of the tables, the data types and the chart types is stored in the internal table T_DT_CT_TN. Existing entries should not be modified, otherwise the forms following might not be able to locate necessary tables!

All external tables that are linked when continuing with the spatial selection will be released when this form is closed to return to the previous one.

At this stage of the program a maximum of ten charts per resource or demand could be defined.

Map

In most cases the next step is the selection of the spatial units to be processed. (For some charts this step will be skipped.) The shape files for the spatial selection will be displayed depending on the data type and the chart type.

All names of the used shape files are stored in the internal table TShapeFiles. It contains a code for the file which also discerns between the type of the geographical information: polygon (01 – 20), point (21 – 40) or line (41 – 60). All modifications of the content of this table should be done with great caution, the codes should not at all be changed! The conjunction of the shape files and the data types is stored in the table T_DT_SF.

The user may choose between up to eight polygon shapes, up to five point shapes and up to three line shapes to be displayed as the basis of his selection.

The selected geographical units are exported to an internal table named SELECTION.

SelectObjects

This form allows the user to select/deselect the geographical items from a table. It can be activated and deactivated within the form Map.

SelectParam*

The forms starting with the characters "SelectParam" deal with the arrangement of the final chart. On them the user enters the constraints (where required) for the values to be retrieved from the database. The last characters of the name of the form (mostly one letter followed by a number) give a rough classification of the data. They are included in the internal table T_DT_CT which already has been mentioned in the description of the form *SelectChart*. The letter E stands for data retrieved from Excel files as provided by *"The study on Water Resources management in Jordan"* from JICA. The letter H indicates historical data from the database, the letter P projection data.

The names (and subdirectories) of the Excel files serving as a data source are stored in the internal table TExcelFiles. All modifications of the content of this table should be done with great caution, the codes should not at all be changed! The internal table T_DT_CT_XL contains the conjunction of the Excel files, the data types and the chart types.

The queries generated on the basis of the user’s selections and constraints are included in the description of the forms. As there are several hundreds of possibilities for different resulting SQL statements they have been tried to reduce to basic types that reappear with only a change of a name. Thus at some places a variable is referred (which is replaced during the program execution by the proper value). This is surrounded by ‘<>’. Explanations will be given with the descriptions.

If charts premise identical constraints they are arranged by the same form, even if the data types are different. This for example is the case with *SelectParamP1* which serves municipal as well as touristic demand projections as well as the projections of losses in the municipal water supply network. The individual forms of this large group now will be described in detail.

SelectParamE1

This form processes an Excel file containing historical population data and population projections, grouped by governorates. The user can either select a time span when he chose historical data, or one or more scenarios. The desired data for the previously geographically selected governorate(s) is copied from the original Excel file to a new one.

SelectParamE2

This form processes the Excel files for population, industrial and touristic growth rates, according to the previously selected data type. All three Excel files have an identical structure, otherwise it would not be possible to use the same method to access the data. The user can select one or more scenarios. As the data is not related to a geographic unit the preceding spatial selection has been skipped. The desired data is copied from the original Excel file to a new one.

SelectParamE3

This form processes an Excel file for consumption projections, grouped by governorates. The user can select one or more scenarios. The desired data for the previously geographically selected governorate(s) is copied from the original Excel file to a new one.

SelectParamE4

This form deals with water transferred between governorates. In one case the user accesses directly the Excel pivot table where he can select the governorates and water resource types he is interested in. In the other case form *SelectParamE4a* is activated for the presentation of he projection of the water transfer in major pipelines.

SelectParamE4a

This form processes the Excel files concerning the projections of water transfer between governorates in major pipelines. The table and the map are copied from the original file to a new one, according to the selected projection year.

SelectParamE5

This form retrieves data on local water production and water import and export from the summary table of a system of interrelated Excel files. The retrieved data for the previously geographically selected governorate is copied form the original table to a new one according to a sample file, and excessive data is removed, leaving only water production or import or export.

SelectParamE6

This form processes the Excel files concerning the projections of water resources potential within the governorates. The table and the map are copied from the original file to a new one, according to the water resource type.

SelectParamH1

This form generates the charts for municipal water use. Depending on the constraints the user enters and on the chart type (quarterly or annually billed water) the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

Following SQL statements result for the different cases:

Quarterly billed water without regard of the use type

```
SELECT
```

```
        a.quarter, 'Quarter '+cstr(a.quarter) AS qname, a.year,
        SUM(a.water_billed)/1000000 AS water
INTO
        temp_data
FROM
        water_use_billing a
WHERE
        cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
        a.year >= <Year1> AND a.year <= <Year2>
        GROUP BY
        a.quarter, a.year
```

Quarterly billed water for one use type

First a temporary table has to be created containing the codes for the use type.

```
SELECT
        bill_cat
INTO
        temp_cat
FROM
        water_use_bill_cat
WHERE
        use_type = <X>
```

Next the final query can be executed.

```
SELECT
        a.quarter, 'Quarter '+cstr(a.quarter) AS qname, a.year,
        SUM(a.water_billed)/1000000 AS water
INTO
        temp_data
FROM
        water_use_billing a
WHERE
        cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
        a.bill_cat IN (SELECT bill_cat FROM temp_cat) AND a.year >=
        <Year1> AND a.year <= <Year2>
GROUP BY
        a.quarter, a.year
```

Annually billed water without regard of the use type

```
SELECT
        s.<fldName>, s.<nameFld>, a.year,
        SUM(a.water_billed)/1000000 AS water
INTO
        temp_data
FROM
        selection s, water_use_billing a
WHERE
```

```
        cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
        s." & codeFld & " = cstr(a.sett_id) AND a.year >= <Year1>
        AND a.year <= <Year2>
GROUP BY
        s.<fldName>, s.<nameFld>, a.year
```

Annually billed water for one use type

First a temporary table has to be created containing the codes for the use type.

```
SELECT
        bill_cat
INTO
        temp_cat
FROM
        water_use_bill_cat
WHERE
        use_type = <X>
```

Next the final query can be executed.

```
SELECT
        s.<fldName>, s.<nameFld>, a.year,
        SUM(a.water_billed)/1000000 AS water
INTO
        temp_data
FROM
        selection s, water_use_billing a
WHERE
        cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
        a.bill_cat IN (SELECT bill_cat FROM temp_cat) AND
        s.<codeFld> = cstr(a.sett_id) AND a.year >= <Year1> AND
        a.year <= <Year2>
GROUP BY
        s.<fldName>, s.<nameFld>, a.year
```

Annually billed water for more than one use type

Because of a very large complexity of the query which can cause problems during execution the data for each selected use type have to be retrieved separately and added to the main temporary table which will be processed for the creation of the Excel chart. So, the following steps will be repeated for each selected use type.

First a temporary table has to be created containing the codes for the use type.

```
SELECT
        bill_cat
INTO
        temp_cat
FROM
        water_use_bill_cat
WHERE
        use_type = <X>
```

Next the final query can be executed.

```
SELECT
       b.code, b.description, a.year, SUM(a.water_billed)/1000000
       AS water
INTO
       temp_d1
FROM
       water_use_types_mwi b, water_use_billing a
WHERE
       cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
       a.bill_cat IN (SELECT bill_cat FROM temp_cat) AND b.code =
       <X> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
       b.code, b.description, a.year
```

Then the content of table temp_d1 will be added to table temp_data, and table temp_d1 can be deleted.

SelectParamH2

This form generates the charts for industrial water use. Depending on the constraints the user enters and on the chart type (quarterly or annually billed water) the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

Following SQL statements result for the different cases:

Quarterly billed water without regard of the water source

```
SELECT
       a.quarter, 'Quarter '+cstr(a.quarter) AS qname, a.year,
       SUM(a.water_billed)/1000000 AS water
INTO
       temp_data
FROM
       qurt_billed_industry a
WHERE
       a.facility_id IN (SELECT <codeFld> FROM selection) AND
       a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
       a.quarter, a.year
```

Quarterly billed water for one water source

```
 SELECT
       a.quarter, 'Quarter '+cstr(a.quarter) AS qname, a.year,
       SUM(a.water_billed)/1000000 AS water
 INTO
       temp_data
 FROM
       qurt_billed_industry a
 WHERE
       a.facility_id IN (SELECT <codeFld> FROM selection) AND
       a.water_source = <X> AND a.year >= <Year1> AND a.year <=
       <Year2>
```

```
GROUP BY
      a.quarter, a.year
```

Annually billed water without regard of the water source

```
SELECT
      s.<fldName>, s.<nameFld>,a.year,
      SUM(a.water_billed)/1000000 AS water
INTO
      temp_data
FROM
      selection s, qurt_billed_industry a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      s.<codeFld> = a.facility_id AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Annually billed water for one water source

```
SELECT
      s.<fldName>, s.<nameFld>,a.year,
      SUM(a.water_billed)/1000000 AS water
INTO
      temp_data
FROM
      selection s, qurt_billed_industry a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.water_source = <X> AND s.<codeFld> = a.facility_id AND
      a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Annually billed water for more than one water sources

```
SELECT
      a.water_source, b.WSName, a.year,
      SUM(a.water_billed)/1000000 AS water
INTO
      temp_data
FROM
      qurt_billed_industry a, TWaterSource b
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      b.WSCode = a.water_source AND a.year >= <Year1> AND a.year
      <= <Year2>
GROUP BY
      a.water_source, b.WSName, a.year
```

SelectParamH3

This form generates the charts for touristic water use. Depending on the constraints the user enters and on the chart type (quarterly or annually billed water) the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

Following SQL statements result for the different cases:

Quarterly billed water

First a temporary table has to be created containing the codes for the use type.

```
SELECT
      bill_cat
INTO
      temp_cat
FROM
      water_use_bill_cat
WHERE
      use_type = '4'
```

Next the final query can be executed.

```
SELECT
      a.quarter, 'Quarter '+cstr(a.quarter) AS qname, a.year,
      SUM(a.water_billed)/1000000 AS water
INTO
      temp_data
FROM
      water_use_billing a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.bill_cat IN (SELECT bill_cat FROM temp_cat) AND a.year >=
      <Year1> AND a.year <= <Year2>
GROUP BY
      a.quarter, a.year
```

Annually billed water

First a temporary table has to be created containing the codes for the use type.

```
SELECT
      bill_cat
INTO
      temp_cat
FROM
      water_use_bill_cat
WHERE
      use_type = '4'
```

Next the final query can be executed.

```
SELECT
      s.<fldNam>, s.<nameFld>, a.year,
      SUM(a.water_billed)/1000000 AS water
INTO
      temp_data
      FROM
            selection s, water_use_billing a
      WHERE
            cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
            a.bill_cat IN (SELECT bill_cat FROM temp_cat) AND
            s.<codeFld> = cstr(a.SETT_id) AND a.year >= <Year1> AND
            a.year <= <Year2>
      GROUP BY
            s.<fldName>, s.<nameFld>, a.year
```

SelectParamH4

This form generates the charts for water transferred between governorates. Depending on the year the user enters the data are retrieved from the pre-defined queries (Q_Transfer, Q_Tr_in_out and Q_Tr_Crosstable) and transferred to a temporary table, from which the resulting Excel table is generated.

SelectParamP1

This form generates the charts for municipal and touristic water demand projections and for the projections of losses in the municipal water network. Depending on the constraints the user enters and on the chart type the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

Following SQL statements result for the different cases:

Municipal water demand projections for one scenario

```
      SELECT
            s.<fldName>, s.<nameFld>, a.year, SUM(a.demand)/1000000 AS
            water
      INTO
            temp_data
      FROM
            selection s, stp_municipal a
      WHERE
            cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i> AND s.<codeFld> = cstr(a.SETT_id) AND
            a.month <= <startMonth + 11> AND a.year >= <Year1> AND
            a.year <= <Year2>
      GROUP BY
            s.<fldName>, s.<nameFld>", a.year
```

Municipal water demand projections for two scenarios

```
      SELECT
            a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
            SUM(a.demand)/1000000 AS water
      INTO
            temp_data
      FROM
            stp_municipal a
      WHERE
            cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
            a.scenario <> <i> AND a.month >= <startMonth> AND a.month
      <= <startMonth + 11> AND a.year >= <Year1> AND a.year <=
      <Year2>
GROUP BY
      a.scenario, a.year
```

Municipal water demand projections for three scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_data
FROM
      stp_municipal a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Touristic water demand projections for one scenario

```
SELECT
      s.<fldName>, s.<nameFld>, a.year, SUM(a.demand)/1000000 AS
      water
INTO
      temp_data
FROM
      selection s, stp_touristic a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND s.<codeFld> = cstr(a.SETT_id) AND
      a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>", a.year
```

Touristic water demand projections for two scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_data
FROM
      stp_touristic a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario <> <i> AND a.month <= <startMonth + 11> AND
      a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
```

```
      a.scenario, a.year
```

Touristic water demand projections for three scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_data
FROM
      stp_touristic a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Projections for municipal network losses (always for only one scenario)

First the municipal water demands are retrieved to the temporary table.

```
SELECT
      1 AS dem_type, 'Municipal demand' AS dem_name, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_data
FROM
      stp_municipal a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.month <= <startMonth + 11> AND
      a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

In the next step the touristic water demands are retrieved to another temporary table.

```
SELECT
      3 AS dem_type, 'Touristic demand' AS dem_name, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_d2
FROM
      stp_touristic a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.month <= <startMonth + 11> AND
      a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

In the next step the industrial water demands are retrieved to another temporary table.

```
SELECT
      2 AS dem_type, 'Industrial demand' AS dem_name, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_d1
FROM
      stp_industry a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.month <= <startMonth + 11> AND
      a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

In the last step the network losses are retrieved to another temporary table.

```
SELECT
      4 AS dem_type, 'Network losses' AS dem_name, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_d3
FROM
      stp_losses a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.month <= <startMonth + 11> AND
      a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Finally all temporary tables are joined together.

SelectParamP2

This form generates the charts for industrial water demand projections and for the projections of industrial wastewater generated. Depending on the constraints the user enters and on the chart type the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

From the combination of constraints there can be created 38 different SQL statements for the retrieval of industrial water demands and the same for wastewater generated by industries. All these 76 cases can be shown by just seven examples, however.

Just some remarks beforehand: The variable *demName* reads *demand* for the water demand and *waste_water* for the wastewater generated, except of this the statements for both are identical. All parts of the statements that are marked by ### at the end mean that they are only added when the user has selected one salinity class or/and one water source or/and one sewer connection indicator. In all cases where these items are not considered separately (that is, the summation is used), the responding part will be left out. The characters ### of course never are part of the SQL statement.

The following SQL statements result for the different basic cases.

Industrial water demand or industrial wastewater projections for three scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.<demName>)/1000000 AS water
```

```
INTO
      temp_data
FROM
      stp_industry a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection)
      AND a.salinity = <i> ###
      AND a.water_source = <X> ###
      AND a.facility_id IN (SELECT facility_id FROM facilities
      WHERE plant_connect IN (<X>, <LCase(X)>) ) ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Industrial water demand or industrial wastewater projections for two scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.<demName>)/1000000 AS water
INTO
      temp_data
FROM
      stp_industry a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario <> <i>
      AND a.salinity = <i> ###
      AND a.water_source = <X> ###
      AND a.facility_id IN (SELECT facility_id FROM facilities
      WHERE plant_connect IN (<X>, <LCase(X)>) ) ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Industrial water demand or industrial wastewater projections for three salinity classes

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a.<demName>)/1000000 AS water
INTO
      temp_data
FROM
      stp_industry a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND a.water_source = <X> ###
      AND a.facility_id IN (SELECT facility_id FROM facilities
      WHERE plant_connect IN (<X>, <LCase(X)>) ) ###
```

```
            AND a.month >= <startMonth> AND a.month <= <startMonth +
            11> AND a.year >= <Year1> AND a.year <= <Year2>
      GROUP BY
            a.salinity, a.year
```

Industrial water demand or industrial wastewater projections for two salinity classes

```
      SELECT
            a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
            SUM(a.<demName>)/1000000 AS water
      INTO
            temp_data
      FROM
            stp_industry a
      WHERE
            a.facility_id IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i>
            AND a.salinity <> <i>
            AND a.water_source = <X> ###
            AND a.facility_id IN (SELECT facility_id FROM facilities
            WHERE plant_connect IN (<X>, <LCase(X)>) ) ###
            AND a.month >= <startMonth> AND a.month <= <startMonth +
            11> AND a.year >= <Year1> AND a.year <= <Year2>
      GROUP BY
            a.salinity, a.year
```

Industrial water demand or industrial wastewater projections for two water sources

```
      SELECT
            a.water_source, b.WSName, a.year, SUM(a.<demName>)/1000000
            AS water
      INTO
            temp_data
      FROM
            stp_industry a, TWaterSource b
      WHERE
            a.facility_id IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i>
            AND a.salinity = <i> ###
            AND a.facility_id IN (SELECT facility_id FROM facilities
            WHERE plant_connect IN (<X>, <LCase(X)>) ) ###
            AND b.WSCode = a.water_source
            AND s.<codeFld> = a.facility_id
            AND a.month >= <startMonth> AND a.month <= <startMonth +
            11> AND a.year >= <Year1> AND a.year <= <Year2>
      GROUP BY
            a.water_source, b.WSName, a.year
```

Industrial water demand or industrial wastewater projections for two sewer connection indicators

```
      SELECT
      f.plant_connect, b.YNText, a.year, SUM(a.<demName>)/1000000
      AS water
INTO
      temp_data
FROM
      facilities f, stp_industry a, TYesNo b
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.water_source = <X> ###
      AND f.facility_id = a.facility_id AND b.YN =
      f.plant_connect
      AND s.<codeFld> = a.facility_id
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      f.plant_connect, b.YNText, a.year
```

Industrial water demand or industrial wastewater projections for a maximum of one selections of each group

```
SELECT
      s.<fldName>, s.<nameFld>, a.year, SUM(a.<demName>)/1000000
      AS water
INTO
      temp_data
FROM
      selection s, stp_industry a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.water_source = <X> ###
      AND s.<codeFld> = a.facility_id ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

SelectParamP3

This form generates the charts for irrigation water demand projections and for the projections of irrigation losses as well as their summarization. Depending on the constraints the user enters and on the chart type the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

The SQL statements for the retrieval of irrigation water demands, irrigation on-farm losses and irrigation distribution system losses look similar for the same constraints. The only difference consists of the STP-table and the fieldname for the water amount. Therefore the examples for the SQL statements will cover all three cases.

Just some remarks beforehand: The variable *demName* reads *demand* for the water demand and *loss* for the losses, the variable *stpTable* reads *stp_agriculture* for the irrigation water demands, *stp_losses_onfarm* for the on-farm losses and *stp_losses_agrodistrib* for the distribution system losses. All parts of the statements that are marked by ### at the end mean that they are only added when the user has selected one salinity class. In all cases where the salinity class are not considered separately (that is, the summation is used), the responding part will be left out. The characters ### of course never are part of the SQL statement.

Following SQL statements result for the different cases:

Irrigation water demand projections or irrigation losses for three scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.<demName>)/1000000 AS water
INTO
      temp_data
FROM
      <stpTable> a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Irrigation water demand projections or irrigation losses for two scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.<demName>)/1000000 AS water
INTO
      temp_data
FROM
      <stpTable> a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario <> <i> AND a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Irrigation water demand projections or irrigation losses for three year types

```
SELECT
      a.type_of_year, b.YTName, a.year, SUM(a.<demName>)/1000000
      AS water
INTO
      temp_data
FROM
      <stpTable> a, TYearTypes b
WHERE
```

```
            cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i>
            AND a.salinity = <i> ###
            AND b.ytcode = a.type_of_year AND a.month >= <startMonth>
            AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
            a.year <= <Year2>
      GROUP BY
            a.type_of_year, b.ytname, a.year
```

Irrigation water demand projections or irrigation losses for two year types

```
      SELECT
            a.type_of_year, b.YTName, a.year, SUM(a.<demName>)/1000000
            AS water
      INTO
            temp_data
      FROM
            <stpTable> a, TYearTypes b
      WHERE
            cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i> AND a.type_of_year <> <X>
            AND a.salinity = <i> ###
            AND b.ytcode = a.type_of_year AND a.month >= <startMonth>
            AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
            a.year <= <Year2>
      GROUP BY
            a.type_of_year, b.ytname, a.year
```

Irrigation water demand projections or irrigation losses for three salinity classes

```
      SELECT
            a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
            SUM(a.<demName>)/1000000 AS water
      INTO
            temp_data
      FROM
            <stpTable> a
      WHERE
            cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i> AND a.type_of_year = <X> AND a.month >=
            <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
            <Year1> AND a.year <= <Year2>
      GROUP BY
            a.salinity, a.year
```

Irrigation water demand projections or irrigation losses for two salinity classes

```
      SELECT
            a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
            SUM(a.<demName>)/1000000 AS water
      INTO
            temp_data
      FROM
      <stpTable> a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X> AND a.salinity <> <i>
      AND a.month >= <startMonth> AND a.month <= <startMonth + 11>
      AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.salinity, a.year
```

Irrigation water demand projections or irrigation losses for one scenario, one year type and one salinity class or the sum of salinity classes

```
SELECT
      s.<fldName>, s.<nameFld>, a.year, SUM(a.<demName>)/1000000
      AS water
INTO
      temp_data
FROM
      selection s, <stpTable> a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND s.<codeFld> = cstr(a.SETT_id) AND a.month >=
      <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
      <Year1> AND a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Combination of irrigation water demand projections and irrigation losses (for one scenario, one year type and one salinity class or the sum of salinity classes)

First the irrigation water demands are retrieved to the temporary table.

```
SELECT
      1 AS dem_type, 'Irrigation demand' AS dem_name, a.year,
      SUM(a.demand)/1000000 AS water
INTO
      temp_data
FROM
      stp_agriculture a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Next the on-farm losses are retrieved to another temporary table.

```
SELECT
```

```
      2 AS dem_type, 'On farm losses' AS dem_name, a.year,
      SUM(a.loss)/1000000 AS water
INTO
      temp_d1
FROM
      stp_losses_onfarm a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Then the distribution system losses are retrieved to another temporary table, if they shall be displayed also.

```
SELECT
      3 AS dem_type, 'Distributon system losses' AS dem_name,
      a.year, SUM(a.loss)/1000000 AS water
INTO
      temp_d2
FROM
      stp_losses_agrodistrib a
WHERE
      cstr(a.sett_id) IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Finally the content of all temporary tables needed for the chart are combined into one table.

SelectParamP4

This form generates the charts for groundwater recharge and base flow projections. Depending on the constraints the user enters and on the chart type the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

Depending on the spatial unit the user has selected the groundwater cells and the base flow segments need to be intersected with the spatial unit. To meet these needs an enhanced selection table (temp_sel_i) has to be created. The different cases are covered by the following SQL statements:

No intersection allotted

Groundwater cells

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.gw_cell_id as source_code, 1 as
      factor
INTO
```

```
      temp_sel_i
FROM
      selection s, stp_groundwater i
```

Base flow segments

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.segment_id as source_code, 1 as
      factor
INTO
      temp_sel_i
FROM
      selection s, stp_baseflow i
```

Intersection allotted

Groundwater cells: spatial unit is groundwater basin

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.gw_cell_id as source_code, 1 as
      factor
INTO
      temp_sel_i
FROM
      selection s, stp_groundwater i
WHERE
      i.gw_cell_id = s.<fldName>
```

Groundwater cells: spatial unit is not groundwater basin

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.source_code, i.factor
INTO
      temp_sel_i
FROM
      selection s, spatial_intersect_units i
WHERE
      i.source_type = 'GW' AND i.balance_unit_type = '<X>' AND
      i.balance_unit_code = s.<fldName>
```

Base flow segments

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.source_code, i.factor
INTO
      temp_sel_i
FROM
      selection s, spatial_intersect_units i
WHERE
      i.source_type = 'BS' AND i.balance_unit_type = '<X>' AND
      i.balance_unit_code = s.<fldName>
```

The SQL statements for groundwater recharge and for base flow look similar for the same restrictions, so they can be presented by one statement for each different case:

Just some remarks beforehand: The variable *Fld1* reads *gw_cell_id* for groundwater and *segment_id* for base flow, the variable *Fld2* reads *groundwater* for groundwater and *baseflow* for base flow, the variable *Tlb* reads *stp_groundwater* for groundwater and *stp_baseflow* for base flow. All parts of the statements that are marked by ### at the end mean that they are only added when the user has selected one salinity class. In all cases where the salinity class are not considered separately (that is, the summation is used), the responding part will be left out. The characters ### of course never are part of the SQL statement. For base flow an additional SQL statement has to be executed to create a second temporary table. The content of both tables is combined to on table afterwards.

Following SQL statements result for the different cases:

Groundwater recharge or base flow for three scenarios

Groundwater/base flow without intersection

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      <Tbl> a
WHERE
      a.salinity = <i> AND ###
      a.month >= <startMonth> AND a.month <= <startMonth + 11>
      AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Groundwater/base flow with intersection

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(s.factor*a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      temp_sel_i s, <Tbl> a
WHERE
      a.<Fld1> IN (SELECT source_code FROM temp_sel_i)
      AND a.salinity = <i> ###
      AND s.source_code = a.<Fld1> AND a.month >= <startMonth>
      AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Additional retrieval for base flow (no intersection needed)

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.baseflow/1000000) AS water
INTO
      temp_d1
```

```
FROM
      stp_baseflow_end a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Groundwater recharge or base flow for two scenarios

Groundwater/base flow without intersection

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      <Tbl> a
WHERE
      a.scenario <> <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Groundwater/base flow with intersection

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(s.factor*a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      temp_sel_i s, <Tbl> a
WHERE
      a.<Fld1> IN (SELECT source_code FROM temp_sel_i) AND
      a.scenario <> <i>
      AND a.salinity = <i> ###
      AND s.source_code = a.<Fld1> AND a.month >= <startMonth>
      AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Additional retrieval for base flow (no intersection needed)

```
SELECT
```

```
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.baseflow/1000000) AS water
INTO
      temp_d1
FROM
      stp_baseflow_end a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Groundwater recharge or base flow for three salinity classes

Groundwater/base flow without intersection

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a. <Fld2>/1000000) AS water
INTO
      temp_data
FROM
      <Tbl> a
WHERE
      a.scenario = <i> AND a.month >= <startMonth> AND a.month <=
      <startMonth + 11> AND a.year >= <Year1> AND a.year <=
      <Year2>
GROUP BY
      a.salinity, a.year
```

Groundwater/base flow with intersection

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(s.factor*a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      temp_sel_i s, <Tbl> a
WHERE
      a.<Fld1> IN (SELECT source_code FROM temp_sel_i) AND
      a.scenario = <i> AND s.source_code = a.<Fld1> AND a.month
      >= <startMonth> AND a.month <= <startMonth + 11> AND a.year
      >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.salinity, a.year
```

Additional retrieval for base flow (no intersection needed)

```
SELECT
```

```
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a.baseflow/1000000) AS water
INTO
      temp_d1
FROM
      stp_baseflow_end a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.month >= <startMonth> AND a.month <= <startMonth + 11>
      AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.salinity, a.year
```

Groundwater recharge or base flow for two salinity classes

Groundwater/base flow without intersection

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      <Tbl> a
WHERE
      a.scenario = <i> AND a.salinity <> <i> AND a.month >=
      <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
      <Year1> AND a.year <= <Year2>
GROUP BY
      a.salinity, a.year
```

Groundwater/base flow with intersection

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(s.factor*a.<Fld2>/1000000) AS water
INTO
      temp_data
FROM
      temp_sel_i s, <Tbl> a
WHERE
      a.<Fld1> IN (SELECT source_code FROM temp_sel_i) AND
      a.scenario = <i> AND a.salinity <> <i> AND s.source_code =
      a.<Fld1> AND a.month >= <startMonth> AND a.month <=
      <startMonth + 11> AND a.year >= <Year1> AND a.year <=
      <Year2>
GROUP BY
      a.salinity, a.year
```

Additional retrieval for base flow (no intersection needed)

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a.baseflow/1000000) AS water
```

```
INTO
      temp_d1
FROM
      stp_baseflow_end a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.salinity <> <i> AND a.month >= <startMonth> AND a.month
      <= <startMonth + 11> AND a.year >= <Year1> AND a.year <=
      <Year2>
GROUP BY
      a.salinity, a.year
```

Groundwater recharge or base flow for one scenario and one salinity class or the sum of all salinity classes

Groundwater/base flow without intersection

```
SELECT
      s.<fldName>, s.<nameFld>, a.year, SUM(a.<Fld2>/1000000) AS
      water
INTO
      temp_data
FROM
      <selTbl> s, <Tbl> a
WHERE
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Groundwater/base flow with intersection

```
SELECT
      s.<fldName>, s.<nameFld>, a.year,
      SUM(s.factor*a.<Fld2>/1000000) AS water
INTO
       temp_data
FROM
      <selTbl> s, <Tbl> a
WHERE
      a.<Fld1> IN (SELECT source_code FROM temp_sel_i) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND s.source_code = a.<Fld1> AND a.month >= <startMonth>
      AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Additional retrieval for base flow (no intersection needed)

```
SELECT
```

```
      s.<fldName>, s.<nameFld>, a.year, SUM(a.baseflow/1000000)
      AS water
INTO
      temp_d1
FROM
      selection s, stp_baseflow_end a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection)
      AND a.salinity = <i> ###
      AND s.<codeFld> = a.wadi_endpoint_id AND a.month >=
      <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
      <Year1> AND a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

SelectParamP5

This form generates the charts for treated municipal (domestic and touristic), industrial or total wastewater effluent projections. Depending on the constraints the user enters and on the chart type the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

For the presentation of the total effluent the data are retrieved only from the table *stp_wwtp*. The industrial effluent had been presented by form *SelectParamP2*, but now not a presentation related to the settlements but to the wastewater treatment plants is needed. The data have to be extracted from the table *stp_industries* using a modified selection table. For the presentation of only the municipal effluent the industrial effluent has to be subtracted from the total effluent. In this case the summarized value of industrial effluent will be retrieved with a minus-sign, and for the temporary tables receiving the data will be used diverse names.

Just some remarks beforehand: The variable *tabName1* reads *temp_data* when only the total effluent shall be presented, and *temp_d1* when the extracted municipal effluent shall be shown. The variable *tabName2* reads *temp_data* when only the industrial effluent shall be presented, and *temp_d2* when the extracted municipal effluent shall be shown. All parts of the SQL statements that are marked by ### at the end mean that they are only added when the user has selected one wwtp owner type. In all cases where the wwtp owner types are not considered separately (that is, the summation is used), the responding part will be left out. The characters ### of course never are part of the SQL statement.

Following SQL statements result for the different cases:

Selection table for industrial effluent

```
SELECT
      w.facility_id AS wwtp, s.<fldName>, s.<nameFld>,
      i.facility_id AS industry, w.owner_id
INTO
      temp_sel
FROM
      selection s, facilities i, facilities w
WHERE
      w.facility_id = s.<codeFld> AND i.facility_id IN (SELECT
      c.facility_id FROM facilities c WHERE c.service_zone_id =
      w.service_zone_id AND c.sic_code <> '9000')
```

Effluent for three scenarios

Total effluent

```
SELECT
```

```
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.wastewater)/1000000 AS water
INTO
      <tabName1>
FROM
      stp_wwtp a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection)
      AND f.owner_id = <X> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth + 11>
      AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Industrial effluent

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      <->SUM(a.waste_water)/1000000 AS water
INTO
      <tabName2>
FROM
      stp_industry a
WHERE
      a.facility_id IN (SELECT industry FROM temp_sel)
      AND s.owner_id = <X> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Building of municipal effluent (the content of temp_d2 has been appended to temp_d1 beforehand)

```
SELECT
      a.scenario, a.sname, a.year, SUM(a.water) AS water
INTO
      temp_data
FROM
      temp_d1 a
GROUP BY
      a.scenario, a.sname, a.year
```

Effluent for two scenarios

Total effluent

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.wastewater)/1000000 AS water
INTO
      <tabName1>
FROM
```

```
      stp_wwtp a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario <> <i>
      AND f.owner_id = <X> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Industrial effluent

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      <->SUM(a.waste_water)/1000000 AS water
INTO
      <tabName2>
FROM
      stp_industry a
WHERE
      a.facility_id IN (SELECT industry FROM temp_sel) AND
      a.scenario <> <i>
      AND s.owner_id = <X> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.scenario, a.year
```

Building of municipal effluent (the content of temp_d2 has been appended to temp_d1 beforehand)

```
SELECT
      a.scenario, a.sname, a.year, SUM(a.water) AS water
INTO
      temp_data
FROM
      temp_d1 a
GROUP BY
      a.scenario, a.sname, a.year
```

Effluent for two wwtp owner types

Total effluent

```
SELECT
      f.owner_id, b.OText, a.year, SUM(a.wastewater)/1000000 AS
      water
INTO
      <tabName1>
FROM
      facilities f, stp_wwtp a, TWWTPOwner b
WHERE
```

```
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND f.facility_id = a.facility_id AND
      b.OWCode = f.owner_id AND a.month >= <startMonth> AND
      a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      f.owner_id, b.OText, a.year
```

Industrial effluent

```
SELECT
      s.owner_id, b.OText, a.year, <->SUM(a.waste_water)/1000000
      AS water
INTO
      <tabName2>
FROM
      temp_sel s, stp_industry a, TWWTPOwner b
WHERE
      a.facility_id IN (SELECT industry FROM temp_sel) AND
      a.scenario = <i> AND s.industry = a.facility_id AND
      b.OWCode = s.owner_id AND a.month >= <startMonth> AND
      a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      s.owner_id, b.OText, a.year
```

Building of municipal effluent (the content of temp_d2 has been appended to temp_d1 beforehand)

```
SELECT
      a.owner_id, a.OText, a.year, SUM(a.water) AS water
INTO
      temp_data
FROM
      temp_d1 a
GROUP BY
      a.owner_id, a.OText, a.year
```

Effluent for one scenario and one owner type or the sum of all owner types

Total effluent

```
SELECT
      s.<fldName>, s.<nameFld>, a.year, SUM(a.wastewater)/1000000
      AS water
INTO
      <tabName1>
FROM
      selection s, stp_wwtp a
WHERE
      a.facility_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND f.owner_id = <X> ###
      AND s.<codeFld> = a.facility_id AND a.month >= <startMonth>
      AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Industrial effluent

```
SELECT
      s.<fldName>, s.<nameFld>, a.year,
      <->SUM(a.waste_water)/1000000 AS water
INTO
      <tabName2>
FROM
      temp_sel s, stp_industry a
WHERE
      a.facility_id IN (SELECT industry FROM temp_sel) AND
      a.scenario = <i>
      AND s.owner_id = <X> ###
      AND s.industry = a.facility_id AND a.month >= <startMonth>
      AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      s.<fldName>, s.<nameFld>, a.year
```

Building of municipal effluent (the content of temp_d2 has been appended to temp_d1 beforehand)

```
SELECT
      a.<fldName>, a.<nameFld>, a.year, SUM(a.water) AS water
INTO
      temp_data
FROM
      temp_d1 a
GROUP BY
      a.<fldName>, a.<nameFld>, a.year
```

SelectParamP6

This form generates the charts for water import and export projections. Depending on the constraints the user enters the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

For the time being only one chart type is generated by this form. It contains three data series: imported water, exported water and transfer losses. To be able to differentiate between import and export it is necessary first to create two modified selection tables. When all data are retrieved they are combined to the temporary table which is the basis of the Excel chart..

All parts of the SQL statements that are marked by ### at the end mean that they are only added when the user has selected one salinity class. In all cases where the salinity class are not considered separately (that is, the summation is used), the responding part will be left out. The characters ### of course never are part of the SQL statement.

Selection tables

Inflow

```
SELECT
      *
INTO
      temp_sel_i
FROM
      Selection
WHERE
      <codeFld> IN (SELECT transfer_point_id FROM transfer_points
      WHERE flow_type = 'I')
```

Outflow

```
SELECT
      *
INTO
      temp_sel_o
FROM
      Selection
WHERE
      <codeFld> IN (SELECT transfer_point_id FROM transfer_points
      WHERE flow_type = 'O')
```

Data retrieval

Imported water

```
SELECT
      3 AS res_type, 'Imported water' AS res_name, a.year,
      SUM(a.transfer)/1000000 AS water
INTO
      temp_data
FROM
      stp_transfer a
WHERE
      a.transfer_point_id IN (SELECT <codeFld> FROM temp_sel_i)
      AND a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Exported water

```
SELECT
      2 AS res_type, 'Exported water' AS res_name, a.year, -
      SUM(a.transfer)/1000000 AS water
INTO
      temp_d1
FROM
      stp_transfer a
WHERE
```

```
      a.transfer_point_id IN (SELECT <codeFld> FROM temp_sel_o)
      AND a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Transfer losses

```
SELECT
      1 AS res_type, 'Transfer losses' AS res_name, a.year, -
      SUM(a.transfer_losses)/1000000 AS water
INTO
      temp_d2
FROM
      stp_transfer_losses a
WHERE
      a.transfer_point_id IN (SELECT <codeFld> FROM selection)
      AND a.scenario = <i>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

SelectParamP7

This form generates the charts for the projections of flood flow, surface water (flood flow and base flow) or groundwater recharge (which includes already the base flow) plus flood flow. Depending on the constraints the user enters and on the chart type the queries needed for data retrieval are compiled and the retrieved data are transferred to a temporary table, from which by a standardized procedure the resulting Excel chart is generated.

All parts of the SQL statements that are marked by ### at the end mean that they are only added when the user has selected one salinity class. In all cases where the salinity class are not considered separately (that is, the summation is used), the responding part will be left out. The characters ### of course never are part of the SQL statement.

Following SQL statements cover all cases for the presentation of flood flow projections:

Flood flow for three scenarios

```
SELECT
      a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
      SUM(a.floodflow)/1000000 AS water
INTO
      temp_data
FROM
      stp_flood_flow_out a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
      GROUP BY
            a.scenario, a.year
```

Flood flow for two scenarios

```
      SELECT
            a.scenario, 'Scenario '+cstr(a.scenario) AS sname, a.year,
            SUM(a.floodflow)/1000000 AS water
      INTO
            temp_data
      FROM
            stp_flood_flow_out a
      WHERE
            a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
            a.scenario <> <i> AND a.type_of_year = <X>
            AND a.salinity = <i> ###
            AND a.month >= <startMonth> AND a.month <= <startMonth +
            11> AND a.year >= <Year1> AND a.year <= <Year2>
      GROUP BY
            a.scenario, a.year
```

Flood flow for three year types

```
      SELECT
            a.type_of_year, b.YTName, a.year, SUM(a.floodflow)/1000000
            AS water
      INTO
            temp_data
      FROM
            stp_flood_flow_out a, TYearTypes b
      WHERE
            a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i>
            AND a.salinity = <i> ###
            AND b.ytcode = a.type_of_year AND a.month >= <startMonth>
            AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
            a.year <= <Year2>
      GROUP BY
            a.type_of_year, b.ytname, a.year
```

Flood flow for two year types

```
      SELECT
            a.type_of_year, b.YTName, a.year, SUM(a.floodflow)/1000000
            AS water
      INTO
            temp_data
      FROM
            stp_flood_flow_out a, TYearTypes b
      WHERE
            a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
            a.scenario = <i> AND a.type_of_year <> <X>
      AND a.salinity = <i> ###
      AND b.ytcode = a.type_of_year AND a.month >= <startMonth>
      AND a.month <= <startMonth + 11> AND a.year >= <Year1> AND
      a.year <= <Year2>
GROUP BY
      a.type_of_year, b.ytname, a.year
```

Flood flow for three salinity classes

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a.floodflow)/1000000 AS water
INTO
      temp_data
FROM
      stp_flood_flow_out a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X> AND a.month >=
      <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
      <Year1> AND a.year <= <Year2>
GROUP BY
      a.salinity, a.year
```

Flood flow for two salinity classes

```
SELECT
      a.salinity, 'Class '+cstr(a.salinity) AS cname, a.year,
      SUM(a.floodflow)/1000000 AS water
INTO
      temp_data
FROM
      stp_flood_flow_out a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X> AND a.salinity <>
      <i> AND a.month >= <startMonth> AND a.month <= <startMonth
      + 11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.salinity, a.year
```

Flood flow for one scenario, one year type and one salinity class or the sum of all classes

```
SELECT
      s.<fldName>, s.<nameFld>, a.year, SUM(a.floodflow)/1000000
      AS water
INTO
      temp_data
FROM
      selection s, stp_flood_flow_out a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X>
            AND a.salinity = <i> ###
            AND s.<codeFld> = a.wadi_endpoint_id AND a.month >=
            <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
            <Year1> AND a.year <= <Year2>
      GROUP BY
            s.<fldName>, s.<nameFld>, a.year
```

For the combination of the flood flow with the groundwater recharge or with the base flow some more actions have to take place. Depending on the spatial unit the user has selected the groundwater cells and the base flow segments need to be intersected with the spatial unit. To meet these needs an enhanced selection table has to be created. The different cases are covered by the following SQL statements:

Groundwater

No intersection allotted

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.gw_cell_id as source_code, 1 as
      factor
INTO
      temp_sel_gw
FROM
      selection s, stp_groundwater i
```

Intersection allotted, spatial unit = groundwater basins

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.gw_cell_id as source_code, 1 as
      factor
INTO
      temp_sel_gw
FROM
      selection s, stp_groundwater i
WHERE
      i.gw_cell_id = s.<fldName>
```

Intersection allotted, other spatial units

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.source_code, i.factor
INTO
      temp_sel_gw
FROM
      selection s, spatial_intersect_units i
WHERE
      i.source_type = 'GW' AND i.balance_unit_type = '<X>' AND
      i.balance_unit_code = s.<fldName>
```

Base flow

No intersection allotted

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.segment_id as source_code, 1 as
      factor
```

```
INTO
      temp_sel_bs
FROM
      selection s, stp_baseflow i
```

Intersection allotted

```
SELECT DISTINCT
      s.<fldName>, s.<nameFld>, i.source_code, i.factor
INTO
      temp_sel_bs
FROM
      selection s, spatial_intersect_units i
WHERE
      i.source_type = 'BS' AND i.balance_unit_type = '<X>' AND
      i.balance_unit_code = s.<fldName>
```

After the proper enhanced selection table has been created, the data can be retrieved.

Flood flow

```
SELECT
      3 AS res_type, 'Flood flow' AS res_name, a.year,
      SUM(a.floodflow)/1000000 AS water
INTO
      temp_data
FROM
      stp_flood_flow_out a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i> AND a.type_of_year = <X>
      AND a.salinity = <i> ###
      AND a.month >= <startMonth> AND a.month <= <startMonth +
      11> AND a.year >= <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

Base flow

First base flow table

```
SELECT
      2 AS res_type, 'Base flow' AS res_name, a.year,
      SUM(s.factor*a.baseflow)/1000000 AS water
INTO
      temp_d1
FROM
      temp_sel_bs s, stp_baseflow a
WHERE
      a.segment_id IN (SELECT source_code FROM temp_sel_bs) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
            AND s.source_code = a.segment_id AND a.month >=
            <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
            <Year1> AND a.year <= <Year2>
      GROUP BY
            a.year
```

Second base flow table

```
SELECT
      2 AS res_type, 'Base flow' AS res_name, a.year,
      SUM(a.baseflow)/1000000 AS water
INTO
      temp_d2
FROM
      selection s, stp_baseflow_end a
WHERE
      a.wadi_endpoint_id IN (SELECT <codeFld> FROM selection) AND
      a.scenario = <i>
      AND a.salinity = <i> ###
      AND s.<codeFld> = a.wadi_endpoint_id
GROUP BY
      a.year
```

Groundwater

```
SELECT
      1 AS res_type, 'Groundwater' AS res_name, a.year,
      SUM(s.factor*a.groundwater)/1000000 AS water
INTO
      temp_d1
FROM
      temp_sel_gw s, stp_groundwater a
      WHERE a.gw_cell_id IN (SELECT source_code FROM temp_sel_gw)
      AND a.scenario = <i>
      AND a.salinity = <i> ###
      AND s.source_code = a.gw_cell_id AND a.month >=
      <startMonth> AND a.month <= <startMonth + 11> AND a.year >=
      <Year1> AND a.year <= <Year2>
GROUP BY
      a.year
```

3.3 Tables

Following the internal tables are listed with a short description of their contents. The fields building an unique index are marked with an asterisk.

TDataType

This table lists the data types to be visualized.

Key	Fieldname	Description of content
*	DTCode	Unique code of the data type. The first character has to be D for demands or R for resources.
	DTName	Description of the data type as to be shown with the selection button

TShapeFiles

This table lists the shape files to be used for spatial selection.

Key	Fieldname	Description of content
*	SFCode	Unique code for the shape file. Codes 01 – 20 indicate polygon shapes, 21 – 40 point shapes, 41 – 60 line shapes.
	AVSFName	Shape file name (without extension .shp)
	AVSubDir	Subdirectory where the shape file is located. The part of the path of the shape files which is common to all, is given by the configuration file DVS:CFG.
	SFName	Description of the shape file as to be shown with the selection button. It also is used for the generation of the Excel chart title.
	CodeField	Name of the field containing the code of the shape in the shape file's database table
	NameField	Name of the field containing the name of the shape in the shape file's database table. Code and name may be identical if only one of them exists.
	SelOutputField	Name of the field which will contain the code of the shape file in the output selection table. The contents has to be different to NameField as in most cases both fieldnames will be used for the generation of the selection table.
	IntersectCode	Intersection code
	LabelThreshold	Threshold when labels (names) should be displayed on the selection map
	LabelSize	Size of the labels
	Color	Color code for the polygons / points / lines

TTableNames

This table lists the names of the external (Oracle or Access) tables used for data retrieval. The names do not contain the table owner which is given by the configuration file DVS.CFG. When importing an Oracle table to Access this owner normally is made part of the new table name. This default behavior should not be changed up when generating a local offline database as described in chapter 2.2.

Key	Fieldname	Description of content
*	TNCode	Unique code of the external table
	TNName	Name of the external table

TExcelFiles

This table lists the Excel files.

Key	Fieldname	Description of content
*	XLCode	Unique code of the excel file

	XLSubDir	Subdirectory where the Excel file is located. The part of the path of the Excel files which is common to all, is given by the configuration file DVS:CFG.
	XLName	Name of the excel file (without the extent .xls)
	XLDescription	Description of the Excel file. The text serves just for information and is not yet used for any processing.

TWaterSource

This is a lookup table with the names of the water source types for industries (municipal network or wells).

Key	Fieldname	Description of content
*	WSCode	One-letter code for the water source type as used in the external tables (W or N)
	WSName	Name of the water source type as to be used in the resulting Excel chart.

TWWTPOwner

This is a lookup table with the names of owner types of WWT plants

Key	Fieldname	Description of content
*	OWCode	Code of the owner type as used in the external tables (PRV or WAJ)
	OText	Name of the owner type as to be used in the resulting Excel chart.

TYearTypes

This is a lookup table with the names of year types.

Key	Fieldname	Description of content
*	YTCode	One-letter code of the year type as used in the external tables (D, M, W)
	YTName	Name of the year type as to be used in the resulting Excel chart.

TYesNo

This is a lookup table for Yes and No written out in full

Key	Fieldname	Description of content
*	YNCode	Unique one-digit internal code for Yes or No
	YN	One-letter code of Yes and No as used in the external tables (Y, y, N, n)
	YNText	Yes or No written out in full

TGovCodes

This is a lookup table for governorate codes and names as used in the JICA Excel sheets.

Key	Fieldname	Description of content
*	Sequ	Sequence in which the governorates should be listed in the resulting sheets.
	Code	Two-letter code of the governorate
	Name	Name of the governorate

T_DT_SF

This table lists the conjunction between data types and shape files.

Key	Fieldname	Description of content
*	DTCode	Code of the data type
*	SFCode	Code of the shape file
	SFjica	Marking if the shape file can be used for selections based on the Excel files. J means it also may be used with the JICA Excel files.
	allowSelect	Indicates if the shape is selectable or only used for display.

T_DT_CT

This table lists the conjunction between data types and chart types and describes the individual chart types.

Key	Fieldname	Description of content
*	DTCode	Code of the data type
*	CTCode	Code of the chart type
	CTjica	Marking if the shape file can be used for selections based on the Excel files. J means it also may be used with the JICA Excel files.
	CTParam	The characters used to discern between the forms of the "SelectParam"-group. This identifies the form which has to be used to enter the constraints and to produce the resulting Excel file.
	CTTitle	Basic chart title which is used to create the titles in the resulting Excel file..
	CTMapMultiSelect	Indicates if multiple elements may be selected on the map. For Excel charts that have to portray more than one data series (for example billed water on a quarterly basis) only one map item may be selected.
	CTSkipMap	Indicates if the spatial selection should be skipped, because it would make no sense. Some of the Excel files are not related to several geographic locations. For all cases where the data is retrieved from the database the spatial selection is mandatory.

T_DT_CT_TN

This table lists the conjunction between data types, chart types and external database tables.

Key	Fieldname	Description of content
*	DTCode	Code of the data type
*	CTCode	Code of the chart type
*	TNCode	Code of the external table

T_DT_CT_XL

This table lists the conjunction between data types, chart types and Excel files.

Key	Fieldname	Description of content
*	DTCode	Code of the data type
*	CTCode	Code of the chart type
*	XLCode	Code of the Excel file

TPrYear

This table lists the projection years as defined by the preprocessing modules. It is not updated automatically but has been predefined.

Key	Fieldname	Description of content
*	PYear	Projection year, from 2005 to 2040 in 5-year steps.

TYears

This table lists the years when historical data are available. It is created automatically from tables containing historical data.

Key	Fieldname	Description of content
	year	Historical year with available data.

TJYHist

This table lists the years of historic population data as entered in the JICA Excel sheet. It is not updated automatically but has been predefined.

Key	Fieldname	Description of content
	year	Historical year with population data.

TJYProj

This table lists the projection years of the JICA Excel sheets, including the reference year. The table is not updated automatically but has been predefined.

Key	Fieldname	Description of content
	year	Projection year.

SELECTION

This table lists the geographical items which have been selected for further processing. The table structure varies depending on the information about fieldnames in table T_DT_CT and on the type of selected items (polygons or points).

SHP_TABLE

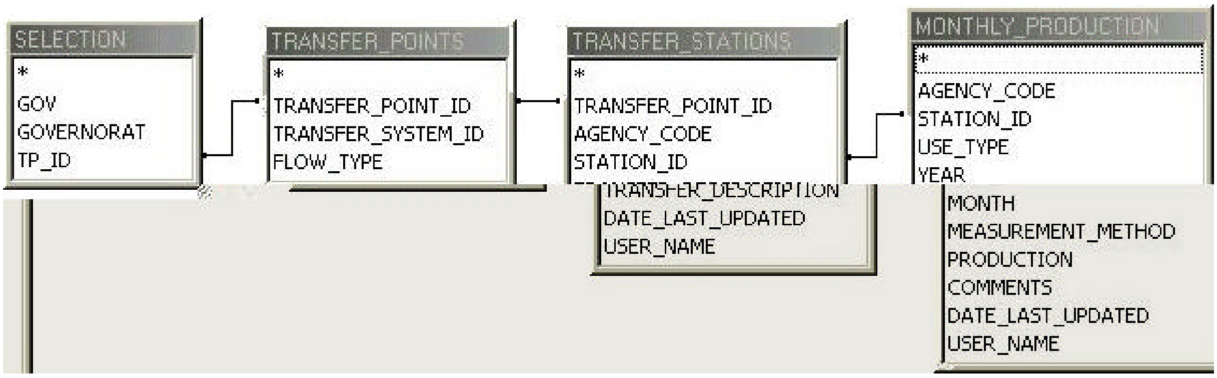
This is an internal table used for the process of selecting items from the shape files.

3.4 Queries

The application works with only a few predefined queries. As those queries base on temporary and on external tables they only can be used during program execution.

Q_Transfer

This query provides a view of yearly amounts of water in the transfer system, related to the governorates. The tables concerned are shown in the figure below. The table SELECTION is generated by the spatial selection, the other tables are links to the external database.

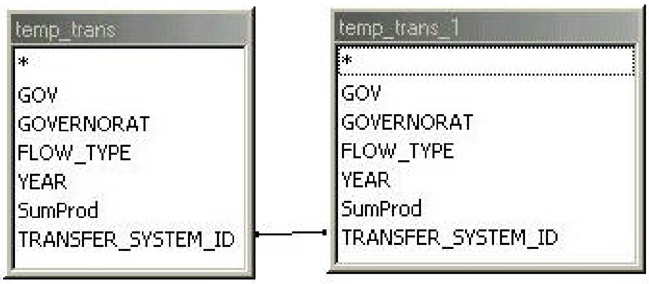


The connection of the tables is expressed by the following SQL statement.

```
SELECT
       SELECTION.GOV, SELECTION.GOVERNORAT,
       TRANSFER_POINTS.FLOW_TYPE, MONTHLY_PRODUCTION.YEAR,
       Sum(MONTHLY_PRODUCTION.PRODUCTION) AS SumProd,
       TRANSFER_POINTS.TRANSFER_SYSTEM_ID
FROM
       (SELECTION INNER JOIN TRANSFER_POINTS ON SELECTION.TP_ID =
       TRANSFER_POINTS.TRANSFER_POINT_ID) INNER JOIN
       (TRANSFER_STATIONS INNER JOIN MONTHLY_PRODUCTION ON
       TRANSFER_STATIONS.STATION_ID =
       MONTHLY_PRODUCTION.STATION_ID) ON
       TRANSFER_POINTS.TRANSFER_POINT_ID =
       TRANSFER_STATIONS.TRANSFER_POINT_ID
GROUP BY
       SELECTION.GOV, SELECTION.GOVERNORAT,
       TRANSFER_POINTS.FLOW_TYPE, MONTHLY_PRODUCTION.YEAR,
       TRANSFER_POINTS.TRANSFER_SYSTEM_ID
HAVING
       (((Sum(MONTHLY_PRODUCTION.PRODUCTION))>0))
ORDER BY
       MONTHLY_PRODUCTION.YEAR, TRANSFER_POINTS.TRANSFER_SYSTEM_ID
```

Q_Tr_in_out

This query interrelates incoming and outgoing water transfers between governorates. For this it uses a temporary table which contains the extract of the data of query Q_Transfer for a selected year.



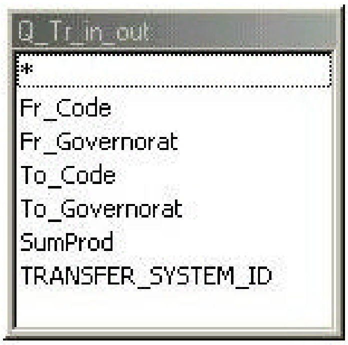
The query is represented by following SQL statement:

```
SELECT
       temp_trans.GOV AS Fr_Code, temp_trans.GOVERNORAT AS
       Fr_Governorat, temp_trans_1.GOV AS To_Code,
       temp_trans_1.GOVERNORAT AS To_Governorat,
       temp_trans.SumProd, temp_trans_1.TRANSFER_SYSTEM_ID
FROM
```

```
              temp_trans INNER JOIN temp_trans AS temp_trans_1 ON
              temp_trans.TRANSFER_SYSTEM_ID =
              temp_trans_1.TRANSFER_SYSTEM_ID
       WHERE
              (((temp_trans.GOV)<>[temp_trans_1].[gov]) AND
              ((temp_trans.FLOW_TYPE)="O") AND
              ((temp_trans_1.FLOW_TYPE)="I"))
       ORDER BY
              temp_trans.GOV, temp_trans_1.GOV
```

Q_Tr_Crosstable

Based on query Q_Tr_in_out a cross-table is defined.



Q_Tr_in_out
*
Fr_Code
Fr_Governorat
To_Code
To_Governorat
SumProd
TRANSFER_SYSTEM_ID

The query is represented by following SQL statement:

```
       TRANSFORM
              Sum(Q_Tr_in_out.SumProd) AS [Summe von SumProd]
       SELECT
              Q_Tr_in_out.Fr_Governorat
       FROM
              Q_Tr_in_out
       GROUP BY
              Q_Tr_in_out.Fr_Governorat
       PIVOT
              Q_Tr_in_out.To_Governorat
```

3.5 Macros

autoexec

This macro refers to a function of the same name in module dvs_gen. It initiates actions to be performed at startup (minimize the database window, activate form NWMP).

3.6 Modules

dvs_gen

This module contains a collection of globally used variables and of subroutines and functions of a more general type.