SQL scripts processing for IBM i

User guide

Contents

Introduction

Creation of this application was motivated by the fact that the popular utility Query/400 (later called Query for i) is unable to display and print all characters coded in character sets UCS-2 (CCSID 13488), UTF-16 (CCSID 1200), or UTF-8 (CCSID 1208).

This application enables creating, saving and running scripts of SQL statements for the IBM DB2 for i. *Script* is defined here as a text of SQL statements written in a text file with the suffix *.sql*. One or more SQL statements delimited by a semicolon can be written in the script. The SQL statements may be of any kind (DDL or DML) and can contain parameters designated by question marks. Most of the time the script will be a query, i. e. a single SELECT statement. That means that the creator of the script must know the SQL language at least at the level of the SELECT statement.

Scripts are usually amended by specially structured comment lines which enable flexible formatting of the script result when displayed on the screen or printed on paper.

Programs are written in Java language and require version Java SE 8 or higher. They cooperate with IBM i Toolbox for Java (or JTOpen framework). The programs were created and tested Mac OS X, macOS, Windows 7, Windows 10 with remote Internet connection to system IBM i.

The application is not to be installed, it is ready to use (see Using the application below).

Using the application

The application is not installed. It is delivered as a directory containing further directories and files.

```
documents
jt400.jar
jt400Small.jar
logfiles
paramfiles
printfiles
Q_Menu.jar
Q_MenuUser.jar
scriptfiles
workfiles
```
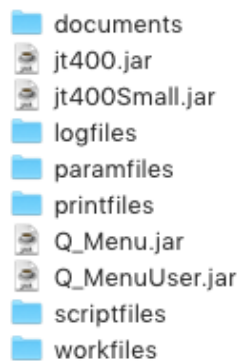
Directories

- *documents* - contains this document in Czech and English languages,
- *logfiles* - contains text files *err.txt* and *out.txt* to which messages are redirected from files System.err and System.out (console).
- *paramfiles* - contains file *Parameters.txt* with the application parameters,
- *printfiles* - contains text files, which are result of script runs,
- *scriptfiles* - contains skript files whose names end with .sql,
- *workfiles* - contains the text file *Print.txt*, result of the most recent script run.

Note: Files *err.txt* and *out.txt* serve to find the cause of an error in program.

Program files

- File *jt400Small.jar* - contains subset of Java classes from the framework IBM i Toolbox for Java.
- File *Q_Menu.jar* - contains Java application classes of programs and starts the application for the application administrator and the creator of scripts.
- File *Q_MenuUser.jar* - contains Java application classes of programs and starts the application for the application user.

Location and running the application

The application directory can be placed anywhere and possibly renamed. Shortcuts (aliases) can be created from files *Q_Menu.jar* (programs for the administrator) *Q_MenuUser.jar* (programs for the user) and placed somewhere.

The application starts by double click on the shortcuts (aliases) or on the original *.jar* files; the application menu will show.

The same copy of the application works in systems macOS and Windows.

Running scripts

In this chapter, running scripts is illustrated in examples.

Press button *Run* in the application main menu and the window with the title "Running SQL scripts" is displayed containing list of scripts. The list is empty initially. In order to get entries in the list, scripts must be created. Usually, the application administrator creates them using the Edit function.

Query without variable parameters
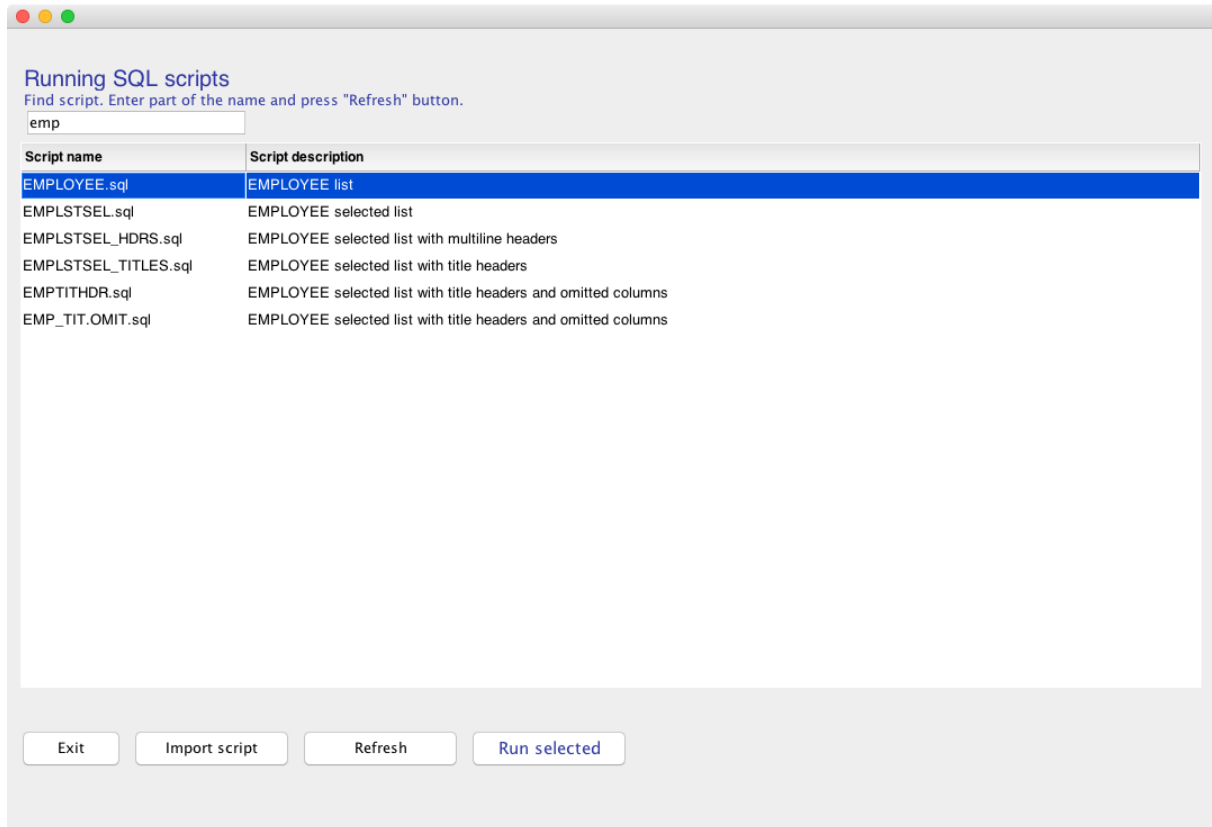
Select the row with the script name "EMPLOYEE.sql" and run the following query using button *Run selected*. The script is as follows.

```
--EMPLOYEE list
SELECT * FROM EMPLOYEE
```

The result of the query is displayed in the window.

Result of the SQL script EMPLOYEE

EMPLOYEE list

Sunday, December 20, 2015 7:12:18 PM

EMPNO	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
000010	CHRISTINE	I	HAAS	A00	3978	1965-01-01	PRES	18	F	1933-08-24	52750.00	1000.00	4220.00
000020	MICHAEL	L	THOMPSON	B01	3476	1973-10-10	MANAGER	18	M	1948-02-02	41250.00	800.00	3300.00
000030	SALLY	A	KWAN	C01	4738	1975-04-05	MANAGER	20	F	1941-05-11	38250.00	800.00	3060.00
000050	JOHN	B	GEYER	E01	6789	1949-08-17	MANAGER	16	M	1925-09-15	40175.00	800.00	3214.00
000060	IRVING	F	STERN	D11	6423	1973-09-14	MANAGER	16	M	1945-07-07	32250.00	500.00	2580.00
000070	EVA	D	PULASKI	D21	7831	1980-09-30	MANAGER	16	F	1953-05-26	36170.00	700.00	2893.00
000090	EILEEN	W	HENDERSON	E11	5498	1970-08-15	MANAGER	16	F	1941-05-15	29750.00	600.00	2380.00
000100	THEODORE	Q	SPENSER	E21	0972	1980-06-19	MANAGER	14	M	1956-12-18	26150.00	500.00	2092.00
000110	VINCENZO	G	LUCCHESSI	A00	3490	1958-05-16	SALESREP	19	M	1929-11-05	46500.00	900.00	3720.00
000120	SEAN		O'CONNELL	A00	2167	1963-12-05	CLERK	14	M	1942-10-18	29250.00	600.00	2340.00
000130	DELORES	M	QUINTANA	C01	4578	1971-07-28	ANALYST	16	F	1925-09-15	23800.00	500.00	1904.00
000140	HEATHER	A	NICHOLLS	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420.00	600.00	2274.00
000150	BRUCE		ADAMSON	D11	4510	1972-02-12	DESIGNER	16	M	1947-05-17	25280.00	500.00	2022.00
000160	ELIZABETH	R	PIANKA	D11	3782	1977-10-11	DESIGNER	17	F	1955-04-12	22250.00	400.00	1780.00
000170	MASATOSHI	J	YOSHIMURA	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680.00	500.00	1974.00
000180	MARILYN	S	SCOUTTEN	D11	1682	1973-07-07	DESIGNER	17	F	1949-02-21	21340.00	500.00	1707.00
000190	JAMES	H	WALKER	D11	2986	1974-07-26	DESIGNER	16	M	1952-06-25	20450.00	400.00	1636.00
000200	DAVID		BROWN	D11	4501	1966-03-03	DESIGNER	16	M	1941-05-29	27740.00	600.00	2217.00
000210	WILLIAM	T	JONES	D11	0942	1979-04-11	DESIGNER	17	M	1953-02-23	18270.00	400.00	1462.00
000220	JENNIFER	K	LUTZ	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840.00	600.00	2387.00
000230	JAMES	J	JEFFERSON	D21	2094	1966-11-21	CLERK	14	M	1935-05-30	22180.00	400.00	1774.00
000240	SALVATORE	M	MARINO	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760.00	600.00	2301.00
000250	DANIEL	S	SMITH	D21	0961	1969-10-30	CLERK	15	M	1939-11-12	19180.00	400.00	1534.00
000260	SYBIL	P	JOHNSON	D21	8953	1975-09-11	CLERK	16	F	1936-10-05	17250.00	300.00	1380.00
000270	MARIA	L	PEREZ	D21	9001	1980-09-30	CLERK	15	F	1953-05-26	27380.00	500.00	2190.00
000280	ETHEL	R	SCHNEIDER	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250.00	500.00	2100.00
000290	JOHN	R	PARKER	E11	4502	1980-05-30	OPERATOR	12	M	1946-07-09	15340.00	300.00	1227.00
000300	PHILIP	X	SMITH	E11	2095	1972-06-19	OPERATOR	14	M	1936-10-27	17750.00	400.00	1420.00
000310	MAUDE	F	SETRIGHT	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900.00	300.00	1272.00
000320	RAMLAL	V	MEHTA	E21	9990	1965-07-07	FIELDREP	16	M	1932-08-11	19950.00	400.00	1596.00
000330	WING		LEE	E21	2103	1976-02-23	FIELDREP	14	M	1941-07-18	25370.00	500.00	2030.00
000340	JASON	R	GOUNOT	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840.00	500.00	1907.00
200010	DIAN	J	HEMMINGER	A00	3978	1965-01-01	SALESREP	18	F	1933-08-14	46500.00	1000.00	4220.00
200120	GREG		ORLANDO	A00	2167	1972-05-05	CLERK	14	M	1942-10-18	29250.00	600.00	2340.00
200140	KIM	N	NATZ	C01	1793	1976-12-15	ANALYST	18	F	1946-01-19	28420.00	600.00	2274.00
200170	KIYOSHI		YAMAMOTO	D11	2890	1978-09-15	DESIGNER	16	M	1951-01-05	24680.00	500.00	1974.00
200220	REBA	K	JOHN	D11	0672	1968-08-29	DESIGNER	18	F	1948-03-19	29840.00	600.00	2387.00
200240	ROBERT	M	MONTEVERDE	D21	3780	1979-12-05	CLERK	17	M	1954-03-31	28760.00	600.00	2301.00
200280	EILEEN	R	SCHWARTZ	E11	8997	1967-03-24	OPERATOR	17	F	1936-03-28	26250.00	500.00	2100.00
200310	MICHELLE	F	SPRINGER	E11	3332	1964-09-12	OPERATOR	12	F	1931-04-21	15900.00	300.00	1272.00
200330	HELENA		WONG	E21	2103	1976-02-23	FIELDREP	14	F	1941-07-18	25370.00	500.00	2030.00
200340	ROY	R	ALONZO	E21	5698	1947-05-05	FIELDREP	16	M	1926-05-17	23840.00	500.00	1907.00

Exit   Print

The result is also written to the file "Print.txt" in directory "workfiles" and also to the file "EMPLOYEE.sql" in directory "printfiles". Names of columns from the table EMPLOYEE are used as headers.

Now, you may be happy with the result and exit from the list or print the result on a printer using the button *Print*.

To perform the printing, a dialog like this is displayed.

Print

General   Page Setup   Appearance

Print Service

Name: Samsung C460 Series (SEC30CDA71A8153)   Properties...

Status: Accepting jobs

Type:

Info: Samsung C460 Series (SEC30CDA71A8153)   ☐ Print To File

Print Range
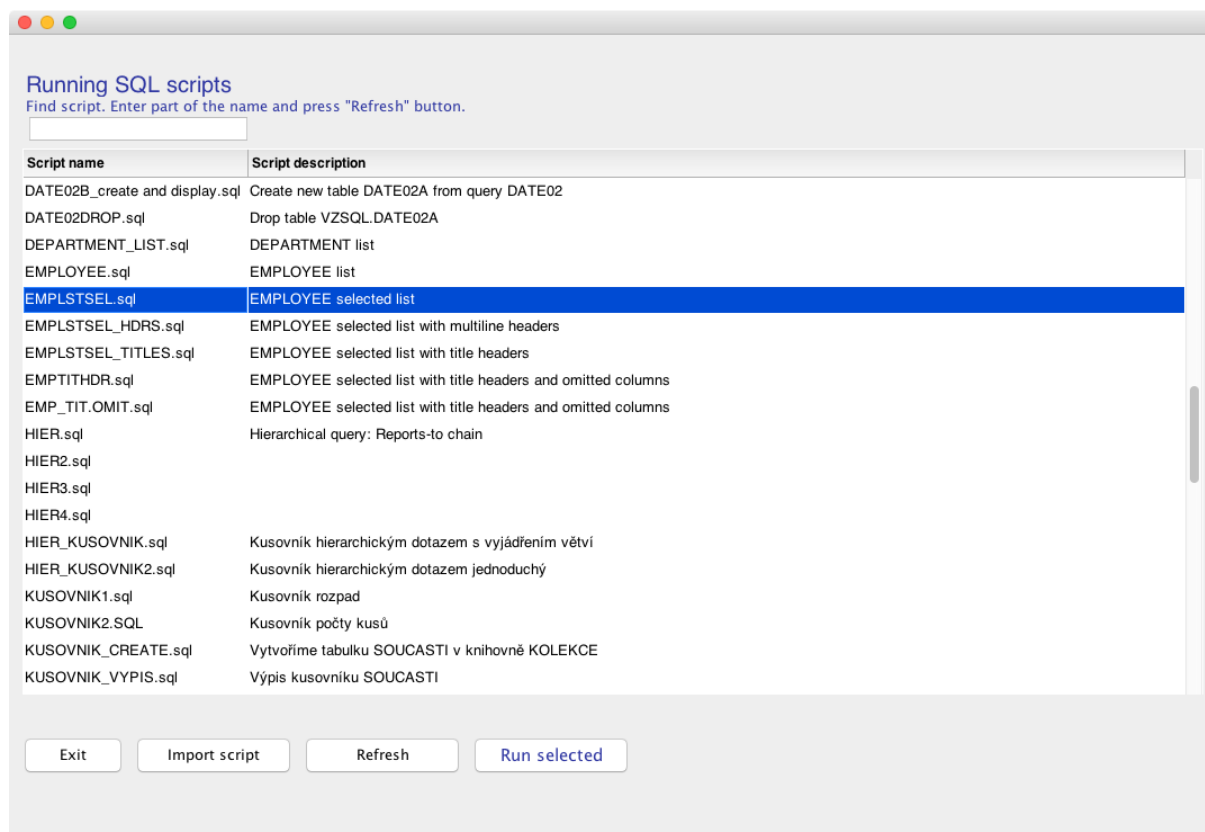 ● All
  ○ Pages   1   To   1

Copies
  Number of copies: 1
  ☐ Collate

Print   Cancel

Query with variable parameters

An SQL statement can contain parameters depicted by question marks.

Select the row with the script name EMPLSTSEL.sql and run the query script below using the button *Run selected*. This query contains three question marks in the SELECT statement that define places for variable values.

```
--EMPLOYEE selected list

--;? 1 ; DATE ; From date: ; 1925-01-01 ;
--;? 2 ; DATE ; To date  : ; 1940-01-01 ;
--;? 3 ; DEC ; Salary does not reach ; 30000.00

SELECT * FROM EMPLOYEE
WHERE BIRTHDATE BETWEEN ? AND ?
      AND SALARY <= ?
```

Three comment lines begin with four-character symbol **--;?** in the first position and contain four entries delimited by semicolon. They enable building a prompt window for the user to enter variable values for question marks.

The *first* entry is a number that must exactly correspond to the sequence of the corresponding question mark. Here, number 1 stands for the first, number 2 for the second, and number 3 for the third question mark.

The *second* entry is the data type of the SQL column (here DATE and DEC). It serves for a preliminary check of the entered value and for documentation. The entry may be empty, but the delimiting semicolon must be present.

The *third* entry is a explaining text for the value to be entered, here it is "From date:", etc. This entry may be empty but with the delimiting semicolon.

8

The *fourth* entry is a default value of the parameter. It must correspond to the data type, here it is the date 1925-01-01 in ISO format, etc. This entry may or may not end with a semicolon.

After clicking on the button *Run selected* a window is shown that contains input fields and explaining texts for the user who may change the default values.

Keeping the default values and pressing *Enter* button shows the following window with the query result.

The user may, like in the previous example, be happy with the result and exit the query, or print the result on the printer.

Non-query script

As an illustration a few SQL statements are entered in the script UTF_8 CREATE2:

```
--Create table PRICES with UTF-8 columns
DROP TABLE VZTOOL.PRICES ;
CREATE TABLE VZTOOL.PRICES
   ( ITEM CHAR(5),
     PRICE DEC(12, 2),
     DESCR CHAR(50) CCSID 1208
   );
SET SCHEMA DEFAULT
```

Every statement but the last is ended by semicolon. DROP and CREATE statements contain the table name PRICES qualified by the schema VZTOOL.

If there is an error in a statement (in the DROP statement the table does not exist yet), a window showing the statement text and the error message is displayed. After pressing *Exit* button, the program continues.
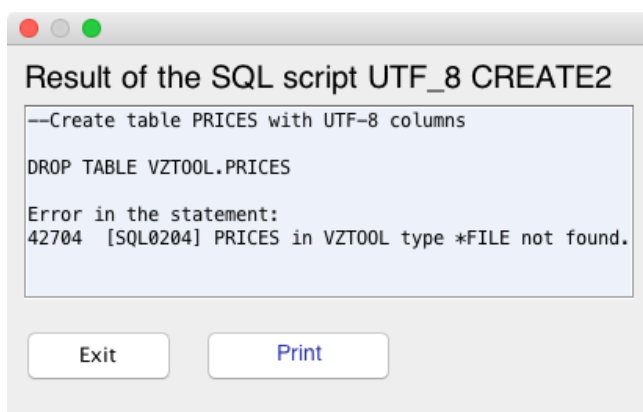
If there is no other error, the remaining statements are performed. Finally a window with summary of all statements and messages is displayed.
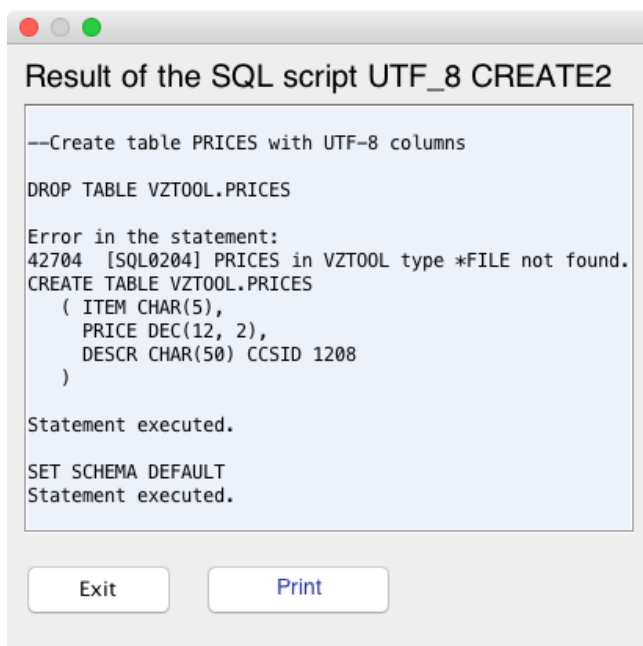
Screens for the application administrator

The first screen of the application is a menu of a set of functions.
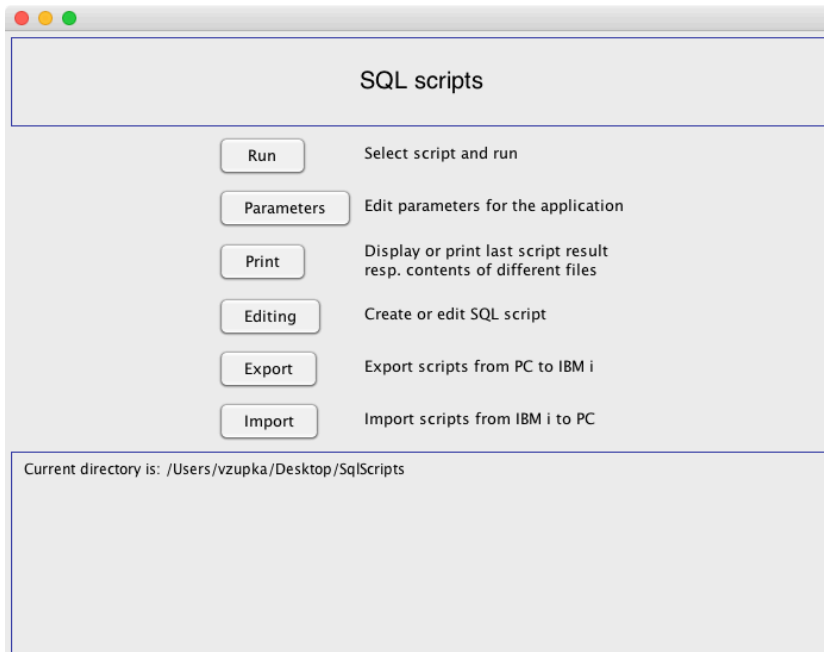
Run function

The program first shows the signon dialog to access system IBM i.
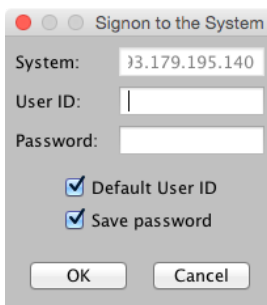
After successful signon the *list of scripts* is shown with buttons and the input field for *selection* of scripts *by a part of the name*. If the user enters a text in the input field and presses the *Refresh* button, only scripts containing the text (ignore case) in the name are listed.

If the text *emp* is entered in the input field, only scripts with the text EMP (with any letter case) in their names are listed.
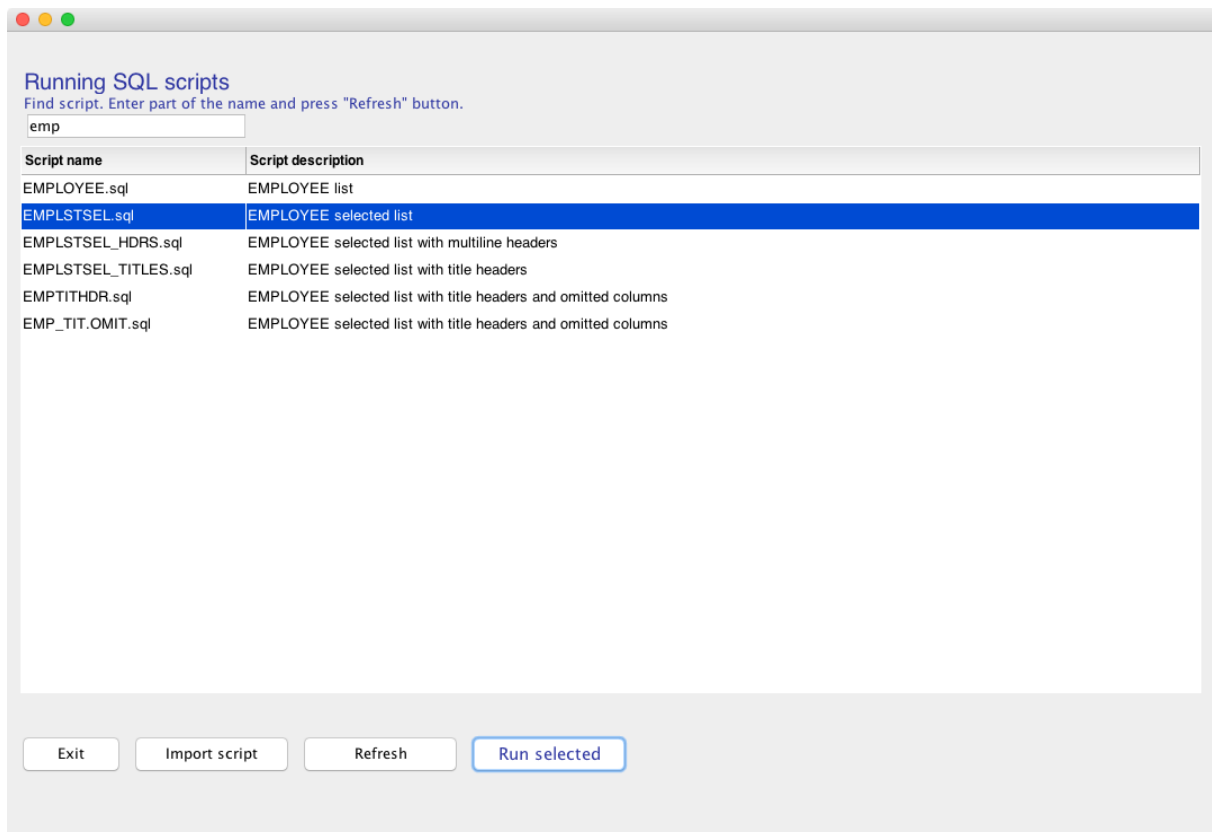
Run selected

The user selects one of the scripts from the list and starts it clicking the button *Run selected*.

Import script

The user can import an *updated* or a *new script* from the central repository. The central repository is an IFS directory defined by the application administrator (see Parameters function below). The imported script is stored to the local directory "scriptfiles".

If the script EMPLSTSEL.sql is selected, then the button *Import script* invokes the signon dialog (if the user is not already signed on). After the user is signed on, a prompt is shown with the file name (ie. script name with .sql suffix) already entered.
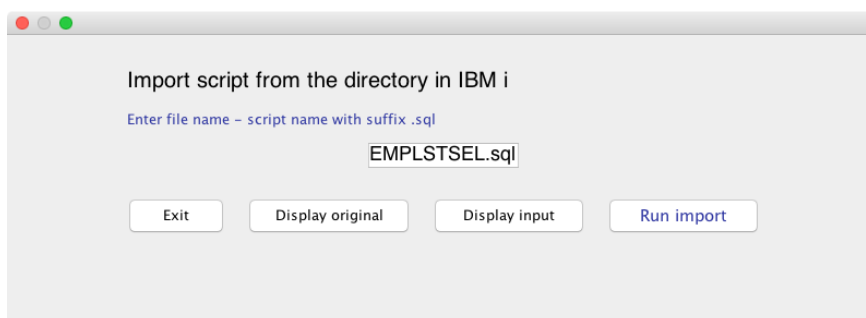
On pressing *Run import* button the program imports the script and a message is displayed.
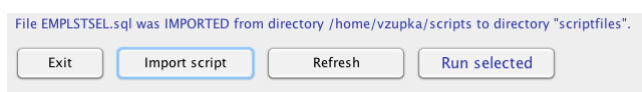
If no row is selected from the list, a prompt for import with an empty new script (the suffix .sql only) is shown.

The user should enter a name of the script to import, e. g. DPTLST.sql, and keep the .sql suffix.

The button *Run import* or *Enter* key starts transfer from IBM i.

The button *Display original* returns the text ".sql".

The button *Display input* makes visible data entered in the accommodated input field.

Parameters function

All users may enter a profile name for access to the system IBM i. This name will be used in the *Signon to the System* dialog as a default.

The *application administrator* can change all values but first of all he must enter

- a server address,
- a list of libraries that contain database tables,
- path to an IFS directory for central scripts repository.

Ordinary users (see Screens for the user below) may choose

- the application language (English or Czech),
- automatic size or fixed dimensions of the result window,
- number of spaces delimiting columns in the query result,
- a font size for printed output.

The *script creator* defines (for all scripts)

- a default mark for null values of columns in the query result,
- a default pattern (mask) for formatting decimal numbers.

Application language

The application can be run in US English (en_US) or Czech (cs_CZ) localization.
Localization concerns titles, messages, button texts, and formatting of decimal numbers in the

script result, and data and time in the result header. After changing this entry the option is fully applied after restarting the application.

Server address

Enter single IP address in dot or domain form.

User name

The administrator enters a profile name with authorization to write files to the IFS directory. If manipulation SQL statements (other than SELECT) are to be run, the corresponding authorization is necessary.

List of libraries with database tables

The application uses naming convention "system" for SQL statement processing. The administrator enters one or several library names separated by a blank or a comma. *LIBL can also be used.

At the time of connecting the database to the program a library list is created that replaces the library list of the database server job. The libraries are searched in the sequence when the SQL statement is processed. If the first entry is *LIBL, the specified list is added to the library list of the database server job.

Note: A library (schema) name qualifying an SQL object in the script proper takes precedence, of course. The separator of the qualifying library can be either a *dot* (.) or a forward *slash* (/). As soon as such a statement is used, subsequent running statements search objects in this library (schema) only. If they do not find them there, the system reports an error message. To remedy this situation you can:

- Write statement SET SCHEMA DEFAULT at the end of the script; it activates the original library list.
- Create a script of a single statement SET SCHEMA DEFAULT and run it before running the statement that did not find the required objects.

IFS directory for central scripts repository

The IFS directory is used as a central repository to *export* and *import* of *scripts* to/from the IBM i server. If no path to a directory is entered these functions cannot be used. Path to the directory must begin and may or may not end with a forward slash /. The application ensures correctnes of the path ending.

The directory must be created beforehand by the CL command CRTDIR (MKDIR, MD), e. g.

```
CRTDIR DIR('/home/vzupka/OlmaOSX')
```

Script texts will be written into the directory in UTF-8 encoding. This encoding can be defined for all the files by the CL command

```
CHGATR OBJ('/home/vzupka/OlmaOSX') ATR(*CCSID) VALUE(1208) SUBTREE(*ALL)
```

Browsing of script texts will then be easier in 5250 emulation using CL command WRKLNK.

Size of the window with the script (query) results

If the checkbox "Automatic size of the window with query results" is selected the result window accommodates to the dimensions of the displayed results. Otherwise, the window will display in the dimensions entered in the fields "Width of the window with query results" and "Height of the window with query results" with sliders, if necessary.

Note: If the input value is not an integer, it is changed to 0.

Mark for null column values

The text (mark) entered in this field (or empty text) will be displayed (and printed) in the query result for all NULL column values.

Note: A text for null column values can also be defined in each individual script using a comment definition line (see below).

Number of spaces separating columns in the query result

This entry designates number of spaces that are appended to each column in header lines and data lines in the query result text. Can also be 0. If the input value is not an integer, it is changed to 0.

Note: The number of spaces can also be defined in each individual script using a comment definition line (see below).

Size of the font in print points

This entry is the number of points, where one point has the size 1/72 of inch. It is applied in printing on paper. If the input value is not an integer, it is changed to 0.

Note: The font size can also be defined in each individual script using a comment definition line (see below).

Pattern for printing numbers

This entry is the pattern (mask) for printing numbers which defines the output picture (format) of decimal numbers in all scripts. A common pattern can be #.00 that suppresses leading zeros and keeps two decimal positions including trailing zeros. If an empty value is entered in this field, the patterns defined for individual columns are applied, or the rule for standard output of numbers will hold (see below).

Note: The pattern can also be defined for individual columns in each script using a comment definition line (see below). The individual pattern always takes precedence.

Edit function

A table with the list of scripts is displayed which is empty initially. The user can shorten the (non-empty) list using an entry in the text field and the button *Refresh*. For example, entry *emp* limits the list to script names containing *emp* (case ingnored).

This function allows to create a new script, edit an existing script, save a script to the server (IFS directory defined in Parameters), delete a script, and read a script to the local directory from the server.

Write text of the script to the empty window or copy it from another source and enter its name in the input field keeping the *.sql* suffix (.SQL, .Sql, ...).

The user can edit the text like in common text editors: copy, cut, paste using key shortcuts. Also *font size* can be changed and two buttons for correcting can be used: buttons *Undo* and *Redo*.

The script text is saved using the button *Save script*. If the script name in the input field is changed, the script will be saved under the new name. The existing script will be kept unchanged. The button *Run script* also saves the script and then runs it.

Save the script using the button *Save script*. The button *Run script* also saves the script and also runs the script. The button *Exit* returns to the preceding window but does not save the text.

<u>Note:</u> The script name may contain only *ASCII* characters and must end with *.sql*.

Edit selected

```
●●●

Edit the SQL script and press Save or Run script.

EMPLSTSEL.sql                              Font size: 14    [ Undo ]    [ Redo ]

  --EMPLOYEE selected list

  --;? 1 ; DATE ; From date: ; 1925-01-01 ;
  --;? 2 ; DATE ; To date  : ; 1940-01-01 ;
  --;? 3 ; DEC ; Salary does not reach ; 30000.00

  SELECT * FROM EMPLOYEE
  WHERE BIRTHDATE BETWEEN ? AND ?
        AND SALARY <= ?




                  [ Exit ]    [ Save script ]    [ Run script ]

```

The selected script is displayed in the text area and its name is displayed in the input field. The user can edit the text the same way as when creating a new script (see above).

If the script name in the input field is changed before it is saved, the script will be saved under the new name and the existing script will be kept unchanged.

Refresh

This button clears the message (if present) and cancels any selection of a script. If a text is entered in the text field only script names containing this text (ignore case) will be displayed in the list. If the text field is empty all scripts are displayed.

Save to server

This button invokes the signon dialog to access system IBM i (if the user is not already signed on). Then the *selected script* is transferred from the local directory "scriptfiles" to the IFS directory (defined in application Parameters). A message about the transfer is displayed.

Read from server

This button invokes the signon dialog to access system IBM i (if the user is not already signed on). Then the *selected script* is transferred from the IFS directory (defined in application Parameters) to the local directory "scriptfiles". A message about the transfer is displayed.

Delete selected

The selected script is deleted from the local directory "scriptfiles" (without any confirmation). If the user is not already signed on to IBM i, the signon dialog to access system IBM i is invoked. If the user cancels the dialog, the script is *not deleted* from the IFS directory. If the user is already signed on, the script is *deleted* from the IFS direcrory (provided it was there). If no scrit is selected from the list, a message is displayed and no script is deleted.

Export function

This function invokes the signon dialog to access system IBM i (if the user is not already signed on). Then *all scripts* are transferred from the local directory "scriptfiles" to the IFS directory (defined in application Parameters). A protocol about transfer is displayed. If the transfer is not completed (or is canceled) the program displays a message.

Import function

This function invokes the signon dialog to access system IBM i (if the user is not already signed on). Then *all scripts* are transferred from the IFS directory (defined in application Parameters) to the local directory "scriptfiles". A protocol about transfer is displayed. If the transfer is not completed (or is canceled) the program displays a message.

Screens for the user
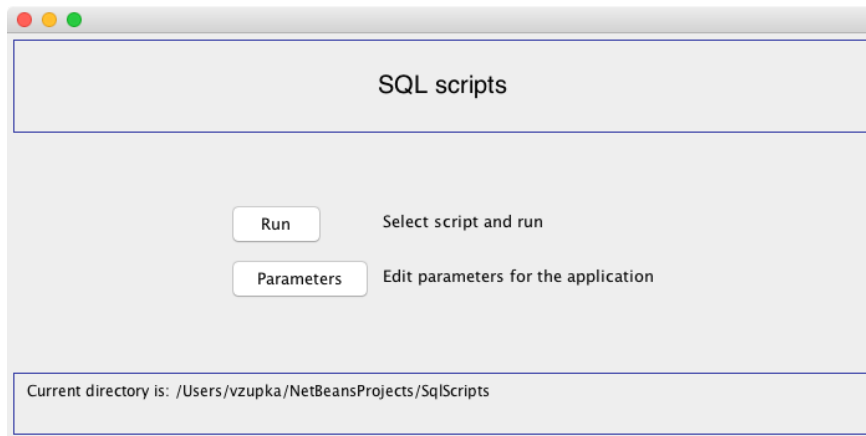
This main screen is shortened version of the one designed for the application administrator.

Run function

This function is identical to the one described above.

Parameters function
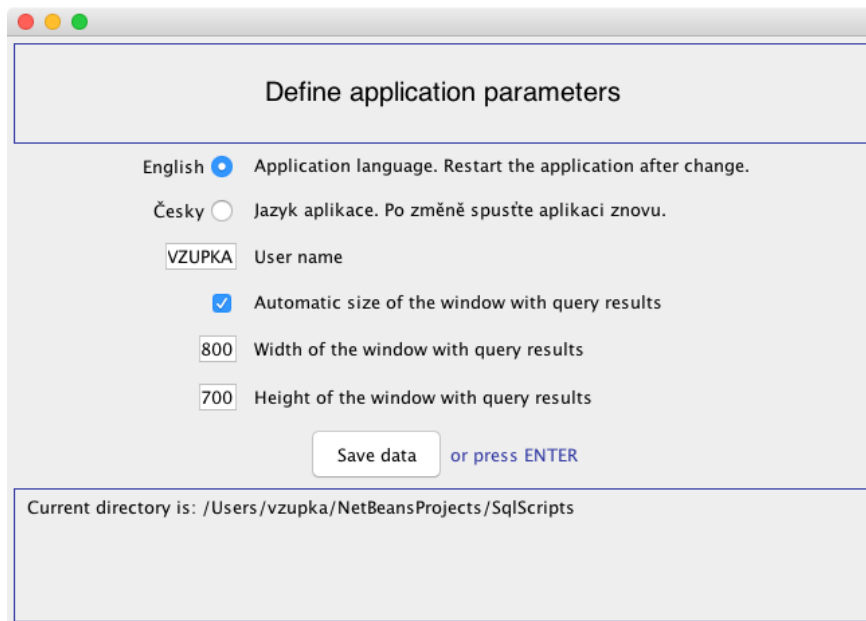
This Parameters function is a simplified version of the one designated for the application administrator. It offers only those parameters the ordinary user makes use of. The parameters are the same as those described above.

Rules for writing scripts

A script is composed of one or more SQL statements delimited by a semicolon. The last statement ends with the last text line (not a semicolon). The script can contain comments – simple and bracketed.

The s*imple comment* begins with two consecutive hyphens (--) and ends with end of line. Simple comments are used in this application for special parameters for the program that evaluates the parameter values before or after the script run.

The *bracketed comment* begins with /* and ends with */ like in different programming languages. It can be placed in the script where the space is required by the statement. It is not used for parameter specification.

The *first line of the script* beginning in the position 1 of the line with the two dash characters (--) is considered *script description*. The text after the two dashes is displayed in the list of scripts as the script description.

The important part of scripts are *special comment lines* beginning in the position 1 of the line with special 4-character symbols. They represent the following parameters:

--;?	variable parameter values (question marks in the SQL statement),
--;t	title header,
--;H	header line for columns in the query result,
--;T	vertical formatting of total (group) lines and symbol for null column values,
--;D	pattern (mask) for output of decimal numbers (DECIMAL and NUMERIC types),
--;O	list of columns to be omitted from the result output,
--;L	group level, separation text and grouping column name,
--;S	summarized column name amd list of summary types (S, A, M, m, C),
--;s	list of leading texts for summary lines (replacing standard SUM, AVG, MAX, MIN, COUNT),
--;P	print parameters.

These four-character symbols were selected so that any characters might follow (except for the semicolon that divides the values).

Definition of variable parameter values

Question marks called *parameter markers* in the SQL statement text serve as placeholders for variable values. Values are assigned to the SQL statement before it is performed. Special comment definition lines serve to this purpose. Number and sequence of the definition lines determine the number and sequence of the input fields in a prompt window. The definition comment line begins with **--;?** in the first position. Format of the definition line is as follows. Entries in brackets are optional.

```
--;? sequence-number; [data-type]; [explanation]; [default-value][;]
```

Number of definition lines must be the same as the number of question marks. Each definition line contains exactly four entries delimited by a semicolon. Leading and trailing spaces in the entries are ignored.

Note: If the script contains more than one SQL statement *only one* SQL statement can make use of parameter markers.

Entries in the definition line

Sequence number

The first entry is a mandatory sequence number which must exactly match the sequence of the corresponding question mark in the SQL statement. Sequence number 1 applies for the first question mark, 2 for the second question mark etc. The sequence of input fields in the window is determined by the sequence of definition lines in the script, not by this number.

Data type

The second entry is optional and designates one of the following data types.

DEC
DECIMAL
NUMERIC
INT
INTEGER
BIGINT
DATE
TIME
TIMESTAMP

The listed types serve for preliminary check of the value entered in the input field of the prompt window, and also for documentation. If one of these types is entered the check is performed *before the SQL statement is run*. If this entry is not present or if another text is entered, the check of the type is done only *at the time of performing of the SQL statement*. This entry may be empty but the semicolon must be present.

Explanation text

The third entry is an optional explanation text to the parameter value, e. g. *From date: .* This entry can also be empty but it must be ended by a semicolon.

Default value

The fourth entry is a default value corresponding to the data type, e. g. *1925-01-01*. This entry can also be empty and may and may not be ended by a semicolon.

Entries of types DEC, DECIMAL, NUMERIC may contain digits, decimal point and a minus sign before the number.

Entries of types INT, INTEGER, BIGINT may contain digits and a minus sign before the number.

Entries of type DATE must be entered in ISO format, YYYY-MM-DD, e. g. 2014-02-15.

Entries of type TIME must be entered in ISO format, HH:MM:SS, e. g. 19:31:05.

Entries of type TIMESTAMP must be entered in the ISO format,

YYYY-MM-DD HH:MM:SS.MMMMMM, e. g. 2000-04-05 23:59:59.999999

The length of the timestamp is 26 characters. A space is between date and time. Leading zeros can be omitted from the months, days, hours, minutes, seconds. Trailing zeros in microseconds may be partly or completely omitted. The form ending with 24:00:00.000000 is allowed.

Note: If the column type is BINARY or VARBINARY the default value should be entered in hexadecimal characters (0123456789abcdefABCDEF). Two characters correspond to a byte.

Definition lines can be entered in arbitrary sequence; their sequence numbers are important.

Examples of definitions:

```
--;?01 ; ; Price from: ; 5.50;
--;? 02;DEC; Price to:, 23000
--;? 03; DATE; From date:;2014-01-29;
--;? 04;  ; Binary data – two bytes: ; 0F9C;
```

Entering parameter values when running scripts

When the prompting window for entering SQL query parameters is displayed both default data from definitions and data entered to input fields by the user are checked. The data must correspond to the type *entered* in --;? definition comment line. This check is done before the SQL statement is started.

- DEC, DECIMAL, NUMERIC type is checked for the number format. The number must contain only digits, decimal separator and minus or plus sign. Decimal separator, if present, must be a dot (.). The sign must be on the left.
- DATE type is checked against format ISO.
- TIME type is checked against format ISO.
- TIMESTAMP type is checked against format ISO.

If the type is missing or a different text is entered in the specification line, a failure (if any) is reported at the time the SQL statement is performed. The SQL message will be apparent in the window that shows after performing the script.

Note 1: If the column type is BINARY or VARBINARY the value should be entered in hexadecimal characters (0123456789abcdefABCDEF). Two hexadecimal characters correspond to a byte. The value is not checked for correctness. Invalid character is translated to 0.

Note 2: Query results of the BINARY or VARBINARY columns are also shown in the hexadecimal form.

Example of entering parameter data

The following script defines three parameters (question marks).

```
--EMPLOYEE selected list

--;? 1 ; DATE ; From date: ; 1925-01-01 ;
--;? 2 ; DATE ; To date  : ; 1940-01-01 ;
--;? 3 ; DEC ; Salary does not reach ; 30000.00

SELECT * FROM EMPLOYEE
WHERE BIRTHDATE BETWEEN ? AND ?
     AND SALARY <= ?
```

After starting the script a prompt dialog is invoked which allows changing the default parameter values and/or running the SQL statement (button Enter).

If we change the hyphen in the first parameter to underscore and the number in the third parameter to w0000.00, a message is displayed for the error in the *first* parameter.

After correcting the first parameter, an error in the *third* parameter is reported.

If we omit the type DEC in the third parameter definition and make the same error, the script will run and the result shows a copy of the script and the SQL error message.
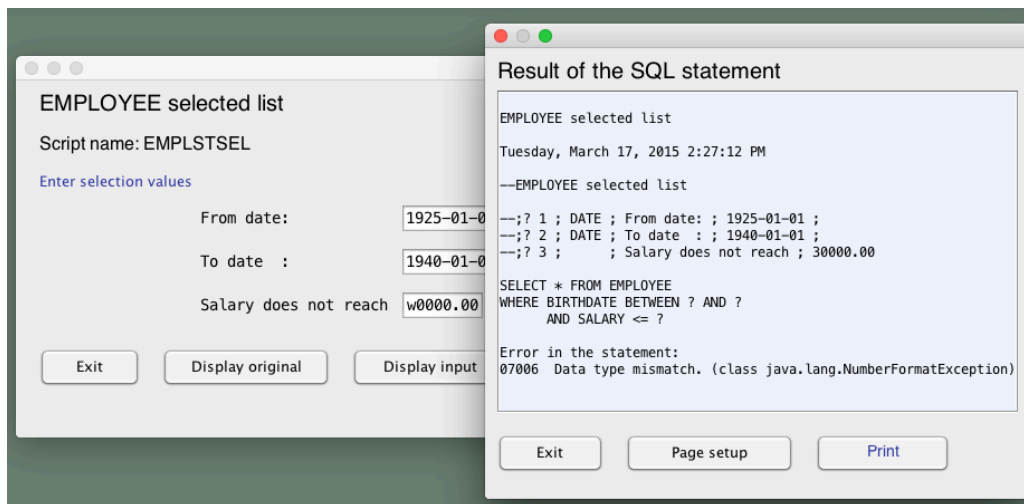
After exiting the prompt dialog is redisplayed with the same error message.
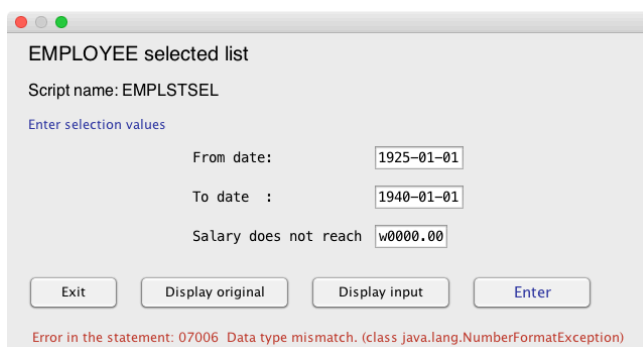
Definition of title headers

Any number of title header lines can be specified for a query script. They are applied in the sequence as written in the script. The title header line specification begins with the symbol `--;t` in the position 1 of the specification line after which any text may follow.

The title header line can contain symbols of variable names that have the following form:

`&column`

where *column* is the column name followed by a space or end of line. The variable is replaced by the column value from the *first line* of the result set even if the column is specified as omitted (see below).

Title headers are written out before column headers and are printed on the first page only.

For example the following specifications

```
--EMPLOYEE selected list with title headers

--;t    Employee number from the first line: &EMPNO , Birth date: &BIRTHDATE
--;t

--;? 1 ; DATE ; From date: ; 1925-01-01 ;
--;? 2 ; DATE ; To date  : ; 1940-01-01 ;
--;? 3 ; DEC  ; Salary does not reach ; 30000.00

SELECT EMPNO,
    FIRSTNME,
    LASTNAME,
```

```
    BIRTHDATE,
    SALARY,
    COMM
FROM EMPLOYEE
    WHERE BIRTHDATE BETWEEN ? AND ?
          AND SALARY <= ?
```

produces title headers with values of columns EMPNO and BIRTHDATE from the first row in the query result:

```
    Employee number from the first line: 000130, Birth date: 1925-09-15
```

```
EMPNO     FIRSTNME      LASTNAME          BIRTHDATE        SALARY          COMM
000130    DELORES       QUINTANA          1925-09-15      23800.00       1904.00
000230    JAMES         JEFFERSON         1935-05-30      22180.00       1774.00
000250    DANIEL        SMITH             1939-11-12      19180.00       1534.00
... etc.
```

Definition of column headers

User headers

Any number of header definition lines can be defined for a query statement (SELECT). The definition lines are applied in the sequence as written in the script. A header definition line begins with the symbol **--;H** after which texts (column titles) delimited by semicolons are entered.

```
--;H [ [text-1] [; text-2] [; ...] [; text-n] [;] ]
```

Texts are taken including leading and trailing spaces. Multiline headers can be aligned or width of displayed columns can be increased this way.

For example the script EMPLSTSEL_HDRS.sql

```
--EMPLOYEE selected list with multiline headers

--;HEmployee;First      ;Mid. ;Last          ;Work;Phone ;Hire     ;Job      ;Ed.;Sex;Birth     ;  Salary;Bonus;Commission
--;Hnumber  ;name       ;init.;name          ;dept;number;date     ;         ;lv.;   ;date      ;        ;     ;
--;H-------;-----------;-----;--------------;----;------;---------;---------;---;---;----------;--------;-----;----------
--;H

--;? 1 ; DATE ; From date: ; 1925-01-01 ;
--;? 2 ; DATE ; To date  : ; 1940-01-01 ;
--;? 3 ; DEC  ; Salary does not reach ; 30000.00

SELECT * FROM EMPLOYEE
WHERE BIRTHDATE BETWEEN ? AND ?
      AND SALARY <= ?
```

produces four column headers in the query result, where the last one is empty:

```
Result of the SQL statement

EMPLOYEE selected list with multiline headers

Wednesday, March 18, 2015 1:03:13 PM

Employee First        Mid.  Last           Work Phone  Hire       Job        Ed. Sex Birth       Salary    Bonus Commission
number   name         init. name           dept number date                  lv.     date
_____ _____ _____ _____ ____ _____ _____ _____  ___ ___ _____  _____  _____ _____

000130   DELORES      M     QUINTANA        C01  4578   1971-07-28 ANALYST     16  F   1925-09-15  23,800    500   1,904
000230   JAMES        J     JEFFERSON       D21  2094   1966-11-21 CLERK       14  M   1935-05-30  22,180    400   1,774
000250   DANIEL       S     SMITH           D21  0961   1969-10-30 CLERK       15  M   1939-11-12  19,180    400   1,534
000260   SYBIL        P     JOHNSON         D21  8953   1975-09-11 CLERK       16  F   1936-10-05  17,250    300   1,380
000280   ETHEL        R     SCHNEIDER       E11  8997   1967-03-24 OPERATOR    17  F   1936-03-28  26,250    500   2,100
000300   PHILIP       X     SMITH           E11  2095   1972-06-19 OPERATOR    14  M   1936-10-27  17,750    400   1,420
000310   MAUDE        F     SETRIGHT        E11  3332   1964-09-12 OPERATOR    12  F   1931-04-21  15,900    300   1,272
000320   RAMLAL       V     MEHTA           E21  9990   1965-07-07 FIELDREP    16  M   1932-08-11  19,950    400   1,596
000340   JASON        R     GOUNOT          E21  5698   1947-05-05 FIELDREP    16  M   1926-05-17  23,840    500   1,907
200280   EILEEN       R     SCHWARTZ        E11  8997   1967-03-24 OPERATOR    17  F   1936-03-28  26,250    500   2,100
200310   MICHELLE     F     SPRINGER        E11  3332   1964-09-12 OPERATOR    12  F   1931-04-21  15,900    300   1,272
200340   ROY          R     ALONZO          E21  5698   1947-05-05 FIELDREP    16  M   1926-05-17  23,840    500   1,907
```

```
Exit          Page setup          Print
```

Standard headers

If there is not any header definition line, the *standard column headers* from the SELECT statement are displayed. The headers are the column names defined in the table resulting from the query. The standard column header may be the original but also renamed column name (simple or enclosed in quotes). All headers in the following script are standard.

```
--DEPARTMENT list
select DEPTNO AS "Dept",
  DEPTNAME AS "Department name",
  MGRNO,
  ADMRDEPT,
  LOCATION
from DEPARTMENT
```

```
Result of the SQL statement

DEPARTMENT list

Tuesday, March 17, 2015 3:34:49 PM

Dept Department name           MGRNO  ADMRDEPT LOCATION

A00  SPIFFY COMPUTER SERVICE DIV.  000010 A00      -
B01  PLANNING                      000020 A00      -
C01  INFORMATION CENTER            000030 A00      -
D01  DEVELOPMENT CENTER            -      A00      -
D11  MANUFACTURING SYSTEMS         000060 D01      -
D21  ADMINISTRATION SYSTEMS        000070 D01      -
E01  SUPPORT SERVICES              000050 A00      -
E11  OPERATIONS                    000090 E01      -
E21  SOFTWARE SUPPORT              000100 E01      -
F22  BRANCH OFFICE F2              -      E01      -
G22  BRANCH OFFICE G2              -      E01      -
H22  BRANCH OFFICE H2              -      E01      -
I22  BRANCH OFFICE I2              -      E01      -
J22  BRANCH OFFICE J2              -      E01      -
```

```
Exit          Page setup          Print
```

Definition of vertical and horizontal division

Only one comment definition line for groups should be specified. If there are more lines, only the first is accepted. The definition begins in the first position of the line by four-character symbol **--;T** and is followed by entries delimited by a semicolon. They define

- number of empty lines before a line or a group of lines,
- number of empty lines after a line or a group of lines,
- symbol replacing the null value of the column,
- number of spaces delimiting columns in the output line - from 0 up,
- one or more column names for suppressing their duplicated values.

```
--;T [ [spaces-before] [; spaces-after] [; null-mark]
      [; spaces-between-columns] [; column-name] ... ] [;]
```

Parameter *spaces-before* has the form **Bn** or **n**, where n is an integer. Parameter *spaces-after* has the form **An** or **n**. Parameter *spaces-between-columns* has the form **Sn** or **n**. The first two parameters may be empty. If one of them is empty or invalid, *zero* is taken. If the parameter *null-mark* or *spaces-between-columns* is empty or invalid, the corresponding value from the application parameters is taken.

The fifth and following parameters specifies the *column name*, that is checked if its value has changed in a group of contiguous rows. The column value will be printed on the first line of such a group while the next lines will contain spaces.

<u>Note:</u> Empty lines are inserted in the query result before or after the lines or line groups resulting from group levels (see below). Empty lines are also inserted before or after individual lines containing any columns with empty (NULL) values when no group levels are not defined.

Examples

The following script contains the clause – function GROUP BY ROLLUP () that groups columns DAY_WEEK and WEEK.

```
--Sales by day-of-week, week, and grand-total

--;T B0 ; A1 ; -

SELECT WEEK(SALES_DATE) AS WEEK,
  DAYOFWEEK(SALES_DATE) AS DAY_WEEK,
  SUM(SALES) AS UNITS_SOLD
FROM SALES
GROUP BY ROLLUP( WEEK(SALES_DATE), DAYOFWEEK(SALES_DATE) )
ORDER BY WEEK, DAY_WEEK
```

This script produces *one* empty line *after* the data line, while the symbol for the null column value is a *hyphen* (character -).

```
● ○ ●
Result of the SQL script SALES_GROUPS

Sales by day-of-week, week, and grand-total

Thursday, December 17, 2015 12:13:19 PM

     WEEK      DAY_WEEK    UNITS_SOLD
      13           6            27
      13           7            46
      13           -            73

      14           1            31
      14           2            43
      14           -            74

      53           1             8
      53           -             8

       -           -           155


    Exit            Print
```

If we change the definition to specify *one* empty line *before* the data line, *one* empty line *after* the data line, *space* for null values, *three column separating spaces*, and column WEEK to *blank out repeated values*:

--;T B1 ; A1 ; ; S3 ; week;

the result will look like this:

```
● ○ ●
Result of the SQL script SALES_GROUPS

Sales by day-of-week, week, and grand-total

Thursday, December 17, 2015 12:17:30 PM

     WEEK      DAY_WEEK    UNITS_SOLD
      13           6            27
                   7            46

                                73

      14           1            31
                   2            43

                                74

      53           1             8

                                 8


                               155


    Exit            Print
```

Formatting numbers for output

Decimal numbers of the type DEC, DECIMAL, NUMERIC are displayed and printed using a format. A format is either *standard* for a given localization or prescribed by a *pattern (mask)*.

Note: Patterns do not apply to integer types (INT, INTEGER, BIGINT).

Symbols for creating patterns

Characters comprising a patern are shown in the following table along with their placement and meaning.

Symbol	Location	Localized?	Meaning
0	Number	Yes	Digit
#	Number	Yes	Digit, zero shows as absent
.	Number	Yes	Decimal separator or monetary decimal separator
-	Number	Yes	Minus sign
,	Number	Yes	Grouping separator
%	Prefix or suffix	Yes	Multiply by 100 and show as percentage
¤	Prefix or suffix	No	Currency sign, replaced by currency symbol. If doubled, replaced by international currency symbol. If present in a pattern, the monetary decimal separator is used instead of the decimal separator.
‰	Prefix or suffix	Yes	Multiply by 1000 and show as per mille value
'	Prefix or suffix	No	Used to quote special characters in a prefix or suffix, for example, "'#'#" formats 123 to "#123". To create a single quote itself, use two in a row: "# o''clock".

<u>Warning:</u> Placing 0 between sequence of # and vice versa is an error.

Examples (US localization)

Number	Pattern (mask)	Output	Comment
123456789.50	#,####.00	1,2345,6789.50	Separates groups of 4 digits in the whole part
1234567.50	,###.00 ¤	1,234,567.50 $	Separates groups of 3 digits in the whole part
1234567.50	#.00 ¤¤	1234567,50 USD	Adds the international currency symbol for dollar
1234567.50	#.00 USD	1234567.50 USD	Adds constant "USD" after the number.
1234567.50	¤#.00	$1234567.50	Floating currency symbol
1234567.50	#0**.00	1234567.50**	Asterisks are added after the number with no space
-1234567.55	#0.0	-1234567.6	Rounds to 1 decimal position (to even)
-0.04	#.00	-.04	Suppresses zero before decimal point
5.55	00000.000	00005.550	Does not suppress leading zeros and adds a 0 digit at the end
0.56	#.00 %	56 %	Expressing the fraction as percentage
0.56	#.00 %	%56	Floating percet sign on the left
0.007	#.0	7,0 ‰	Expressing the fraction as per mille
12345	-#.00	-12345	Floating minus sign on the left. Does not invert positive number to negative.
-12345	-#.00	--12345	If the number is negative adds one more minus sign.

Defining patterns

A pattern may be defined in application Parameters, or in comment definition lines in the script, or both methods may be combined. The pattern defined in Parameters applies for all numeric columns of the script for which no individual definition is defined in a comment definition line.

A *pattern definition line* begins in the first position with the symbol **--;D** which is followed by two entries: a pattern and a column name.

Pattern definition has the following format.

```
--;D pattern ; column-name [;]
```

Example of currency symbols
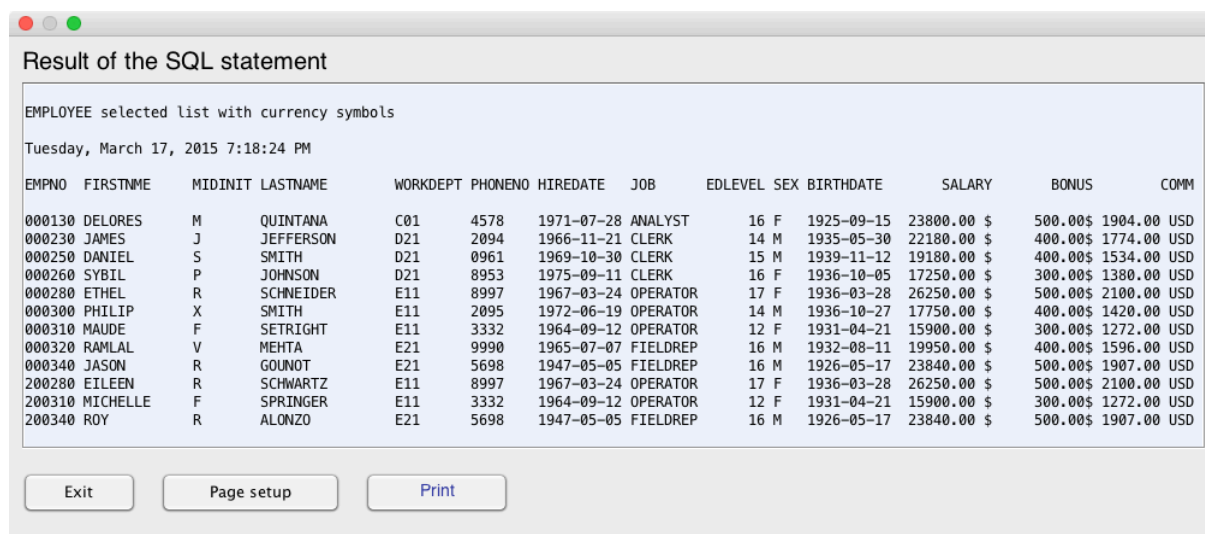
```
--EMPLOYEE selected list with currency symbols

--;? 1 ; DATE ; From date: ; 1925-01-01 ;
--;? 2 ; DATE ; To date  : ; 1940-01-01 ;
--;? 3 ;      ; Salary does not reach ; 30000.00

--;D #.00 $ ; SALARY
--;D #.00¤ ; bonus
--;D #.00 ¤¤; Comm

SELECT * FROM EMPLOYEE
WHERE BIRTHDATE BETWEEN ? AND ?
     AND SALARY <= ?
```

This script defines three patterns for three columns and produces the following output.

Formatting numbers without patterns

If no pattern is defined neither in application Parameters nor in the definition lines, the decimal number is formatted according to *standard rules* in a given locality. For USA these rules apply:

- Whole part of the number is divided into comma separated groups of 3 digits.
- Decimal fraction lacks trailing zeros.
- Zero decimal fraction causes suppression of the decimal point.

Example of standard localization of decimal numbers

If the pattern in the application Parameters is *empty* and no individual pattern definitions are defined in comment definition lines, numbers in the the output of the script EMPLSTSEL in US standard localization will look like this:

```
Result of the SQL statement

EMPLOYEE selected list

Tuesday, March 17, 2015 7:43:26 PM

EMPNO  FIRSTNME    MIDINIT LASTNAME    WORKDEPT PHONENO HIREDATE   JOB       EDLEVEL SEX BIRTHDATE  SALARY  BONUS   COMM

000130 DELORES     M       QUINTANA    C01      4578    1971-07-28 ANALYST       16 F   1925-09-15  23,800  500    1,904
000230 JAMES       J       JEFFERSON   D21      2094    1966-11-21 CLERK         14 M   1935-05-30  22,180  400    1,774
000250 DANIEL      S       SMITH       D21      0961    1969-10-30 CLERK         15 M   1939-11-12  19,180  400    1,534
000260 SYBIL       P       JOHNSON     D21      8953    1975-09-11 CLERK         16 F   1936-10-05  17,250  300    1,380
000280 ETHEL       R       SCHNEIDER   E11      8997    1967-03-24 OPERATOR      17 F   1936-03-28  26,250  500    2,100
000300 PHILIP      X       SMITH       E11      2095    1972-06-19 OPERATOR      14 M   1936-10-27  17,750  400    1,420
000310 MAUDE       F       SETRIGHT    E11      3332    1964-09-12 OPERATOR      12 F   1931-04-21  15,900  300    1,272
000320 RAMLAL      V       MEHTA       E21      9990    1965-07-07 FIELDREP      16 M   1932-08-11  19,950  400    1,596
000340 JASON       R       GOUNOT      E21      5698    1947-05-05 FIELDREP      16 M   1926-05-17  23,840  500    1,907
200280 EILEEN      R       SCHWARTZ    E11      8997    1967-03-24 OPERATOR      17 F   1936-03-28  26,250  500    2,100
200310 MICHELLE    F       SPRINGER    E11      3332    1964-09-12 OPERATOR      12 F   1931-04-21  15,900  300    1,272
200340 ROY         R       ALONZO      E21      5698    1947-05-05 FIELDREP      16 M   1926-05-17  23,840  500    1,907


    Exit            Page setup          Print
```

Decimal numbers show without fractions and are divided into comma separated groups of 3 digits. Other standard rules apply to percentages, currency symbols etc.

Omitting columns from output

The definition line begins in the position 1 with the symbol **--;O** which is followed by column names delimited by semicolon. Any number of definition lines may be specified.

The definition has the following format.

```
--;O [ column-name ; column-name ; ... [;] ]
```

Columns specified in this specification are omitted from the display and print which concerns both values and column headers.

Example with omitted columns and title headers

Script EMP_TIT.OMIT.sql specifies three title headers, two omitted columns along with some question mark parameteres.

```
--EMPLOYEE selected list with title headers and omitted columns

--;? 1 ; DATE ; From date: ; 1925-01-01 ;
--;? 2 ; DATE ; To date   : ; 1940-01-01 ;
--;? 3 ; DEC ; Salary does not reach ; 30000.00

--;tEmployee number: &EMPNO , Bonus: &BONUS
--;tBirthdate: &BIRTHDATE
--;t

--;O EMPNO ; MIDINIT;
--;O BONUS; BIRTHDATE

SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME,
       BIRTHDATE, SALARY, BONUS, COMM
   FROM EMPLOYEE
   WHERE BIRTHDATE BETWEEN ? AND ?
         AND SALARY <= ?
```

Omitted columns are not included in the output but their values from the first row are visible in title headers:

```
EMPLOYEE selected list with title headers and omitted columns

Thursday, December 10, 2015 2:25:55 PM

Employee number: 000130, Bonus: 500.00
Birthdate: 1925-09-15

FIRSTNME        LASTNAME            SALARY          COMM
DELORES         QUINTANA          23800.00       1904.00
JAMES           JEFFERSON         22180.00       1774.00
DANIEL          SMITH             19180.00       1534.00
SYBIL           JOHNSON           17250.00       1380.00
ETHEL           SCHNEIDER         26250.00       2100.00
PHILIP          SMITH             17750.00       1420.00
MAUDE           SETRIGHT          15900.00       1272.00
RAMLAL          MEHTA             19950.00       1596.00
JASON           GOUNOT            23840.00       1907.00
EILEEN          SCHWARTZ          26250.00       2100.00
MICHELLE        SPRINGER          15900.00       1272.00
ROY             ALONZO            23840.00       1907.00
```

Editing of BONUS is now performed according to the specification in application parameters which is #.00.

Note: There is no necessity to bind title headers to omitted columns. Both can be used independently.

Print parameters

The definition line begins in the position 1 with the symbol **--;P** which is followed by entries delimited by semicolon. Only one definition line may be specified. If more are specified, the first one is taken, others are ignored.

The entries have the following meaning.

- paper size A4 or A3,
- font size of the form FSn, where n is number of print points, e. g. fs9,
- page orientation PORTRAIT or P, or LANDSCAPE or L,
- left margin of the form LMn, where n is number of milimeters from the edge of paper printable area,
- right margin of the form RMn,
- top margin of the form TMn,
- bottom margin of the form BMn,

The definition has the following format.

```
--;P  An; FSn; orientace; LMn; RMn ; TMn ;BMn [;]
```

The line may or may not be ended by semicolon. Individual entries may be omitted but number of semicolons before the first non-empty entry must be retained.

- If paper size is empty or invalid A4 is taken.
- If font size is empty or non-numeric the value from the application parameters is taken.
- If orientation is empty or invalid PORTRAIT is taken.
- If a margin is empty or invalid 10 is taken (10 mm).

Summarization of query result

The result of a query (the result set) can be summarized according to group levels (level summarization). Number of levels is not explicitly limited. The group summarization is defined by the set of three parametric definitions:

--;L   group level, its leading text and the name of its group column,
--;S   name of the summarized column and a list of summarization types (S, A, M, m, C),
--;s   list of leading texts for summary lines (instead of standard SUM, AVG, MAX, MIN, COUNT),

Number of group levels is defined by the number of --;L lines. Number of summarized columns is defined by the number of --;S lines. At most one --;s line can be specified. Lines --;S and --;s have some sense only when some --;L definitions are specified.

Definitions of group levels

The definition line begins in the position 1 with the symbol **--;L** which is followed by entries delimited by semicolon.

The definition has the following format.

```
--;L [level]; [group-leading-text]; [level-column]; [NP] [;]
```

The first entry *level* is either 0 or any text (even empty).

The second entry *group-leading-text* is any text containing optional variables of the form **&column** followed by a space or end of line. The variable is replaced by the column value from the *last line* of the group even if the column is specified as omitted in the --;O parameters.

The third entry *level-column* represents the column defining the level group. This entry has no meaning in level 0.

The fourth entry NP (New Page) is applied when printing on paper. If the group with this entry changes, the first row of the new group is printed on the next page.

L0 entry defines the last level called final total or grand total. L0 specification line is optional but if present it must be the first (or the only) one. If L0 is not present the last level is not processed. The *level* entry in other ("lower") group levels will usually be the serial number of the level but it can be any text. For example, the following specification lines define two group levels L2 and L1, and the last level L0.

```
--;L0 ;*** GRAND TOTAL ***
--;L1 ;=== &SALES_PERSON ===            ; SALES_PERSON
--;L2 ;--- &SALES_PERSON  --- &REGION ; REGION
```

Hierarchy of levels is determined by the sequence of specification lines, not by the number at the letter L. The last line defines the lowest level.

Definitions of summarized columns

In order to summarize (accumulate), colum names with their summarizing type(s) must be specified along with some group levels (L0, L1, ...).

One or more definition lines --;S is needed to summarize column values. The definition lines need not be specially ordered. Each definition line begins in the position 1 with the symbol **--;S** which is followed by entries delimited by semicolon. The entries define a column name and the type of summarization of its values. The definition has the following format.

```
--;S [summarized-column; [; ind] [; ind] [; ind] [; ind] [; ind] [;]]
```

where *ind* is one of the symbols – indicators S, A, M, m, C. They represent summarization types *sum*, *average*, *maximum*, *minimum* and *count*. These indicators need not be entered in any specific order and they need not be specified at all. Summary lines in the query result are ordered in this specific order: S - sum, A - average, M - maximum, m - minimum, C - count).

If empty or invalid value is in one of indicator's positions, it is assumed not being specified.

If no specification line --;S is present, no level processing is done even if specifications --;L are present and vice versa.

All five types of summarization can be done for columns of types DECIMAL, NUMERIC, INTEGER, SMALLINT a BIGINT. Only maximum, minimum and count can be used for types CHAR, VARCHAR, VARGRAPHIC, DATE, TIME, and TIMESTAMP.

Null values are excluded from summarization.

<u>Note:</u> If, by mistake, sum (S) or average (A) is specified for *non-numeric* column, no value is computed and printed but the line with the summary type indication (see below) is printed (even if no *numeric* column is summarized).

For example, the specification

```
--;S SALES_DATE ; M ; m
--;S SALES ; S ; A ; M ; m ; C
```

define two summarized columns:

SALES_DATE – evaluated for maximum (M) and minimum (m),
SALES – evaluated for sum (S), average (A), Maximum (M), minimum (m), and count (C).

Summary type indications

Standard summary type indications are SUM for sum, AVG for average, MAX for maximum, MIN for minimum, and COUNT for count. These indications can be replaced by different texts using specification line --;s.

The definition line begins in the position 1 with the symbol **--;s** which is followed by entries delimited by semicolon.

The definition has the following format.

```
--;s [ [sum-text]; [avg-text]; [max-text]; [min-text]; [count-text] [;] ]
```

where the texts from corresponding positions replace standard indications. If the text is missing in a position, standard indication for the position is taken.

For example, the specification line

```
--;s Sum of sales: ; Average: ; Maximum: ; Minimum: ;  Number of days
```

replaces all standard indications.

Example

The following script specifies user headers (--;H), vertical structure (--;T), three group levels (--;L) over REGION, SALES_PERSON and final total, and one summarized column SALES with all possible summary types. Specification --;s changes standard summary indications.

```
--Sales summarized by person and region

--;HSales date;Sales person;Region;Sales
--;H----------;------------;------;-----
--;H

--;T 1; 1; null ;  ; SALES_PERSON; REGION

--;L0 ;*** GRAND TOTAL ***;
--;L1 ;=== &SALES_PERSON ===           ; SALES_PERSON
--;L2 ;--- &SALES_PERSON  --- &REGION ; REGION

--;S SALES ; S ; A ; M ; m ; C

--;s Sum of sales: ; Average: ; Maximum: ; Minimum: ;  Work shifts:

SELECT  SALES_DATE , SALES_PERSON , REGION , SALES
FROM SALES
ORDER BY
    SALES_PERSON , REGION , SALES_DATE
```

The script produces the following result.

```
Sales date    Sales person    Region              Sales
----------    ------------    ------              -----


1996-03-29    GOUNOT          Manitoba                7
1996-03-30                                            1
1996-04-01                                            7

--- GOUNOT --- Manitoba
                              Sum of sales:          15
                                   Average:           5
                                   Maximum:           7
                                   Minimum:           1
                              Work shifts:            3

1996-04-01                    Ontario-North           1

--- GOUNOT --- Ontario-North
                              Sum of sales:           1
                                   Average:           1
                                   Maximum:           1
                                   Minimum:           1
                              Work shifts:            1

1996-03-29                    Ontario-South           3
1996-03-30                                            2
1996-03-31                                            2
1996-04-01                                            3


... etc.


1995-12-31                    Ontario-South           1
1996-03-29                                            3
1996-03-30                                            1
1996-04-01                                            3

--- LUCCHESSI --- Ontario-South
```

```
                              Sum of sales:                 8
                                  Average:                  2
                                  Maximum:                  3
                                  Minimum:                  1
                              Work shifts:                  4


1996-03-29                    Quebec                        1
1996-03-30                                                  2


--- LUCCHESSI --- Quebec
                              Sum of sales:                 3
                                  Average:                  1
                                  Maximum:                  2
                                  Minimum:                  1
                              Work shifts:                  2


=== LUCCHESSI ===
                              Sum of sales:                14
                                  Average:                  1
                                  Maximum:                  3
                                  Minimum:                  1
                              Work shifts:                  9


*** GRAND TOTAL ***
                              Sum of sales:               155
                                  Average:                  3
                                  Maximum:                 18
                                  Minimum:                  1
                              Work shifts:                 40
```

Processing members of physical and logical files

An example will illustrate how to work with data members of physical and logical files.

Files must be created using DDS (Data Description Specifications) because tables created by SQL statements CREATE TABLE and CREATE VIEW do not enable to add or remove data members.

Creation of physical and logical file with members

Create physical file PRICES.

```
 *******************************************
 *   File PRICES - Item prices
 *******************************************
A                                    UNIQUE
A          R PRICESR
 *   Item number
A            ITEMNBR       5A
 *   Unit price
A            UNITPR        9P 2
 *   Item description
A            DESCR         50A
 *   Key - Item number
A          K ITEMNBR
```

Create logical file PRICESL.

```
 ***********************************************
 *   File PRICESL
 *   Logical file
 ***********************************************
A          R PRICESR              PFILE(PRICES)
A            ITEMNBR
A            UNITPR
A          K ITEMNBR
```

Define maximum number of members in file PRICES.

```
CHGPF      FILE(VZTOOL/PRICES) MAXMBRS(12)
```

Remove the standard member from file PRICES.

```
RMVM       FILE(VZTOOL/PRICES) MBR(PRICES)
```

Add three members to physical file PRICES.

```
ADDPFM     FILE(VZTOOL/PRICES) MBR(PRICES_01)
ADDPFM     FILE(VZTOOL/PRICES) MBR(PRICES_02)
ADDPFM     FILE(VZTOOL/PRICES) MBR(PRICES_03)
```

Define maximum number of members in logical file PRICESL.

```
CHGLF  FILE(PRICESL) MAXMBRS(12)
```

Add three members also to logical file PRICESL.

```
ADDLFM FILE(VZTOOL/PRICESL) MBR(PRICESL_01) DTAMBRS((VZTOOL/PRICES(PRICES_01)))
ADDLFM FILE(VZTOOL/PRICESL) MBR(PRICESL_02) DTAMBRS((VZTOOL/PRICES(PRICES_02)))
ADDLFM FILE(VZTOOL/PRICESL) MBR(PRICESL_03) DTAMBRS((VZTOOL/PRICES(PRICES_03)))
```

Script to create alias objects and insert data in members

```
/* Remove alias objects for the physical and logical file. */
DROP ALIAS VZTOOL/PRICES_01 ;
DROP ALIAS VZTOOL/PRICES_02 ;
DROP ALIAS VZTOOL/PRICES_03 ;
DROP ALIAS VZTOOL.PRICESL_01 ;
DROP ALIAS VZTOOL.PRICESL_02 ;
DROP ALIAS VZTOOL.PRICESL_03 ;

/* Create alias objects for members of physical file PRICES. */
CREATE ALIAS VZTOOL.PRICES_01 FOR VZTOOL.PRICES(PRICES_01) ;
CREATE ALIAS VZTOOL.PRICES_02 FOR VZTOOL.PRICES(PRICES_02) ;
CREATE ALIAS VZTOOL.PRICES_03 FOR VZTOOL.PRICES(PRICES_03) ;

/* Create alias objects for members of logical file PRICESL. */
CREATE ALIAS VZTOOL.PRICESL_01 FOR VZTOOL.PRICESL(PRICESL_01) ;
CREATE ALIAS VZTOOL.PRICESL_02 FOR VZTOOL.PRICESL(PRICESL_02) ;
CREATE ALIAS VZTOOL.PRICESL_03 FOR VZTOOL.PRICESL(PRICESL_03) ;

/* Delet records from physical file PRICES. */
DELETE FROM PRICES_01 ;
DELETE FROM PRICES_02 ;
DELETE FROM PRICES_03 ;

/* Insert records to two members of physical file PRICES. */
INSERT INTO PRICES_01 values ('00001', 8.99, 'Chocolate cakes') ;
INSERT INTO PRICES_01 values ('00002', 459.00, 'Tooth paste Kalodont') ;
INSERT INTO PRICES_01 values ('00003', 1.25, 'Washing line') ;

INSERT INTO PRICES_02 values ('00004', 10.50, 'Men''s socks black') ;
INSERT INTO PRICES_02 values ('00005', 120.00, 'T-shirt white') ;
INSERT INTO PRICES_02 values ('00006', 10.55, 'Men''s socks white, new')
```
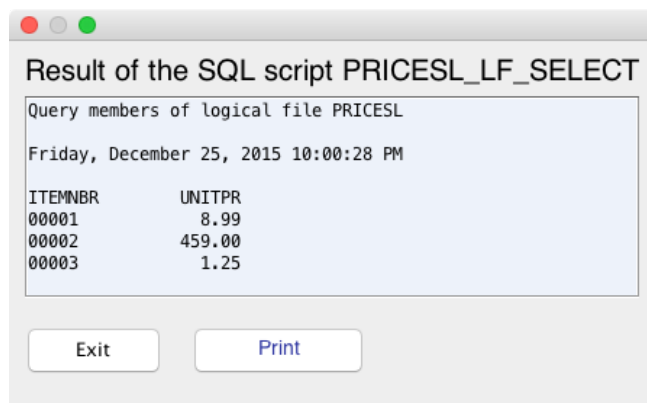
Query data members of the logical file

Queries are specified in the following script to check result of preceding script.

```
--Query members of logical file PRICESL

SELECT * FROM PRICESL_01;
SELECT * FROM PRICESL_02;
SELECT * FROM PRICESL_03;
SELECT * FROM PRICESL
```
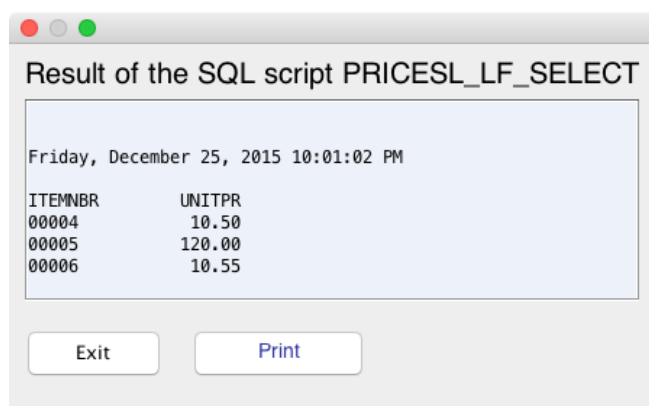
The following windows are shown with contents of the logical members.
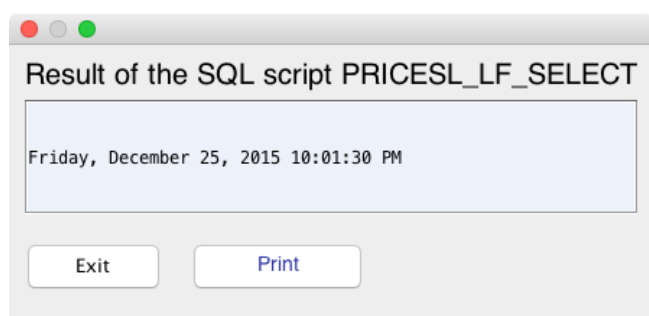
Member PRICESL_01:

Member PRICESL_02:

```
●  ○  ●
Result of the SQL script PRICESL_LF_SELECT

Friday, December 25, 2015 10:01:02 PM

ITEMNBR      UNITPR
00004          10.50
00005         120.00
00006          10.55

    Exit            Print
```

Member PRICESL_03 (empty):

```
●  ○  ●
Result of the SQL script PRICESL_LF_SELECT

Friday, December 25, 2015 10:01:30 PM

    Exit            Print
```

All members together (note that the standard member PRICESL was not removed):

```
●  ○  ●
Result of the SQL script PRICESL_LF_SELECT

Friday, December 25, 2015 10:02:15 PM

ITEMNBR      UNITPR
00001           8.99
00002         459.00
00003           1.25
00004          10.50
00005         120.00
00006          10.55

    Exit            Print
```