

Maintenance of database files for IBM i

User Guide

Contents

Introduction	3
Using the application	4
<i>Objects used in the application</i>	<i>4</i>
<i>Directories</i>	<i>4</i>
<i>Program files</i>	<i>4</i>
<i>Start of the application.....</i>	<i>5</i>
Parameters	6
<i>Application language</i>	<i>6</i>
<i>Server address.....</i>	<i>6</i>
<i>User name.....</i>	<i>6</i>
<i>Window size</i>	<i>6</i>
<i>Mark for null field values.....</i>	<i>6</i>
<i>Size of the font for printing data in print points.....</i>	<i>6</i>
<i>Maximum number of records to display</i>	<i>6</i>
<i>Limit for length of displayed data fields</i>	<i>6</i>
<i>Character set - charset and Select character set for CLOB.....</i>	<i>7</i>
<i>Library with database files.....</i>	<i>8</i>
<i>Select database file.....</i>	<i>8</i>
<i>File member</i>	<i>8</i>
Run.....	9
Selection and ordering of records	10
Insert a new record	12
Changing fields in the record	13
Changing a cell (field in a record)	15
Deletion of a row	16
Selection of columns.....	16
<i>Example</i>	<i>17</i>
Data members of physical and logical files	19
<i>Alias objects</i>	<i>19</i>
Data types CLOB and BLOB	19
<i>Insert a new row</i>	<i>20</i>
<i>Updating a row</i>	<i>21</i>
<i>Updating CLOB column.....</i>	<i>21</i>
<i>Overwriting text</i>	<i>23</i>
<i>Finding text by a pattern.....</i>	<i>23</i>
<i>Saving changed text.....</i>	<i>23</i>
<i>Changing column contents by reading a file</i>	<i>24</i>
<i>Saving column value to a file</i>	<i>25</i>
<i>Updating BLOB column.....</i>	<i>27</i>
<i>Changing column contents by reading a file</i>	<i>27</i>
<i>Problems with big columns.....</i>	<i>30</i>

Introduction

Motivation for this application was the fact that the popular utility DFU (Data File Utility) is unable to display or print all Unicode characters, especially UTF-8, UTF-16 or UCS-2.

The application serves to entering and updating data in database files. It works with physical and logical files. Only one file can be processed at a time.

The physical file must contain at least the first member. For example, a reference file without a data member cannot be displayed.

The logical file that refers to a single physical file can be used to enter and update data.

The file (physical or logical) with multiple members can be processed if ALIAS objects corresponding to the members exist in the library.

Traditional AS/400 terms are mainly used in the application. Correspondence of AS/400 and SQL terms are listed in the following table.

<i>Traditional names</i>	<i>SQL names</i>
library	schema
physical file	table
field	column
record	row
unique key	primary key unique key
key	index
logical file with selection	view
DDM file for a member	ALIAS object

Programs are written in Java language and require version *Java SE 8* or higher. They cooperate with programs from package *IBM i Toolbox for Java* (or *JTOpen*). Programs were created in the OS X operating system. They were tested in systems OS X 10.9 Mavericks, 10.10 Yosemite, 10.11 El Capitan and Windows 7 with remote internet connection to the IBM i, version 7.1.

Using the application

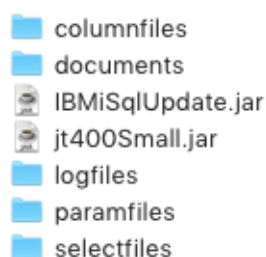
The application is not installed. It is delivered as a directory that can be placed at any location in the personal computer. An alias or shortcut can be created from the file *SqlUpdate.jar* and located somewhere. The application is launched by double click on the shortcut or the original of the file *IBMiSqlUpdate.jar*.

The application works without change in both macOS and Windows systems.

Objects used in the application

Objects contained in the application must be placed in the current directory. This is ensured at installation of the application, that is delivered as a whole directory. When the application is launched the directory becomes current.

Operational directories and program files are inside the application directory:



Directories

- *columnfiles* - contains text files with column lists for the SELECT statement,
- *documents* - contains this document in Czech and English languages,
- *logfiles* - contains text files *err.txt* and *out.txt*, to which redirected output of the files, System.err a System.out (i.e. console),
- *paramfiles* - contains the file *U_Parameters.txt* with application parameters,
- *selectfiles* - contains text files with data for selection and ordering of records in database files.

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

Directories must not be deleted or renamed. The files in directories *selectfiles* and *columnfiles* can be deleted without damage, they are created if needed. Contents of these files should not be changed manually.

Program files

- File *jt400Small.jar* - contains a subset of classes from the package IBM i Toolbox for Java.
- File *IBMiSqlUpdate.jar* - contains Java classes of the application and launches the application.

Start of the application

After starting the application the window *Set application parameters and run* is displayed. The dialog for signon parameters to access to the IBM i system is displayed at the same time. After signing on the user can adjust parameter values using the *Save data* button or *Enter* key (or leave the values unchangend), and then press the *Run* button.

Signon to the System

System: 193.179.195.133

User ID: VZUPKA

Password:

☒ Default User ID

☒ Save password

OK Cancel

Set application parameters and run

English ☒ Application language. Restart the application after change.

Česky ☐ Jazyk aplikace. Po změně spusťte aplikaci znovu.

193.179.195.133 Server address

VZUPKA User name

☒ Automatic window size

950 Window width

890 Window height

Mark for null field values

12 Size of the font for printing data in print points

1000 Maximum number of records to display

1000 Limit for length of displayed data field

UTF-8 Character set for CLOB

UTF-8 Select character set for CLOB

VZTOOL Library with database files

CENY2 Select database file

*FIRST File member

Save data or press ENTER Run

Current directory is: /Users/vzupka/Documents/eclipseJK/workspace/SqlUpdate_02

Parameters

Note: If the user enters some important parameters incorrectly, then after clicking the *Run* button the following message appears

SQL STATEMENT ERROR or CONNECTION TO THE SERVER LOST.

Besides signon entries, the most sensitive entry is the *library name*.

Application language

The application can be processed in English (en_US) or Czech (cs_CZ) localization. The localization concerns titles, messages, button labels. The user can choose the localization by clicking the button. The selected option is applied fully (including the application menu) after ending the application and launching it again.

Server address

The user enters a single IP address in dot or domain form.

User name

The user enters the name of the profile that has the authority to write and change data in database files. This name will be prescribed in the dialog window *Signon to the System*.

Window size

If the box *Automatic window size* is checked the window is accommodated to the size of the displayed results. Otherwise the window will have dimensions specified in input fields *Window width* and *Window height*.

Note: If the value is not a whole number, zero is assumed.

Mark for null field values

The user enters a symbol that will be shown everywhere the field value is NULL. This symbol is also used for entering the NULL value in the table cell or input field. It is *not used* for fields of types CLOB and BLOB.

Size of the font for printing data in print points

This entry represents the number of points on the display. It determines size of letters in titles and cells in table cells and also in input data fields.

Note: If the value is not a whole number, zero is assumed.

Maximum number of records to display

This entry represents maximum number of records that are selected from the database file and that will be shown in the window table.

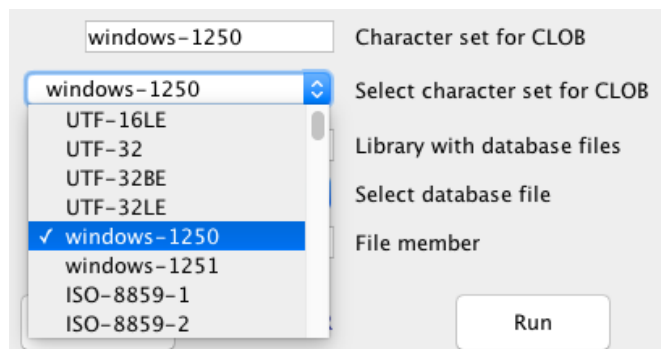
Limit for length of displayed data fields

This entry represents maximum length of the display field containing the data field value.

Character set - charset and Select character set for CLOB

The user defines the character set for processing columns that have type of CLOB (Character Large Objects). The character set concerns the *text file* from which data is transferred to the column or the text file to which data is transferred from the column. The user can enter the symbol of the character set to the input field manually or use the button *Select character set for CLOB* for CLOB.

Uživatel stiskne tlačítko a z nabídky vybere symbol pro znakovou sadu. Předvolená znaková sada je UTF-8. V seznamu jsou zařazeny všechny podporované kódy.



There are many character sets supported in Java, all are listed in the menu of the button (see below). The most frequented charsets are at the beginning of the list:

UTF-8	Eight-bit Unicode (or UCS) Transformation Format
US-ASCII	American Standard Code for Information Interchange
UTF-16	Sixteen-bit Unicode (or UCS) Transformation Format, byte order identified by an optional byte-order mark
UTF-16BE	Sixteen-bit Unicode (or UCS) Transformation Format, big-endian byte order
UTF-16LE	32-bit Unicode (or UCS) Transformation Format, little-endian byte order
UTF-32	32-bit Unicode (or UCS) Transformation Format, byte order identified by an optional byte-order mark
UTF-32BE	32-bit Unicode (or UCS) Transformation Format, big-endian byte order
UTF-32LE	32-bit Unicode (or UCS) Transformation Format, little-endian byte order
windows-1250	Windows Eastern European (Latin 2)
windows-1252	Windows Latin-1
ISO-8859-1	ISO-8859-1, Latin Alphabet No. 1
ISO-8859-2	Latin Alphabet No. 2

Note: List of the charsets supported in Java can be found in the documentation on the address <http://docs.oracle.com/javase/8/docs/technotes/guides/intl/encoding.doc.html>. List of all charsets is in the publications *IANA Charset Registry* on the address <http://www.iana.org/assignments/character-sets/character-sets.xhtml>.

The proper *contents of the CLOB column* as stored in the database has the character set corresponding to the column type specified when the file (table) was created. The column type can be CLOB (Character Large Object), NCLOB (National Character Large Object) or DBCLOB (Double Byte Character Large Object).

- For type **CLOB** the column has one of the codes for SBCS (Single Byte Character Set - SBCS). The most general of them is **UTF-8** (CCSID 1208). The others are designated for individual areas, e. g.

IBM37 (CCSID 37), **IBM870** (CCSID 870), etc., from EBCDIC type, **windows-1250** (CCSID 1250, Windows Latin 2), **ISO-8859-2** (CCSID 912, ISO Latin 2), etc. from ASCII type.

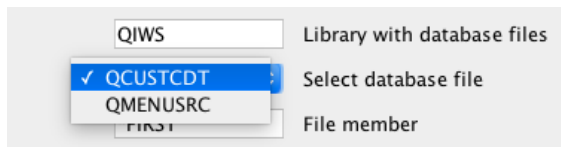
- For type **NCLOB** the column has character set UTF-16 (CCSID 1200).
- For type **DBCLOB** the column has character set UTF-16 (CCSID 1200).

Library with database files

The name convention “*system*” is used to process SQL statements. The user enters a library name containing database files to work with of which one file is subsequently chosen.

Select database file

The user presses the button and selects a name from the name list, e. g. QCUSTCDT.



The selected name is then shown on the button and in the input fields *Database file* and input field *File member*.

File member

The user may overwrite the default value, usually *FIRST, by a member name of the database file. See the chapter [Data members of physical and logical files](#) below.

Run

The button *Run* invokes the window with a table containing maximum number of the first records selected from the file specified in application parameters (see *Parameters* above).

File QCUSTCDT in library QWIS was chosen in this example.

Table QIWS/QCUSTCDT

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	CUSNUM	LSTNAM	...	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CDTDUE
1	938472	Henning	G K	4859 Elm Ave	Dallas	TX	75217	5000	3	37.00	0.00
2	839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
3	392859	Vine	S S	P0 Box 79	Broton	VT	5046	700	1	439.00	0.00
4	938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
5	397267	Tyron	W E	13 Myrtle Dr	Hector	NY	14841	1000	1	0.00	0.00
6	389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	44.50
7	846283	Alison	J S	787 Lake Dr	Isle	MN	56342	5000	3	10.00	0.00
8	475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
9	693829	Thomas	A N	3 Dove Circle	Casper	WY	82609	9999	2	0.00	0.00
10	593029	Williams	E D	485 SE 2 Ave	Dallas	TX	75218	200	1	25.00	0.00
11	192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50
12	583990	Abraham	M T	392 Mill St	Isle	MN	56342	9999	3	500.00	0.00

Enter condition WHERE for row selection and press Refresh.

Enter condition ORDER BY for row ordering and press Refresh.

select rrn(QCUSTCDT) as RRN, CUSNUM, LSTNAM, INIT, STREET, CITY, STATE, ZIPCOD, CDTLMT, CHGCOD, BALDUE, CDTDUE
from QIWS/QCUSTCDT
fetch first 1000 rows only

ExitInsert new rowEdit selectedRefreshDelete selectedSelect columns

The complete **SELECT** statement is shown under two text fields (above the button row). The list of columns and maximum number of records is visible from the statement. The expression *rrn(QCUSTCDT) as RRN* represents relative number of the record in the file. It is displayed for information in the first column of the table in blue color and is used for updating or deleting of the record. This value cannot be changed.

Selection and ordering of records

The first display of the file is a result of the **SELECT** statement without using clauses **WHERE** or **ORDER BY**, so the first unordered records are selected in the maximum number specified in Parameters. More accurately, the records are ordered by the relative record number named as **RRN**. This order corresponds to the arrival sequence in which the records were written to the file.

To display records of interest input fields under the table of records can be used.

- Expression as in the **WHERE** clause or the **SELECT** statement is entered in the first field.
- Expression as in the **ORDER BY** clause of the **SELECT** statement is entered in the second field.

After entering an expression in one or both fields, the user presses the *Refresh* button and the program performs new selection and/or ordering of records.

For example, if the entry in the first field is **BALDUE >= 30 and CHGCOD < 3** and the entry in the second field is **BALDUE DESC**, the following window is displayed with selected and ordered records and the modified **SELECT** statement.

Table QIWS/QCUSTCDT

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	CUSNUM	LSTNAM	...	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CDTDUE
4	938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
11	192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50
3	392859	Vine	S S	PO Box 79	Broton	VT	5046	700	1	439.00	0.00
8	475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
2	839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
6	389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	44.50

Enter condition WHERE for row selection and press Refresh.

BALDUE >= 30 and CHGCOD < 3

Enter condition ORDER BY for row ordering and press Refresh.

BALDUE DESC

select rrn(QCUSTCDT) as RRN, CUSNUM, LSTNAM, INIT, STREET, CITY, STATE, ZIPCOD, CDTLMT, CHGCOD, BALDUE, CDTDUE
from QIWS/QCUSTCDT
WHERE BALDUE >= 30 and CHGCOD < 3
ORDER BY BALDUE DESC
fetch first 1000 rows only

ExitInsert new rowEdit selectedRefreshDelete selectedSelect columns

The expressions in input fields are stored in a text file named after the library and the file with ending .sel. This file is stored in the directory *selectfiles*.

Contents of the text file has the following form.

```
selection;  
ordering
```

where *selection* is either empty text or the expression for WHERE clause and *ordering* is either empty text or the expression for ORDER BY clause. The two parts are separated by semicolon.

The file from the example is named **QIWS-QCUSTCDT.sel** and contains

```
BALDUE >= 30 and CHGCOD < 3;  
BALDUE desc
```

Note 1: As an entry in both fields, the expression **RRN(QCUSTCDT)** denoting the relative record number can be used.

Note 2: To select a binary field (type BINARY, VARBINARY), function HEX() must be used because the value must be written as couples of hexadecimal characters. For example, if the field BIN01 of the BINARY type the expression for WHERE is entered as follows.

```
hex(BIN01) like '%cd%'
```

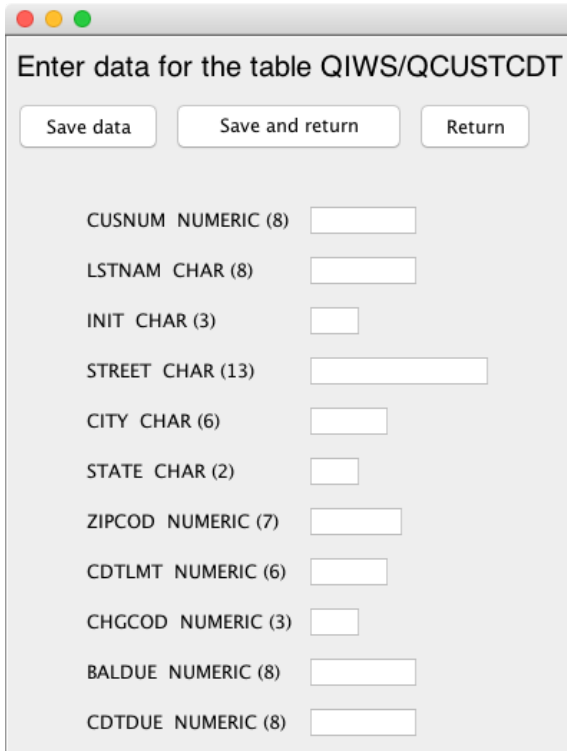
or

```
hex(BIN01) >= 1A
```

and the like.

Insert a new record

To insert a new record press the button *Insert a new row*. A list of field (column) names with empty input fields is displayed in a new window.



Enter data for the table QIWS/QCUSTCDT

Save data Save and return Return

CUSNUM NUMERIC (8)

LSTNAM CHAR (8)

INIT CHAR (3)

STREET CHAR (13)

CITY CHAR (6)

STATE CHAR (2)

ZIPCOD NUMERIC (7)

CDTLMT NUMERIC (6)

CHGCOD NUMERIC (3)

BALDUE NUMERIC (8)

CTDUE NUMERIC (8)

The user enters values in the input fields according to the field type as defined in the file for the record field/column. For character type (CHAR, VARCHAR, ...) empty value may remain.

- Values of the character type CHAR, VARCHAR, ... must correspond to the character set defined in the file for the field/column.
for CCSID 1208 UTF-8,
for CCSID 1200 UTF-16,
for CCSID 13488 UCS-2 (Unicode 2.0, UTF-16 BE with IBM PUA),
for CCSID 37 English,
for CCSID 870 Czech,
etc.
- Values of the types DEC, DECIMAL, NUMERIC may contain only digits, decimal point, and the sign before the number.
- Values of the types INT, INTEGER, BIGINT may contain only digits and the sign before the number.
- Values of the type DATE must be entered in the format ISO, thus YYYY-MM-DD, e.g. 2014-02-15.
- Values of the type TIME must be entered in the format ISO, thus HH:MM:SS, e.g. 19:31:05.
- Values of the type TIMESTAMP must be entered in the format ISO, thus YYYY-MM-DD HH:MM:SS.MMMMMM, e.g. 2000-04-05 23:59:59.999999.
- Values of types BINARY or VARBINARY are entered in *couples of hexadecimal characters*. If the user enters an invalid character, zero is assumed. The input fields are therefore twice as long.
- NULL value is entered how defined in *Parameters*, e.g. *null*.

If the user presses button *Save* or *ENTER* key, the new record is written to the file and all the values in the fields are retained. The user can rewrite the values and press the button *Save*. This method can be used for entering larger number of records.

If the user presses button *Save and return*, the new record is written to the file and the list of records in the preceding window is displayed.

If the user presses button *Return*, the list of records in the preceding window is displayed. The same occurs if the user closes the window.

Note: If no table row was selected, the fields will be empty. If, however, a row was selected, then the fields will contain values from the row.

Changing fields in the record

The user selects a row (record) from the list in the table and presses button *Edit selected*.

Table QIWS/QCUSTCDT

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	CUSNUM	LSTNAM	...	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CTDUE
4	938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
11	192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50
3	392859	Vine	S S	P0 Box 79	Broton	VT	5046	700	1	439.00	0.00
8	475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
2	839283	Jones	B D	218 NW 135 St	Clay	NY	13041	400	1	100.00	0.00
6	389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	44.50

Enter condition WHERE for row selection and press Refresh.

BALDUE >= 30 and CHGCOD < 3

Enter condition ORDER BY for row ordering and press Refresh.

BALDUE DESC

```
select rrn(QCUSTCDT) as RRN, CUSNUM, LSTNAM, INIT, STREET, CITY, STATE, ZIPCOD, CDTLMT, CHGCOD, BALDUE, CDTDUE
from QIWS/QCUSTCDT
WHERE BALDUE >= 30 and CHGCOD < 3
ORDER BY BALDUE DESC
fetch first 1000 rows only
```

Exit Insert new row Edit selected Refresh Delete selected Select columns

The same window is displayed as before but with the current values of the fields. The user can rewrite the values according to their types (see above).

Enter data for the table QIWS/QCUSTCDT

Save data Save and return Return

CUSNUM	NUMERIC (8)	389572
LSTNAM	CHAR (8)	Stevens
INIT	CHAR (3)	K L
STREET	CHAR (13)	208 Snow Pass
CITY	CHAR (6)	Denver
STATE	CHAR (2)	CO
ZIPCOD	NUMERIC (7)	80226
CDTLMT	NUMERIC (6)	400
CHGCOD	NUMERIC (3)	1
BALDUE	NUMERIC (8)	58.75
CDTDUE	NUMERIC (8)	1.50

If the user presses button *Save* or *ENTER* key, the new record is updated in the file and all the values in the fields are retained. This method can be used for entering larger number of records.

If the user presses button *Save and return*, the new record is updated in the file and the list of records in the preceding window is displayed.

If the user presses button *Return*, the list of records in the preceding window is displayed.

Changing a cell (field in a record)

A field can be updated directly in the list of records using double click. The cell is then enabled for editing (rewriting).

Table QIWS/QCUSTCDT

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	CUSNUM	LSTNAM	...	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CDTDUE
4	938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
11	192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50
3	392859	Vine	S S	P0 Box 79	Broton	VT	5046	700	1	439.00	0.00
8	475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
2	839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
6	389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	1.00

After the user writes the new value (or leaves the old one) and presses the *ENTER* key, the value is stored back in the cell and the field is rewritten in the record at the same time.

Table QIWS/QCUSTCDT

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	CUSNUM	LSTNAM	...	STREET	CITY	STATE	ZIPCOD	CDTLMT	CHGCOD	BALDUE	CDTDUE
4	938485	Johnson	J A	3 Alpine Way	Helen	GA	30545	9999	2	3987.50	33.50
11	192837	Lee	F L	5963 Oak St	Hector	NY	14841	700	2	489.50	0.50
3	392859	Vine	S S	P0 Box 79	Broton	VT	5046	700	1	439.00	0.00
8	475938	Doe	J W	59 Archer Rd	Sutter	CA	95685	700	2	250.00	100.00
2	839283	Jones	B D	21B NW 135 St	Clay	NY	13041	400	1	100.00	0.00
6	389572	Stevens	K L	208 Snow Pass	Denver	CO	80226	400	1	58.75	1.00

The user can press TAB key or click (once or twice) on any other cell.

Note: Values of types BINARY or VARBINARY are entered in *couples of hexadecimal characters*. If the user enters an invalid character, zero is assumed. The input fields are therefore twice as long.

Deletion of a row

If the user selects a row (record) from the list and presses the button *Delete selected* the selected record is removed from the list and deleted from the file.

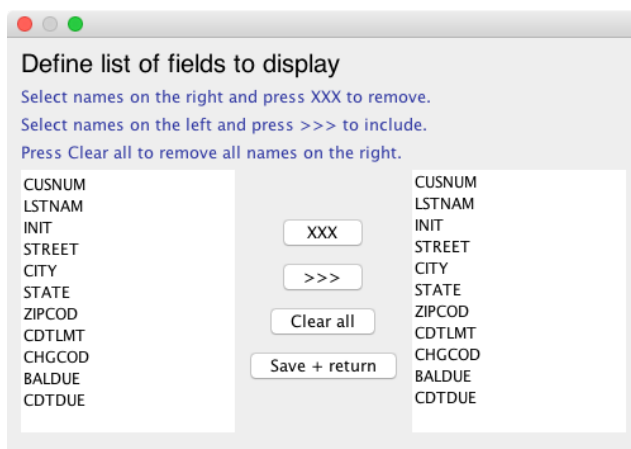
Important: The deleted record cannot be returned back!

Note: The number RRN of the deleted record is also removed, the record is not accessible any more but it occupies its place in the data space. The data space can be compressed and the numbering reordered by CL command RGZPFM (Reorganize Physical File Mbr) or by copying to a work file and back by CL command CPYF.

Selection of columns

This function enables reducing and ordering columns in the list of records.

If the user presses the button *Select columns*, a window appears with two frames and buttons between.



The left frame contains always the complete list of fields/columns of the database file. The right frame contains the same list at the beginning. The user can vary the list in the right frame using the buttons.

- Button *XXX* removes selected names from the right frame.
- Button *>>>* copies selected names from the left frame to the right frame on the last position.
- Button *Clear all* clears the whole contents of the right frame.
- Button *Save + return* saves the list of field names from the right frame to a text file and shows the list with the selected and ordered columns.

The user can select one or more names from the frames. A single name is selected by clicking on it. A group of names is selected by holding *Shift* key and pulling the mouse. More than one individual names is selected by holding *Ctrl* key (not in OS X).

All functions can be performed as well as in the original record display.

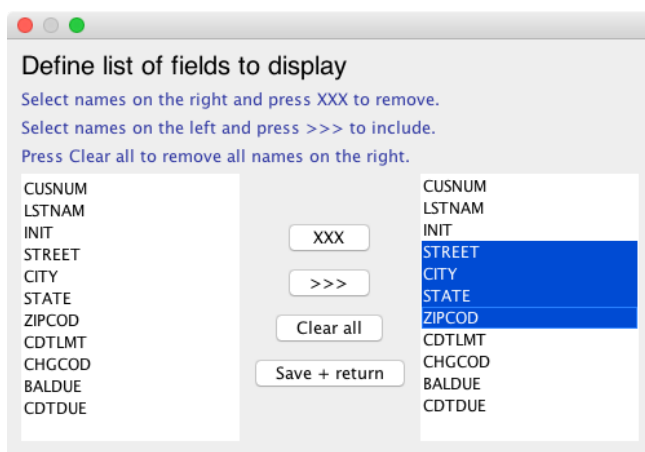
The resulting list of selected fields is amended by separating commas and is stored in a text file named by the library and file name with ending *.col*.

If the window with the list of records is closed, the text file will contain the original field list again. The list of selected fields is valid only during the work with the database file.

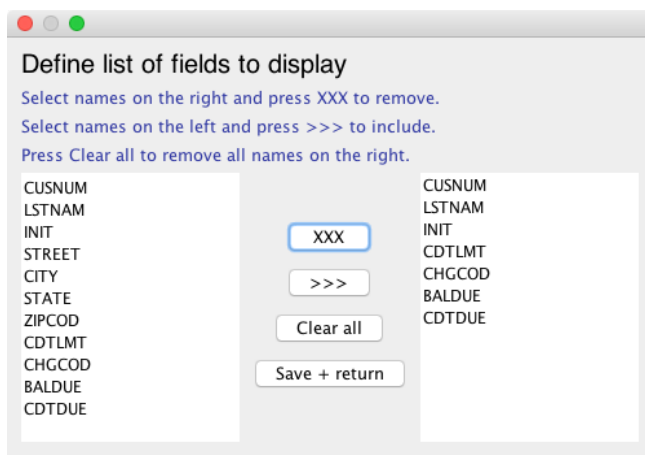
Text files *.col* may be removed from the directory *columnfiles* without any damage.

Example

After pressing the button *Select columns* the following window is displayed. We select field
Select fields STREET, CITY, STATE, ZIPCOD in the right frame.



Press button XXX, the right field list is shortened.



Press button *Save + return* the resulting list is stored in the text file and the list of records is displayed with two columns.

Table QIWS/QCUSTCDT

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	CUSNUM	LSTNAM	...	CDTLMT	CHGCOD	BALDUE	CDTDUE
4	938485	Johnson	J A	9999	2	3987.50	33.50
11	192837	Lee	F L	700	2	489.50	0.50
3	392859	Vine	S S	700	1	439.00	0.00
8	475938	Doe	J W	700	2	250.00	100.00
2	839283	Jones	B D	400	1	100.00	0.00
6	389572	Stevens	K L	400	1	58.75	1.00

Enter condition WHERE for row selection and press Refresh.

BALDUE >= 30 and CHGCOD < 3

Enter condition ORDER BY for row ordering and press Refresh.

BALDUE DESC

```
select rrn(QCUSTCDT) as RRN, CUSNUM, LSTNAM, INIT, CDTLMT, CHGCOD, BALDUE, CDTDUE
from QIWS/QCUSTCDT
WHERE BALDUE >= 30 and CHGCOD < 3
ORDER BY BALDUE DESC
fetch first 1000 rows only
```

Exit

Insert new row

Edit selected

Refresh

Delete selected

Select columns

The fiels list is stored to the text file named QIWS-QCUSTCDT.col and has the following form.

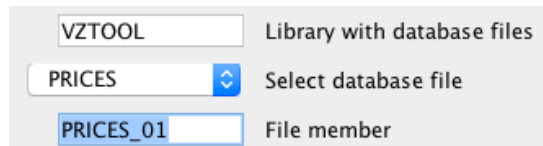
, CUSNUM, LSTNAM, INIT, CDTLMT, CHGCOD, BALDUE, CDTDUE

Note: The first comma separates the list from the field/column *rrn(QCUSTCDT)* as *RRN*, which is always displayed.

Data members of physical and logical files

The application can process data members of physical and logical files as if there were files. An alias object must be created in the library for the member whose name differs from the default of the file (*FIRST or file name). The *application* creates the alias object *itself* if it does not exist.

For example, the user selects the file VZTOOL/PRICES in *Parameters* and changes the input field *File member* to PRICES_01. If the member exists, the application automatically creates a new alias object named PRICES_01 and displays the list of records contained in the member. As soon as the application exits the data window the corresponding alias is deleted.



VZTOOL	Library with database files
PRICES	Select database file
PRICES_01	File member

Alias objects

The alias objects are physical files (object type *FILE) with attribute DDMF. DDMF (Distributed Data Management File) is an object serving in access from a Local Location to a Remote Location in SNA, APPC architecture. Here, both locations are the same and the alias object serves as a mediator of access to the data file.

The user can also create alias objects using SQL statements using SQL statement CREATE ALIAS. A different means than this application must be used to do it, e.g. STRSQL CL command. For example, if the physical file PRICES has members PRICES_01, PRICES_02, the user creates alias objects using the following SQL statements.

```
CREATE ALIAS PRICES_01 FOR VZTOOL.PRICES(PRICES_01)
CREATE ALIAS PRICES_02 FOR VZTOOL.PRICES(PRICES_02)
```

Note: The alias names need not be the same as member names but it is practical.

The objects in the library look like this:

PRICES	*FILE	PF-DTA
PRICES_01	*FILE	DDMF
PRICES_02	*FILE	DDMF

The same method can be used for logical file members. If the logical file is named PRICESL and has members PRICESL_01, PRICESL_02, the user creates alias objects using the following SQL statements.

```
CREATE ALIAS VZTOOL.PRICESL_01 FOR VZTOOL.PRICESL(PRICESL_01)
CREATE ALIAS VZTOOL.PRICESL_02 FOR VZTOOL.PRICESL(PRICESL_02)
```

Data types CLOB and BLOB

In this chapter we use terms row and column instead of record and field, because we deal with SQL objects rather than traditional data fields.

These data types large objects - LOB containing data of so called advanced types CLOB (Character Large Object) and BLOB (Binary Large Object).

Variants of the type CLOB exist in IBM i:

- NCLOB (National Character Large Object). The column has character set UTF-16 (CCSID 1200).
- DBCLOB (Double Byte Character Large Object). The column has character set UTF-16 (CCSID 1200).

Columns of these types are not displayed in the list of rows in the table because the amount of data is too large or not able to be displayed in a table cell. Data is displayed only at updating in a separate window when the column already contains some value that can be displayed. This means a text in case of CLOB or a bit map of certain type (e.g. a photo) in case of BLOB.

Columns can obtain a value either at insertion of a new row or at updating of an existing row.

We will work with a table BLOBCLOB

V dalším výkladu budeme pracovat s tabulkou BLOBCLOB created by the following statement.

```
CREATE TABLE VZTOOL.BLOBCLOB (
  COL0 DECIMAL (7, 0),
  CLOB1 CLOB (1000),
  COL2 CHAR (5),
  CLOB2 CLOB (500),
  BLOB1 BLOB (10000000),
  BLOB2 BLOB
)
```

Note the sizes in parentheses at the columns of CLOB and BLOB types.

Columns CLOB1 and CLOB2 are relatively short with capacities 1000 and 500 characters respectively. We can observe filling them with smaller or larger files and behavior of the application when the capacity is exceeded.

Columns BLOB1 and BLOB2 are long with capacities 10 million bytes and no size (1 MB is the default value) respectively. They can absorb contents of large files.

Insert a new row

Only columns of normal types are displayed in the rows.

Table VZTOOL/BLOBCLOB

Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).

RRN	COL0	COL2
-----	------	------

Exit
Insert new row
Edit selected
Refresh
Delete selected
Select columns

After pressing the button *Insert new row* a window with the list of columns is displayed.

Enter data for the table VZTOOL/BLOBCLOB

Save data
Save and return
Return

COL0 DECIMAL (9)
COL2 CHAR (5)
CLOB1 CLOB (1000)
CLOB2 CLOB (500)
BLOB1 BLOB (10000000)
BLOB2 BLOB (1048576)

Normal columns are placed at the beginning even if they were not defined in this sequence. Then CLOB type columns follow and BLOB types at the end. Numbers in parentheses denote capacity - maximum size of the contents. For CLOB it is the number of characters, for BLOB it is the number of bytes.

The user enters data in columns COL1 and COL2 (e.g. number 1 and character A) and presses the button *Save and return*. Data is stored to the row where the large columns have now NULL value and the list of rows is displayed showing the data entered.

Table VZTOOL/BLOBCLOB		
Change a cell value: Double click, rewrite the cell value and press ENTER (or click TAB or another cell).		
RRN	COL0	COL2
1	1	A

Updating a row

The user selects the row in the table and presses the button *Edit selected*. The window for updating the columns is displayed.

Enter data for the table VZTOOL/BLOBCLOB

Save data

Save and return

Return

COL0 DECIMAL (9)

1

COL2 CHAR (5)

a

COLB1 CLOB (1000)

CLOB1

COLB2 CLOB (500)

CLOB2

BLOB1 BLOB (10000000)

BLOB1

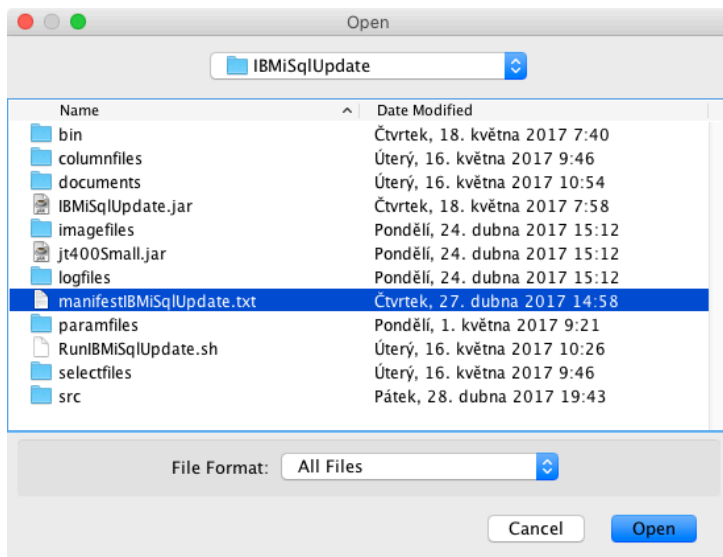
BLOB2 BLOB (1048576)

BLOB2

Normal columns can be changed as usual. Large object columns can be changed using *buttons* with their names (instead of input fields).

Updating CLOB column

First, we assume that the contents of the column is NULL (which is when ne row is first inserted). Press button CLOB2 and a menu for file selection is displayed.



If we select an input file, e.g. manifestIBMiSqlUpdateMenu.text, a message with the size of the file is displayed but the text of the column is not for now.

Enter data for the table VZTOOL/BLOBCLOB

Save data Save and return Return

Column length is 80.

COL0 DECIMAL (9)	<input type="text" value="1"/>
COL2 CHAR (5)	<input type="text" value="a"/>
CLOB1 CLOB (1000)	<input type="button" value="CLOB1"/>
CLOB2 CLOB (500)	<input type="button" value="CLOB2"/>
BLOB1 BLOB (10000000)	<input type="button" value="BLOB1"/>
BLOB2 BLOB (1048576)	<input type="button" value="BLOB2"/>

To see the column contents we *press the button CLOB2 again*. Then the following window with the text of the column is displayed.

Column CLOB2 - UTF-8

Start of text: Length of text: Find text:

```
Manifest-Version: 1.0
Class-Path: ./jt400Small.jar
Main-Class: update/U_Menu
```

Return Refresh Save data Get File Put File Page Setting Print

Column length is 77.

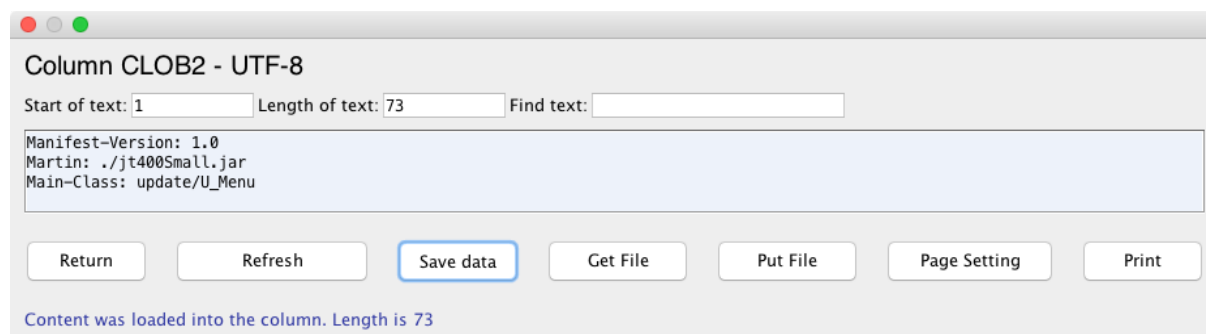
This example shows the case when the input file is shorter (77) than the capacity of the column (500). Data in input fields are set to initial values:

- *Start of text* is 1.
- *Length of text* is 77, i.e. length of all text contained in the column CLOB2.
- *Find text* is empty.

Overwriting text

The text in the window can be overwritten or added and saved using the button *Save data*.

For example, we change *Class-Path* to *Martin* and press *Save data*. Then we press *Refresh* button and get the following result:



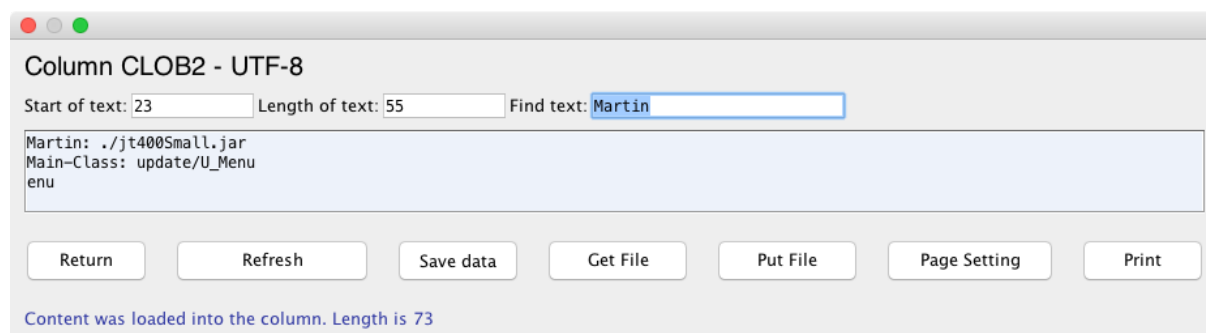
The screenshot shows a window titled "Column CLOB2 - UTF-8". At the top, there are three input fields: "Start of text:" with the value "1", "Length of text:" with the value "73", and "Find text:" which is empty. Below these is a text area containing the following text:
Manifest-Version: 1.0
Martin: ./jt400Small.jar
Main-Class: update/U_Menu
Below the text area is a row of buttons: "Return", "Refresh", "Save data" (highlighted with a blue border), "Get File", "Put File", "Page Setting", and "Print". At the bottom of the window, a status bar displays the message: "Content was loaded into the column. Length is 73".

Text length changed to 73 because text *Martin* is 4 characters shorter than the rewritten text *Class-path*. The overall length of the column remains unchanged. The whole text of the column was overwritten only by the displayed text area of length 73. The last four characters are hidden.

Finding text by a pattern

If we enter *Martin* (pattern to be found) to the field *Find text*, and press *Refresh* button or *ENTER* key, the text is searched for the pattern from the position 1 in *Start of text*.

If the pattern is found, the text from the beginning of found data is displayed. Position 23 of the found data is written to *Start of field*. Length 55 is written to *Text length*. It is the length of *displayed* text.



The screenshot shows the same window "Column CLOB2 - UTF-8". The "Start of text:" field now contains "23", and the "Length of text:" field contains "55". The "Find text:" field contains "Martin". The text area below now displays:
Martin: ./jt400Small.jar
Main-Class: update/U_Menu
enu
The buttons and status bar remain the same as in the previous screenshot.

The text *enu* (including by a new-line character before) at the end is a remainder of the original text. Text *Martin* is 4 characters shorter than the replaced text *Class-path*. The resulting text was shifted to the left. Only the text from the position 23 is displayed but the overall length 77 remains unchanged ($22 + 55 = 77$).

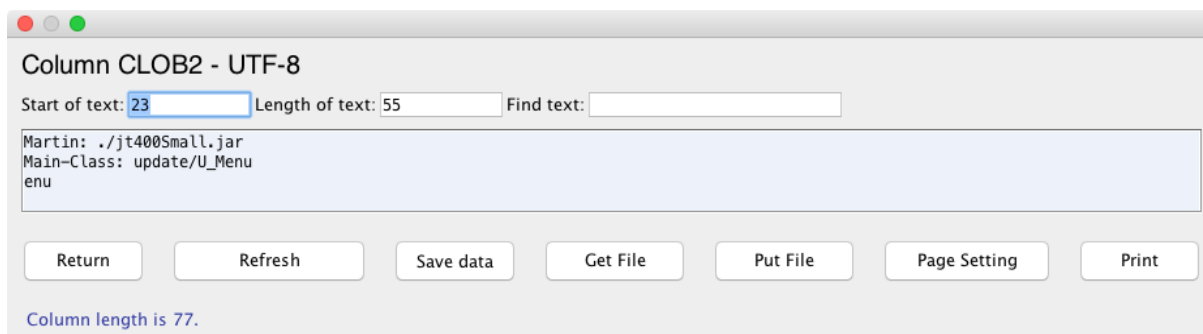
Note: Positions are counted from 1.

Saving changed text

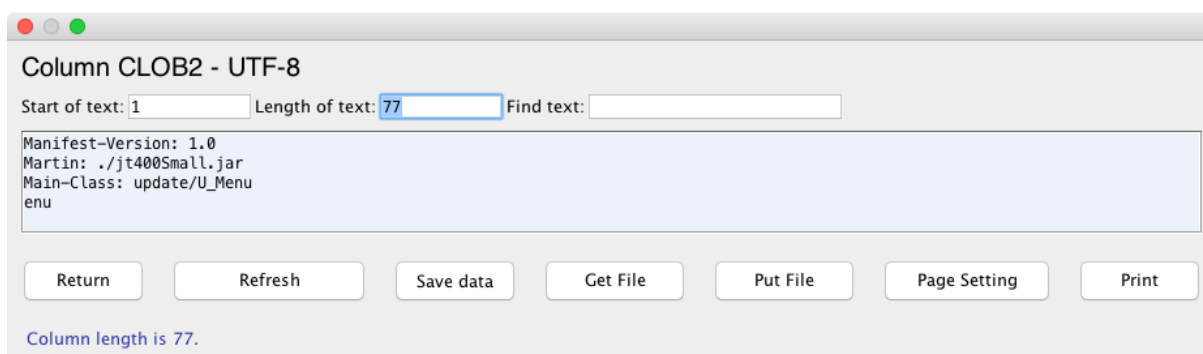
Pressing button *Save data* overwrites the column contents by the text just displayed in the window from the *Start position* in the *Length of text*.

Pressing button *Return* also overwrites column contents like the button *Save data*. Moreover the window with the list of columns is then displayed.

If we display the column by the button CLOB2 again, the same picture is displayed but with empty field *Find text*.



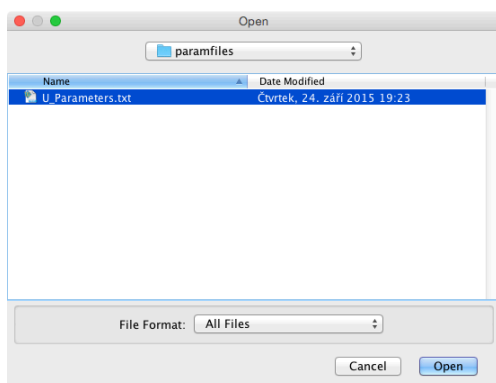
If we change the position to 1 and the length to a greater value (e.g. 100) and press *Refresh* button or *ENTER* key, the whole text in length 77 is displayed.



Changing column contents by reading a file

The button *Get file* allows to overwrite the whole contents of the column by contents of a selected text file.

For example, we decide that we wish to see the new text from position 1. So we enter 1 to the field *Start of text* and press the button *Get file*. A window with the list of files is displayed from which we select e.g. the file *U_Parameters.txt*.



After pressing the button *Open* the text taken from the selected file is displayed in the window.

Column CLOB2 - UTF-8

Start of text: Length of text: Find text:

```
#Table Update for IBM i, © Vladimír \u017Dupka 2015
#Mon Jan 04 14:48:58 CET 2016
RESULT_WINDOW_WIDTH=950
USER_NAME=VZUPKA
FILE=BLOBCLOB
LIBRARY=VZTOOL
MEMBER=BLOBCLOB
AUTO_WINDOW_SIZE=Y
RESULT_WINDOW_HEIGHT=850
LANGUAGE=en-US
FETCH_FIRST=1000
CHARSET=UTF-8
FONT_SIZE=12
HOST=193.179.195.140
NULL_MARK=null
PRINT_FONT_SIZE=9
```

Return Refresh Save data Get File Put File Page Setting Print

Column length is 325

If a file whose contents is larger than column capacity a message is displayed and the column contents is not changed.

Column CLOB2 - UTF-8

Start of text: Length of text: Find text:

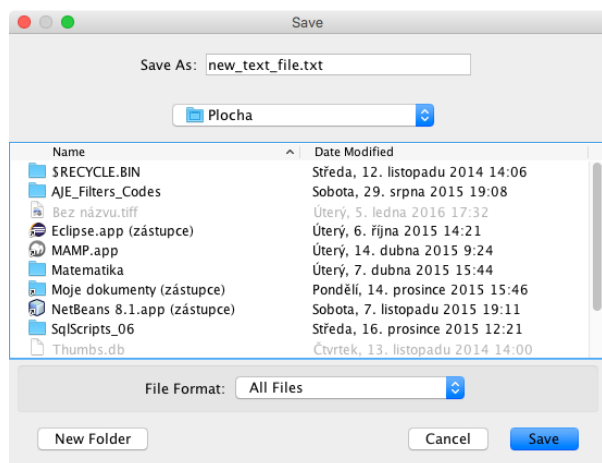
```
#Table Update for IBM i, Vladimír \u017Dupka 2015
#Mon Jan 04 14:48:58 CET 2016
RESULT_WINDOW_WIDTH=950
USER_NAME=VZUPKA
FILE=BLOBCLOB
LIBRARY=VZTOOL
MEMBER=BLOBCLOB
AUTO_WINDOW_SIZE=Y
RESULT_WINDOW_HEIGHT=850
LANGUAGE=en-US
FETCH_FIRST=1000
CHARSET=UTF-8
FONT_SIZE=12
HOST=193.179.195.140
NULL_MARK=null
PRINT_FONT_SIZE=9
```

Return Refresh Save data Get File Put File Page Setting Print

Column capacity was exceeded. File length is 556.

Saving column value to a file

The button *Put file* invokes a window where we enter the name of a file to the input field *Save As:* and select a directory. Then after pressing the *Save* button the file is created in the directory.



Nový soubor bude uložen ve *znakové sadě* zadané v *Parametrech*.

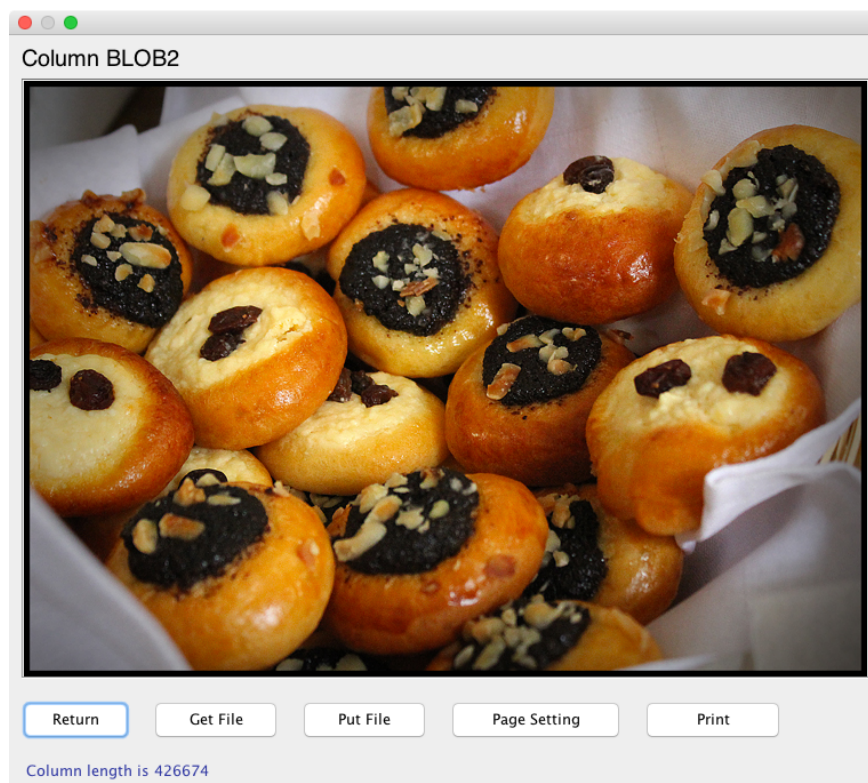
Updating BLOB column

Enter data for the table VZTOOL/BLOBCLOB

Content of the file or part of it was loaded into the column. Length is 426674

COL0 DECIMAL (9)	<input type="text" value="1"/>
COL2 CHAR (5)	<input type="text" value="a"/>
CLOB1 CLOB (1000)	<input type="button" value="CLOB1"/>
CLOB2 CLOB (500)	<input type="button" value="CLOB2"/>
BLOB1 BLOB (10000000)	<input type="button" value="BLOB1"/>
BLOB2 BLOB (1048576)	<input type="button" value="BLOB2"/>

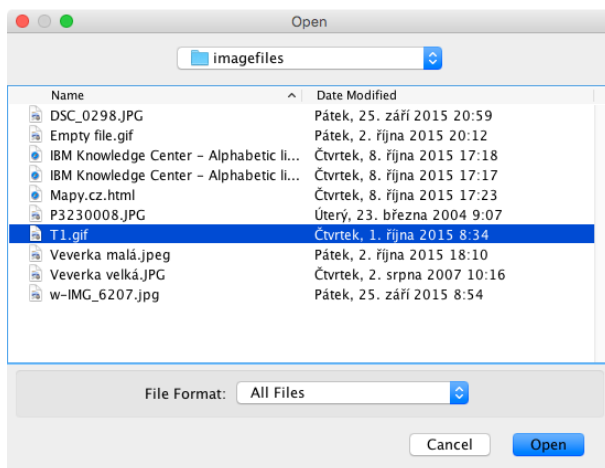
Press, say, the button *BLOB2*. We assume that the column has already a non-null contents, e.g. an image of file type JPG (a photo). Then the image is displayed in a window.



Note: Bitmap files of types JPEG, JPG, GIF, PNG, BMP, WBMP can be displayed as a picture, other types cannot.

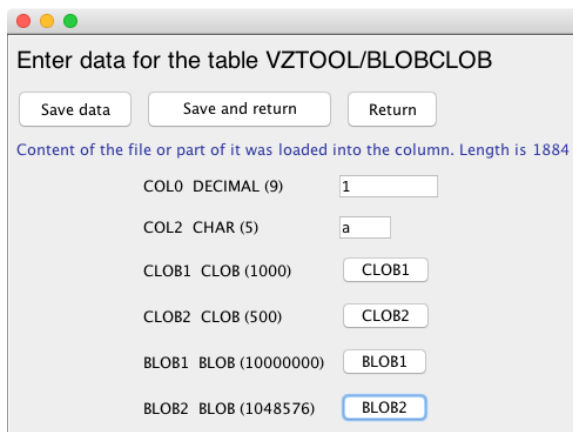
Changing column contents by reading a file

The button *Get file* allows to overwrite the whole contents of the column by contents of a selected text file. A window with the list of files is displayed from which we select e.g. file T1.gif.

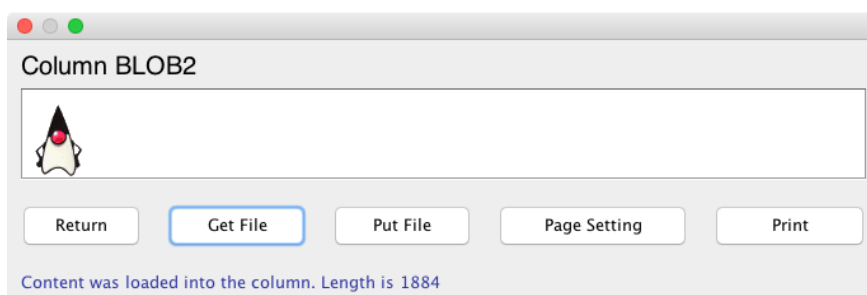


After pressing the button *Open* the picture is taken from the selected file and is displayed in the window.

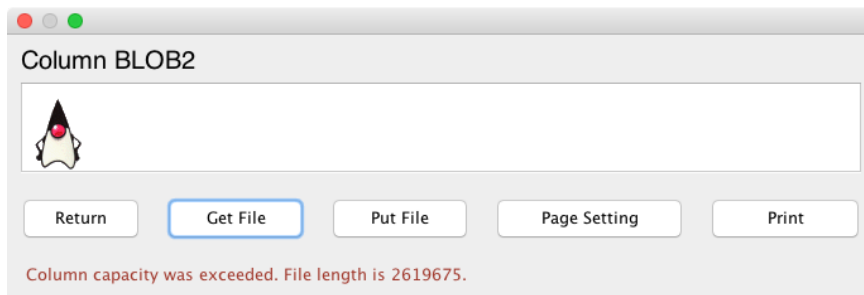
Note: If we press the *Cancel* button, NULL value is stored to the column.



Only after pressing the button BLOB2 again, its contents is displayed, but only if the type of the file is suitable to display.

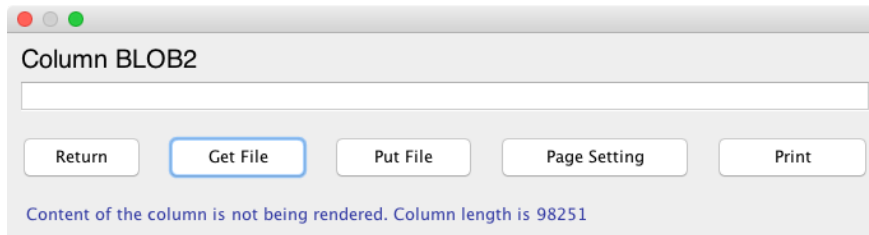


If the contents of the selected file is greater than the capacity of the column (stated in parentheses at the column name), a message is displayed telling that the capacity of the column is not capable to absorb contents of the new file. The picture is not changed.



After pressing the button *Return* the list of columns is displayed again with the message telling that the capacity was exceeded and that the contents remains unchanged.

If the contents of the column is not displayed as a picture, the window is shown with the message telling the length of the column.



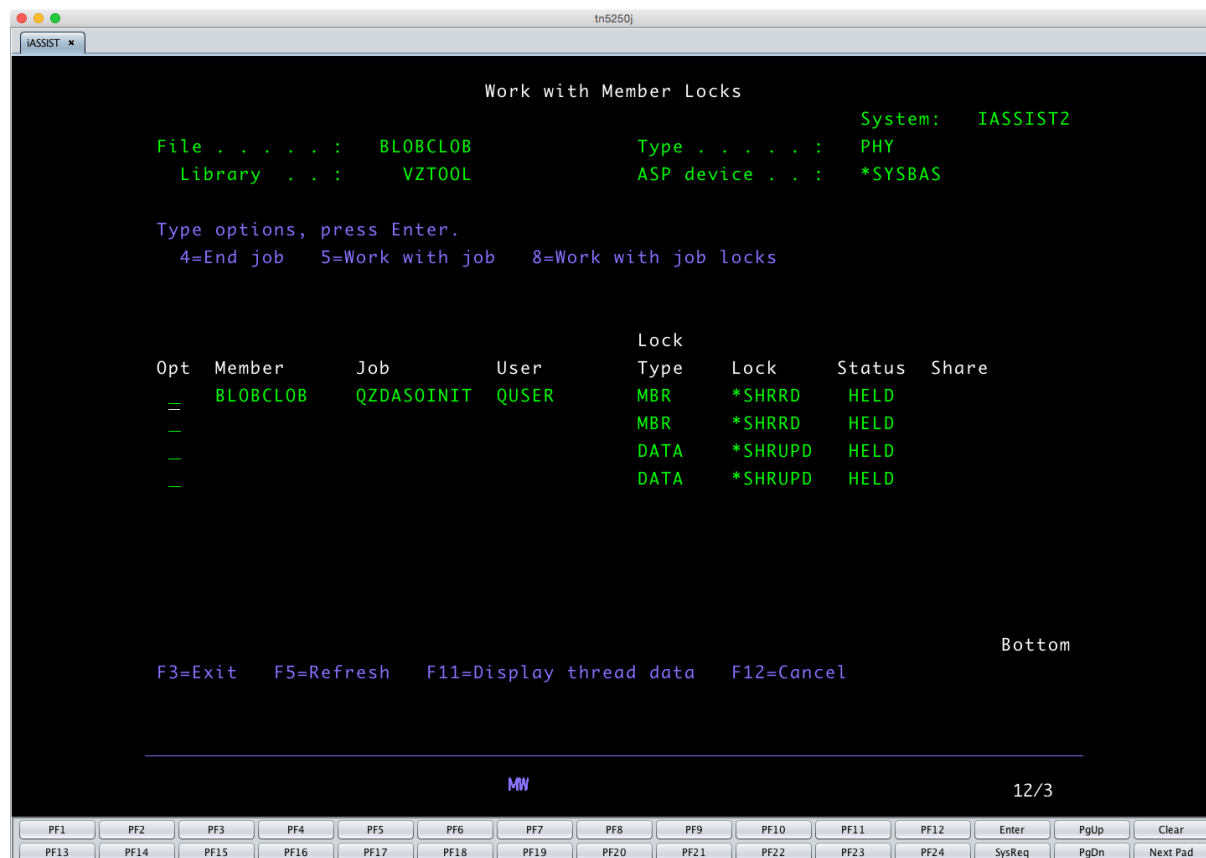
Problems with big columns

Transfer of large volumes of data between the database and the personal computer lasts rather long, especially if flowing through Internet. It is recommended that you are patient until a message is shown (positive or negative).

Sometimes it happens that the connection between the personal computer and the database server is interrupted at the moment when the row (record) is locked for update. In such a case the application is waiting a certain time for a response from the database. If there is no response a message is reported in the file *err.txt* in directory *logfiles* or in the window.

This situation can be remedied if we cancel the job in which the communication was run. It is found using the CL command WRKOBJLCK (Work with Object Locks):

```
WRKOBJLCK OBJ(VZTOOL/BLOBCLOB) OBJTYPE(*FILE) MBR(*FIRST)
```



The job is named QZDASOINIT and we can find it using the CL command WRKACTJOB and then cancel. There may be multiple jobs of that name depending on how many users are working with the database. We should find the job where name is shown who works with the file (BLOBCLOB in the example). Canceling the job releases the record and the application can be launched again and work with the same record.