SQLTEX
v1.4

Oscar van Eijk

May 2, 2002

Contents

---

1 Introduction

SQLTEX is a preprocessor to enable the use of SQL statements in LATEX. It is a perl script that reads an input file containing the SQL commands, and writes a LATEX file that can be processed with your LATEX package.

The SQL commands will be replaced by their values. It's possible to select a single field for substitution substitution in your LaTeX document, or to be used as input in another SQL command.

When an SQL command returns multiple fields and or rows, the values can only be used for substitution in the document.

1.1 Known limitations

- The current version only supports the MySQL database environment.
- SQLTeX reads only one input file; the LaTeX `\include` directive is ignored.
- Currently, only 9 command- line parameters (1-9), and 10 variables (0-9) can be used in SQL statements.
- With SQLTeX it is only possible to read from a database (no updates).
- Replace files can hold only 1,000 items.

2 Installing SQLTeX

Before installing SQLTeX, you need to have it. The latest version can always be found at http://freeware.oveas.com/sqltex. The download consists of this documentation, an installation script for Unix (`install`), and the Perl script `SQLTeX`, and a replace- file (`SQLTeX_r.dat`) for manual installation on non- unix platforms[1].

On a Unix system, make sure the file `install` is executable by issueing the command:

```
bash$ chmod +x install
```

then execute it with:

```
bash$ ./install
```

The script will ask in which directory SQLTeX should be installed. If you are logged in as 'root', the default will be `/usr/local/bin`, otherwise the current directory.
Make sure the directory where SQLTeX is installed is in your path.

For other operating systems, there is no install script, you will have to install it manually.

On OpenVMS it would be something like:

```
$ SET FILE/PROTECTION=(W:RE) SQLTEX.
$ COPY SQLTEX. SYS$SYSTEM:
$ COPY SQLTEX_R.DAT SYS$SYSTEM:
```

[2]

However, on OpenVMS you also need to define the command `SQLTEX` by setting

[1] on Unix, this file will be generated by the install script

[2] Note the dot ('.') at the end of the file; on OpenVMS systems, all files must to have a file extension, which can be empty, in which case the filename ends with a dot.

a symbol, either in the `LOGIN.COM` for all users who need to execute this script, or in some group– or system wide login procedure, with the command:
`$ SQLTEX :== "PERL SYS$SYSTEM:SQLTEX."`

2.1 Configuration

The program starts with a configuration section. The default values are displayed here:

```
#
##############################################################################
# Configurable part
#
$main::texex       = "tex";    # default tex- file extension
$main::stx         = "_stx";   # file name extension to insert before the last '.'
#
$main::cmd_prefix  = "sql";    # prefix for sql-commands (\sql<command>[]{})
$main::sql_open    = "db";     # database declaration, e.g. \sqldb[user,passw]{database}
$main::sql_field   = "field";  # select a single field from db, e.g. \sqlfield{select field from....}
$main::sql_row     = "row";    # select rows from db, e.g. \sqlfield{select * from....}
#
$main::less_av     = 1;        # Is the command 'less' available on this system ?
$main::more_av     = 1;        # Is the command 'more' available on this system ?
#
$main::repl_step   = "OSTX";   # Temporary value for replace
#
##############################################################################
# Do not make any modifications below this line                              #
##############################################################################
```

These values are default values; most values can be overwritten using command line options (see section 4.2). When the command line options are omitted, the default values from the configuration section will be used.

$main::texex The default file extension for LaTeX file. When SQLTeX is called, the first parameter should be the name of the input file. If this filename has no extension, SQLTeX looks for one with the default extension.

$main::stx An output file can be given explicitly using the '`-o`' option. When omitted, SQLTeX composes an output file name using this string.
E.g, if your input file is called `db-doc.tex`, SQLTeX will produce an outputfile with the name `db-doc_stx.tex`.

$main::cmd_prefix SQLTeX looks for SQL commands in the input file. Commands are specified in the same way all LaTeX commands are specified: a backslash (\) followed by the name of the command.
All SQLTeX commands start with the same string. By default, this is the string `sql`. When user commands are defined that start with the same string, this can be changed here to prevent conflicts.

$main::sql_open This string is appended to the $main::cmd_prefix to form the complete SQLTeX for opening a database.
With the default configuration this command is "`\sqldb`".

$main::sql_field This string is appended to the $main::cmd_prefix to form the complete SQLTeX to read a single field from the database.
With the default configuration this command is "`\sqlfield`".

$main::sql_row This string is appended to the $main::cmd_prefix to form the complete SQLTeX to read one or more rows from the database.
With the default configuration this command is "`\sqlrow`".

$main::less_av & $main::more_av These settings are used to determin how the *help* output should be displayed. If the command '`less`' is available on the current system, the output will be parsed through this program. Otherwise the output will be parsed through the program '`more`' if availeble. Both programs are usually available on Unix system (`more` is standard on most Unix systems), but ports for other operating systems are available as well.
Set the values to "0" for the program(s) that is (are) not available, or if you don't want to use it.
If none of these programs is available, the *help* output is plain echoed to the display.

$main::repl_step Replacing strings (see section 2.2 below) is done two steps, to prevent values from being replaced twice. This setting—followed by a three-digit integer - "000" to "999"—is used in the first step and replaces values from the first column. In the second step, values from the second column replace the temporary value.
If the first column in the replace file contains a character sequence that occurs in this temporary value, or if query results might contain the full string followed by three digits, this value might need to be changed in something unique.

2.2 Create replace files

Replace files can be used to substitute values in the output of your SQL commands with a different value. This is especially usefull when the database contains characters that are special characters in LaTeX, like the percent sign ('%'), underscore ('_') etc.

When SQLTeX is installed, it comes with a standard file—`SQLTeX_r.dat`—which is located in the same directory where SQLTeX is installed, with the following replacements:

```
$       \$
_       \_
%       \%
&       \&
<       \texttt{<}
>       \texttt{>}
{       \{
```

```
}       \}
#       \#
~       \~{}
\       \ensuremath{\backslash}
```

These are all single character replacements, but you can add your own replacements that consist of a single character or a character sequence. To do so, enter a new line with the character(string) that should be replaced, followed by a `TAB`- charater (*not* blanks!) and the character(string) it should be replaced with.

If the first non-blank character is a semicolon (';'), the line is considered a comment line. Blank lines are ignored.

The contents of the file are case sensitive, so of you add the line:
`LaTeX          \LaTeX\`
the word "LaTeX" will be changed, but "latex" is untouched.

Different replace files can be created. To select a different replace file for a certain SQLTEX source, use the commandline option '`-r` *filename*'. To disable the use of replace files, use '`-rn`'.

3 Write your SQLTEX file

For SQLTEX, you write your LATEX document just as you're used to. SQLTEX provides you with some extra commands that you can include in your file.
The basic format[3] of an SQLTEX command is:
`\sql`*cmd*`[options]{SQL statement}`

All SQLTEX commands can be specified anywhere in a line, and can span multiple lines. When SQLTEX executes, the commands are read, executed, and their results—if they return any—are written to the output:

Input file:	*Output file:*
`\documentclass[article]`	`\documentclass[article]`
`\pagestyle{empty}`	`\pagestyle{empty}`
`\sqldb[oscar]{mydb}`	
`\begin{document}`	`\begin{document}`

Above you see the SQLTEX command `\sqldb` was removed. Only the command was removed, not the *newline* character at the end of the line, so an empty line will be printed instead. The example below shows the output is an SQLTEX command was found on a line with other LATEX directives:

[3]in this document, in all examples will be asumed the default values in the configuration section as described in section 2.1, have not been changed

Input file:	*Output file:*
`\documentclass[article]`	`\documentclass[article]`
`\pagestyle{empty}\sqldb[oscar]{mydb}`	`\pagestyle{empty}`
`\begin{document}`	`\begin{document}`

In these examples the SQLTEX commands did not return a value. When commands actually read from the database, the returned value is written instead:

Input file:	*Output file:*
`This invoice has \sqlfield{SELECT COUNT(*) FROM INVOICE_LINE WHERE INVOICE_NR = 12345} lines.`	`This invoice has 4 lines`

3.1 SQL statements

This document assumes the reader is familiar with SQL commands. This section only tells something about implementing them in SQLTEX files, especially with the use of command parameters and variables. Details about the SQLTEX commands will be described in the next sections.

Let's look at a simple example. Suppose we want to retreive all header information from the database for a specific invoice. The SQL statement could look something like this:
`SELECT * FROM INVOICE WHERE INVOICE_NR = 12345;`
To implement this statement in an SQLTEX file, the `\sqlrow` command should be used (see section 3.4):

First, it is important to know that SQL statements should *not* contain the ending semicolon (;) in any of the SQLTEX commands. The command in SQLTEX would be:
`\sqlrow{SELECT * FROM INVOICE WHERE INVOICE_NR = 12345}`

Next, SQLTEX would be useless if you have to change your input file every time you want to generate the same document for another invoice.

Therefore, you parameters or variables can be used in your SQL statement. Parameters are given at the command line (see section 4.1), variables can be defined using the `\sqlfield` command as described in section 3.3.1.

Given the example above, the invoice number can be passed as a parameter by rewriting the command as:
`\sqlrow{SELECT * FROM INVOICE WHERE INVOICE_NR = $PAR1}`
or as as variable with the code line:
`\sqlrow{SELECT * FROM INVOICE WHERE INVOICE_NR = $VAR0}`

Note you have to know what datatype is expected by your database. In the example here the datatype is INTEGER. If the field "INVOICE_NR" contains a VARCHAR type, the `$PAR`amater or `$VAR`iable should be enclosed by quotes:
`\sqlrow{SELECT * FROM INVOICE WHERE INVOICE_NR = '$PAR1'}`

3.2 Opening the database

Before any information can be read from a database, this database should be opened. This is done with the `\sqldb` command. `\sqldb` requires the name of the dabatase. Optionally, a username and password can be given. When omitted, SQLTEX assumes no username and password is required to connect to the database (the user that executes SQLTEX should have access to the specified database).
The format of the command is:
`\sqldb[username,password]{database}`

The command can be used anywhere in your input file, but should occur before the first command that tries to read data from the database.

3.3 Reading a single field

When a single field of information is to be read from the database, the command `\sqlfield` is used. By default, the command in the inputfile is replaced by its result in the outputfile.
The SQL command is enclosed by curly braces. Square brackets can optionally be used to enter some extra options. Currently, the only supported option is `setvar` (see section 3.3.1).

The full syntax or the `\sqlfield` command is:
`\sqlrow[`*`options`*`]{SELECT `*`fieldname`*` FROM `*`tablename`*` WHERE `*`your where-clause`*`}`
By default, the SQLTEX command is replaced with the value returned by the SQL query. This behaviour can be changed with options.

3.3.1 Define variables

The `\sqlfield` can also be used to set a variable. The value returned by the SQL query is not displayed in this case. Instead, a variable is created which can be used in any other SQL query later in the document (see also section 3.1).

Therefore, the option `[setvar=`*`n`*`]` is used, where n is an integer between 0 and 9.

Suppose you have an invoice in LATEX. SQLTEX is executed to retrieve the invoice header information from the database for a specific customer. Next, the invoice lines are read from the database.

You could pass the invoice number as a paramater to SQLTEX for use in your queries, but that could change every month. It is easier to :

- pass the customer number as a parameter,
- retrieve the current date (asuming that is the invoice date as stored in the database by another program), and store it in a variable:
  `\sqlfield[setvar=0]{SELECT DATE_FORMAT(NOW(), "%Y-%m-%d")}`
  This creates a variable that can be used as `$VAR0`,

- retrieve the invoice number using the customer number (a command line parameter, see also section 4.1) and the variable containing the invoice date. Store this invoice number in `$VAR1`:
  `\sqlfield[setvar=1]{SELECT INVOICE_NR FROM INVOICES WHERE CUST_NR = '$PAR1' AND INVOICE_DATE = '$VAR0'}`
- use `$VAR1` to retrieve all invoice information.

The SQL queries used here do not display any output in your LaTeXdocument.

3.4 Reading rows of data

When an SQL query returns more information than one single field, the SQLTeX command `\sqlrow` should be used. As with the `\sqlfield`, command, SQLTeX replaces the command with the values it returns, but `\sqlrow` accepts different options for formating the output.

By default, fields are separated by a comma and a blank (‘`, `’), and rows by a newline character (‘\\’). To change this, the options “`fldsep`” and “`rowsep`” can be used.

e.g. In a `tabular` enviroment the fields should be seperated by an amphesand (`&`), perhaps a line should seperate the rows of information. (\\ `\hline`). To do this, the options can be used with `\sqlrow` as shown here:

```
\sqlrow[fldsep=&,rowsep=\\ \hline]{SELECT I.LINE_NR, A.ARTICLE_NR, A.PRICE,
I.AMOUNT, (A.PRICE * I.AMOUNT) FROM ARTICLE A, INVOICE_LINE I WHERE
I.INVOICE_NR = $VAR1 AND I.ARTICLE_NR = A.ARTICLE_NR}
```

This will produce an output like:

```
1 & 9712 & 12 & 1 & 12 \\ \hline
2 & 4768 & 9.75 & 3 & 29.25 \\ \hline
3 & 4363 & 1.95 & 10 & 19.5 \\ \hline
4 & 8375 & 12.5 & 2 & 25 \\ \hline
```

4 Process your SQLTeX file

To process you SQLTeX file and create a LaTeX file with all information read from the database, call SQLTeX with the parameter(s) and (optional) command-line options as described here:

4.1 Parameters

SQLTeX accepts more than one parameter. The first parameter is required; this should be the input file, pointing to your LaTeX document containing the SQLTeX commands.

By default, SQLTeX looks for a file with extension ‘`.tex`’.

All other parameters are used by the queries, if required. If an SQL query contains the string `$PAR`n[4], it is replaced by that parameter (see also section 3.1).

4.2 Command line options

SQLTEX accepts the followint command- line options:

-E *string* replace input file extension in outputfile: `input.tex` will be `input.`*`string`*
For further notes, see option `-e` below

-N `NULL` return values allowed. By default SQLTEX exits if a query returns an empty set.

-P prompt for database password. This overwrites the password in the input file.

-U *user* database username. This overwrites the username in the input file.

-V print version number and exit.

-e *string* add *string* to the output filename: `input.tex` will be `input`*`string`*`.tex`.
This overwrites the configuration setting `$main::stx`
In *string*, the values between curly braces {} will be substituted:

P*n* parameter n

M current monthname (*Mon*)

W current weekday (*Wdy*)

D current date (*yyyymmdd*)

DT current date and time (*yyyymmddhhmmss*)

T current time (*hhmmss*)

e.g., the command '`SQLTeX -e _{P1}_{W} my_file code`' will read '`my_file.tex`' and write '`myfile_code_Tue.tex`' The same command, but with option `-E` would create the outputfile `myfile._code_Tuesday` By default (without `-e` or `-E`) the outputfile `myfile_stx.tex` would have been written. The options `-E` and `-e` cannot be used together or with `-o`.

-f force overwrite of existing files. By default, SQLTEX exists with a warning message it the outputfile already exists.

-h print this help message and exit.

-o *file* specify an output file. Cannot be used with `-e` or `-E`.

-p *prefix* prefix used in the SQLTEX file. Default is `sql` (see also section 2.1 on page 3. This overwrites the configurarion setting `$main::cms_prefix`.

[4] where n is a number between 1 and 9. Note parameter '0' cannot be used, since that contains the filename!

-q run in quiet mode.

-r ***replace*** Specify a file that contains the replace characters (see section 2.2). This is a list with two `TAB`- seperated fields per lione. The first field holds a string that will be replaced in the SQL output

-rn Do not use a replace file. `-rn` and `-r` *`file`* are handled on the same order in which they appear on the commandline and overwrite each other.

-s ***server*** SQL server to connect to. Default is `localhost`.

5 SQLTEX errors and warnings

`no input file specified`

SQLTEX was called without any parameters.
Action: Specify at least one parameter at the commandline. This parameter should be the name of your input file.

`File` *`input filename`* `does not exist`

The input file does not exist.
Action: Make sure the first parameter points to the input file.

`outputfile` *`output filename`* `already exists`

The outputfile cannot be created because it already exists.
Action: Specify another output filename with command line option `-e`, `-E` or `-o`, or force an overwrite with option `-f` (see also section4.2).

`no database opened at line` *`line nr`*

A query starts at line *line nr*, but at that point no database was opened yet.
Action: Add an `\sqldb` command prior to the first query statement.

`insufficient parameters to substitute variable on line` *`line nr`*

The query starting at line *line nr* uses a parameter in a WHERE- clause with `$PAR`*n*, where n is a number bigger than the number of parameters passed to SQLTEX˙
Action: Specify all required parameters at the command line.

`trying to substitute with non existing on line` *`line nr`*

The query starting at line *line nr* requires a variable `$VAR`*n* in its WHERE- clause, where n points to a variable that has not (yet) been set.
Action: Change the number or set the variable prior to this statement.

`trying to overwrite an existing variable on line` *`line nr`*

At line *line nr*, a `\sqlfield` query tries to set a variable n using the option `[setvar=`*n*`]`, but `$VAR`*n* already exists at that point.
Action: Change the number.

`no result set found on line` *`line nr`*

The query starting at line *line nr* returned a `NULL` value. If the option `-N` was specified at the commandline, this is just a warning message. Otherwise, SQLTEX exits.
Action: None.

`result set too big on line` *`line nr`*

The query starting at line *line nr*, called with `\sqlfield` returned more than one field.
Action: Change your query or use `\sqlrow` instead.

`unrecognized command on line` *`line nr`*

At line *line nr*, a command was found that starts with "`\sql`", but this command was not recognized by SQLTEX˙
Action: Check for typos. If the command is a user- defined command, it will conflict with default SQLTEX commands. Change the SQLTEX command prefix (see section 2.1).

`no sql statements found in` *`input filename`*

SQLTEX did not find any valid SQLTEX commands.
Action: Check your input file.

6 Copyright and disclaimer

The latest release is always available at http://freeware.oveas.com/sqltex For bugs, questions and comments, please use the forum available at http://freeware.oveas.com/sqltex/forum.ht