

SQL_TEX

v1.x

Oscar van Eijk

November 21, 2007

Contents

1	Introduction	2
1.1	Known limitations	2
2	Installing SQL_TEX	2
2.1	Configuration	3
2.2	Create replace files	5
3	Write your SQL_TEX file	6
3.1	SQL statements	6
3.2	Opening the database	7
3.3	Reading a single field	7
3.3.1	Define variables	8
3.4	Reading rows of data	8
3.4.1	Output rows on seperate lines	9
3.5	Output multiple documents	9
3.6	Update database records	9
4	Process your SQL_TEX file	10
4.1	Parameters	10
4.2	Command line options	10
5	SQL_TEX errors and warnings	11
6	Copyright and disclaimer	13

1 Introduction

SQL_{TEX} is a preprocessor to enable the use of SQL statements in L^AT_EX. It is a perl script that reads an input file containing the SQL commands, and writes a L^AT_EX file that can be processed with your L^AT_EX package.

The SQL commands will be replaced by their values. It's possible to select a single field for substitution in your L^AT_EX 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

- SQL_{TEX} reads only one input file; the L^AT_EX `\include` directive is ignored.
- Currently, only 9 command- line parameters (1-9), and 10 variables (0-9) can be used in SQL statements.
- Replace files can hold only 1,000 items.
- In multidocument mode, only one parameter can be retrieved.

2 Installing SQL_{TEX}

Before installing SQL_{TEX}, 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 SQL_{TeX}, and a replace- file (SQL_{TeX}.`r.dat`) for manual installation on non- unix platforms¹.

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

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

then execute it with:

```
bash$ ./install
```

The script will ask in which directory SQL_{TEX} 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 SQL_{TEX} 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.2
```

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

²Note 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.

```
$ COPY SQLTEX. SYS$SYSTEM:
$ COPY SQLTEX.R.DAT SYS$SYSTEM:
However, on OPENVMS you also need to define the command SQLTEX by setting
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::dbdriver    = 'mysql';      # Pg, Sybase, Oracle, Ingres, mSQL, ...
$main::oracle_sid  = 'ORASID';     # SID for Oracle users, ignored for other databases
#
$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,passwd]{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::sql_params  = 'setparams';  # set the input parameterlist in multidocument mode.
$main::sql_update  = 'update';    # update rows in db, e.g. \sqlupdate{update table set....}
#
$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::dbdriver Database driver. The default is `mysql`. Other supported databases are `Pg`, `Sybase`, `Oracle`, `Ingres`, `mSQL` and `PostgreSQL`, but also others might work without modification.

If your database driver is not support, look for the function `db_connect` to add support (and please notify me :)

\$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 file-name 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`, SQL_{TEX} will produce an outputfile with the name `db-doc_stx.tex`.

\$main::cmd_prefix SQL_{TEX} looks for SQL commands in the input file. Commands are specified in the same way all L_AT_EX commands are specified: a backslash (\) followed by the name of the command.

All SQL_{TEX} 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 get the complete SQL_{TEX} command 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 get the complete SQL_{TEX} command 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 get the complete SQL_{TEX} command to read one or more rows from the database.

With the default configuration this command is `"\sqlrow"`.

\$main::sql_params This string is appended to the `$main::cmd_prefix` to get the complete SQL_{TEX} command to retrieve a list if fields that will be used as parameters (`$PAR1`, see section 4.1) in the multidocument environment (see section 3.5).

With the default configuration this command is `"\sqlparams"`.

\$main::sql_update This string is appended to the `$main::cmd_prefix` to get the complete SQL_{TEX} command to update one or more rows in the database.

With the default configuration this command is `"\sqlupdate"`.

\$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 available. 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:

\$	$\backslash\$$
-	$\backslash_$
%	$\backslash\%$
&	$\backslash\&$
<	$\backslash\texttt{<}$
>	$\backslash\texttt{>}$
{	$\backslash\{$
}	$\backslash\}$
#	$\backslash\#$
~	$\backslash\sim\{$
\	$\backslash\texttt{ensuremath}\{\backslash\texttt{backslash}\}$

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:

$\text{\LaTeX}\qquad\qquad\backslash\text{\LaTeX}\backslash$

the word “ \LaTeX ” will be changed, but “ $\text{\textit{latex}}$ ” is untouched.

Different replace files can be created. To select a different replace file for a certain \SQLTeX source, use the commandline option $\text{\texttt{--r filename}}$. To disable the use of replace files, use $\text{\texttt{--rn}}$.

3 Write your SQL_{TEX} file

For SQL_{TEX}, you write your L^AT_EX document just as you're used to. SQL_{TEX} provides you with some extra commands that you can include in your file.

The basic format³ of an SQL_{TEX} command is:

```
\sqlcmd[options]{SQL statement}
```

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

Input file:

```
\documentclass[article]
\pagestyle{empty}
\sqldb[oscar]{mydb}
\begin{document}
```

Output file:

```
\documentclass[article]
\pagestyle{empty}
\begin{document}
```

Above you see the SQL_{TEX} 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 SQL_{TEX} command was found on a line with other L^AT_EX directives:

Input file:

```
\documentclass[article]
\pagestyle{empty}\sqldb[oscar]{mydb}
\begin{document}
```

Output file:

```
\documentclass[article]
\pagestyle{empty}
\begin{document}
```

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

Input file:

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

Output file:

```
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 SQL_{TEX} files, especially with the use of command parameters and variables. Details about the SQL_{TEX} commands will be described in the next sections.

³in 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

Let's look at a simple example. Suppose we want to retrieve 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 SQL_{TEX} 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 SQL_{TEX} commands. The command in SQL_{TEX} would be:

```
\sqlrow{SELECT * FROM INVOICE WHERE INVOICE_NR = 12345}
```

Next, SQL_{TEX} 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 a 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 \$PARAMETER or \$VARIABLE 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 database. Optionally, a username and password can be given. When omitted, SQL_{TEX} assumes no username and password is required to connect to the database (the user that executes SQL_{TEX} 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 of 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 parameter 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 (assuming 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 `LATEX` document.

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 formatting 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` environment the fields should be separated by an ampersand (&), perhaps a line should separate 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
```

3.4.1 Output rows on separate lines

Some `LATEX` packages require input on a separate line. If this output is to be read from a database, this can be set with the `rowsep` option using the fixed text “`NEWLINE`”.

3.5 Output multiple documents

A single input file can be created to generate more output files. This option retrieves the first parameter (see section 4.1) from the database (ignoring any parameters that were given on the command line!).

The input document must contain the command `\sqlsetparams` (in the default configuration) without any options. The query that follows can return an unlimited number of rows all containing exactly 1 field:

```
\sqlsetparams{SELECT INVOICE_NR FROM INVOICES WHERE PAY_DATE = NULL}
```

By processing this command, `SQLTEX` builds a list with all values retrieved and processes the input file again for each value.

In those runs, the queries are executed as described in the previous sections, using the value as a parameter:

```
\sqlrow{SELECT * FROM INVOICES WHERE INVOICE_NR = $PAR1}
```

To enable the multidocument mode, the command line switch `-m` must be given and no parameters are allowed.

Without the `-m` switch, a parameter can be given and a single output document will be created, ignoring the `\sqlsetparams` command.

Output filenames will be numbered *filename_1.tex* to *filename_n.tex*.

3.6 Update database records

Since version 1.5, `SQLTEX` supports database updates as well:

```
\sqlupdate{UPDATE INVOICES SET REMINDERS = REMINDERS + 1, LAST_REMINDER
= NOW() INVOICE_NR = $VAR1}
```

This command accepts no options.

4 Process your SQL_{TEX} file

To process you SQL_{TEX} file and create a L_{ATEX} file with all information read from the database, call SQL_{TEX} with the parameter(s) and (optional) command-line options as described here:

4.1 Parameters

SQL_{TEX} accepts more than one parameter. The first parameter is required; this should be the input file, pointing to your L_{ATEX} document containing the SQL_{TEX} commands.

By default, SQL_{TEX} 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

SQL_{TEX} 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 SQL_{TEX} 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 `inputstring.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*)

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

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 if the outputfile already exists.
- h** print this help message and exit.
- m** Multidocument mode; create one document for each parameter that is retrieved from the database in the input document (see section 3.5). This option cannot be used with `-o`.
- 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 4. This overwrites the configuration setting `$main::cms_prefix`.
- 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-separated fields per line. 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 section 4.2).

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.

no parameters for multidocument found on line *line nr*

`SQLTEX` is executed in multidocument mode, but the statement on line *line nr* did not provide any parameters for the documents.

Action: Check your query.

too many fields returned in multidocument mode on line *line nr*

In multidocument mode, the list of parameters retrieved on line *line nr* returned more than one fields per row.

Action: Check your query.

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 SQL_{TEX} commands. Change the SQL_{TEX} command prefix (see section 2.1).

no sql statements found in *input filename*

SQL_{TEX} did not find any valid SQL_{TEX} 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.html>.

Copyright© 2001-2007 - Oscar van Eijk, Oveas Functionality Provider

This software is subject to the terms of the LaTeX Project Public License; see <http://www.ctan.org/tex-archive/help/Catalogue/licenses.lppl.html>.