# Using sqlite3 in Maple. Saving code in the database and evaluating

Nasser M. Abbasi

January 28, 2024 Compiled on January 28, 2024 at 4:54am

## Contents



## 1 Introduction

This describes how to store Maple code fragments in SQL lite3 database, and how to later retrieve and evaluate it and then store the result back into the database.

Assume there is a a set of problems, say a number of ode's, that needs to be saved in the database, then later read and solved. Also the solution to each ode is to be saved in the database.

The Maple code to solve the ode is stored in field of type `TEXT` (or it can be `VARCHAR`).

The same for the solution field. In saving Maple code, any single quote `'` used in the code must be escaped using `''`. The Maple code entered in the database can be free form, just like it is typed in the worksheet. Each command on a new line and terminated by `;`.

Each statement of the Maple code being stored in the database must end by `;` and not `:`. The reason is, when the code is later read, the stored string is split on `;`. If `:` is allowed in the input code, splitting on `:` causes problem as it interfere with assignments in the code that uses `:=`

Hence `;` and not `:` must be used when storing the code in the database.

When reading the Maple code using SQL, each `\n` is first removed from the string and then the string is split `;`. After this, `parse` and `eval` is used on each command.

The following shows step by step how to do all the above, starting from creating the database file, and the schema and running all the needed SQL commands. This was all done inside a worksheet using Maple 2021.2. The database used in sqlite3, which is free.

# 2 The process

## 2.1 Create the physical database file on disk

Locate a folder where to store the database physical file. Let the database file be called `test.db`. In this example, it was created in `/mnt/g/public_html/my_notes/solving_ODE/current_ver`

```
>cd /mnt/g/public_html/my_notes/solving_ODE/current_version/TESTS/test_new_db
>which sqlite3
/home/me/anaconda3/bin//sqlite3
sqlite3 test.db
```

The above creates `test.db` which is now just an empty file.

## 2.2 Use Maple to create the database TABLE initially

Now open a worksheet, and create the Table. Let say for now the table will have two fields. One is for the Maple commands, and second field for the solution of the ODE.

The following needs to be done once, or anytime you want to delete the TABLE and recreate it again.

The Table is given name `PROBLEMS`.

```
restart;
currentdir("G:/public_html/my_notes/solving_ODE/current_version/TESTS/test_new_db");

#connect to the datbase
db := cat(currentdir(),"\\test.db");

try
   conn:=Database[SQLite]:-Open(db):
catch:
   error lastexception;
```

```
end try;

s:=cat("
BEGIN TRANSACTION;
DROP TABLE IF EXISTS PROBLEMS ;
CREATE TABLE PROBLEMS (
   key INTEGER PRIMARY KEY AUTOINCREMENT,
   command VARCHAR,
   result VARCHAR
   );");
Database[SQLite]:-Execute(conn,s);
Database[SQLite]:-Execute(conn, "COMMIT;");
```

Now add couple of problems to fill in the TABLE

```
s:=cat("INSERT INTO PROBLEMS ( command) VALUES(
'
interface(''warnlevel''=4);
kernelopts(''assertlevel''=2);
ode:=diff(y(x),x)=sin(x);
sol:=dsolve(ode,y(x));
'
);");
Database[SQLite]:-Execute(conn,s);

s:=cat("INSERT INTO PROBLEMS ( command ) VALUES(
'
interface(''warnlevel''=4);
kernelopts(''assertlevel''=2);
ode:=diff(y(x),x$2)+diff(y(x),x)+y(x)=sin(x);
sol:=dsolve(ode,y(x));
'
);");
Database[SQLite]:-Execute(conn,s);
```

Now close the database

```
Database[SQLite]:-Close(conn):
```

The result of the above, is that now the database has 2 rows in the Table. There are two columns. The first is the Maple code, and the second is the solution of the ODE

which is added next after reading the table and processing each command.

## 2.3 Read the TABLE and process its content

Now that the database contain the Maple code in it, each problem is read using SQL. This is done as follows

```
restart;
currentdir("G:/public_html/my_notes/solving_ODE/current_version/TESTS/test_new_db");
db := cat(currentdir(),"\\test.db");

#open the datbase
try
   conn:=Database[SQLite]:-Open(db):
catch:
   error lastexception;
end try;

number_of_problems:=Database[SQLite]:-Prepare(conn,
                        "SELECT COUNT(*) from PROBLEMS;");

number_of_problems:=Database[SQLite]:-FetchAll(number_of_problems)[1,1];

#                    number_of_problems := 2

for N from 1 to number_of_problems do
   stmt := Database[SQLite]:-Prepare(conn,
              cat("SELECT command FROM PROBLEMS where rowid=",N,";"));

   stmt := Database[SQLite]:-FetchAll(stmt)[1,1]:
   stmt := StringTools:-Remove("\n",stmt):  #remove \n from string
   stmt := StringTools:-Split(stmt,";" ):   #must split on ";".
   for c in stmt do
        eval(parse(c)):
   od:
   Database[SQLite]:-Execute(conn, "BEGIN TRANSACTION;");
   Database[SQLite]:-Execute(conn,
     cat("UPDATE problems SET result='",convert(sol,string),"' WHERE rowid =",N,";"));

   Database[SQLite]:-Execute(conn, "COMMIT;");
od:
```

```
Database[SQLite]:-Close(conn):
```

Now the database has the table with both the input and also the Maple output stored in it.

The following is a screen shot of the Table in the database using the software `DB Browser for SQLlit` which is very useful and allows viewing the database and its content using GUI based interface.

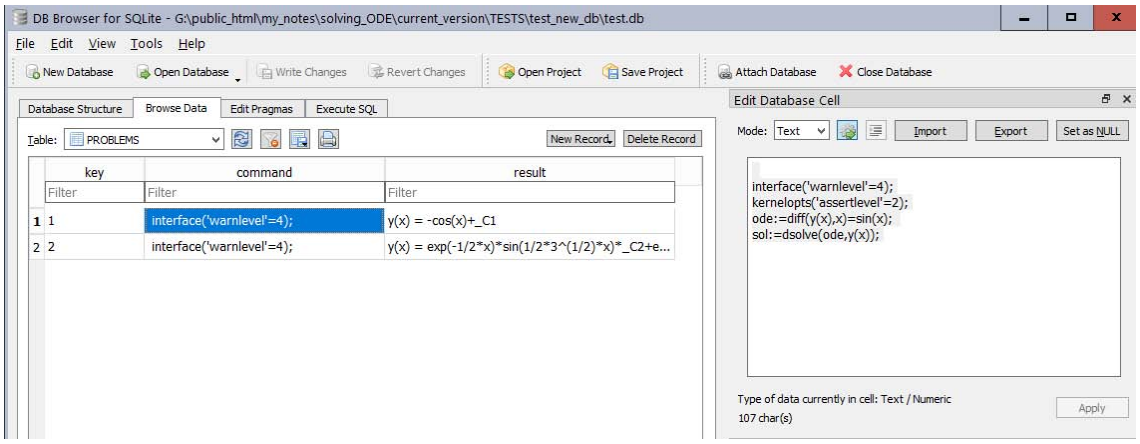


Figure 1: Database view