Creating, Modifying and Populating FileMaker Pro Tables Using BE_FileMakerSQL

by Oliver Reid

FileMaker Pro ("FMP") incorporates a SQL API

> FMP's ExecuteSQL *function* allows you to run SQL "SELECT" queries against TOs in your Schema, but not to run other SQL commands. The Execute SQL *script step* allows you perform a wide range of SQL operations on SQL databases, including an FMP Database, accessible via ODBC.
>
> The BE_FileMakerSQL function, included in the free BaseElements plugin, provides the same access to the SQL API *directly*.

This API is documented here

The accompanying file, Filemaker_SQL.fmp12, includes scripting and some custom functions used to demonstrate execution of these three commands:

`DROP TABLE` *`table_name`* `,`

`CREATE TABLE` *`table_name`* `,` and

`INSERT INTO TABLE` *`table_name`*

This article covers some issues I have worked though when coding with the FileMaker SQL API:

> **Safety Requirements**
>
> **Coding Considerations**

SAFETY REQUIREMENTS

Modifying the file structure while anything else is happening is dangerous.

Ideally, use the above SQL commands:

> Only in development mode - not with live data
>
> Use only in single-user mode

CODING CONSIDERATIONS

Check for completion of API commands

There is no "wait for completion" option for API statements.

I have found that performing any script step that assumes a previous API command has completed, when it has not, can cause both operations to fail.

So the scripting in the demo files checks for completion before finishing.

E.g., for `DROP table` ==>

```
If [ Tables::Table_created ]
    # table exists - uses c.f @baseTable_exists
    Set Variable [ $tos ; Value: @TOs_for_basetable ( Tables::Table Name ) ]
    # find all TO's for base table
    Set Variable [ $to_count ; Value: ValueCount($tos) ]
    Set Variable [ $n ; Value: 1 ]
    Loop
        # deletes all TOs for Base Table
        Set Variable [ $to ; Value: GetValue($tos;$n) ]
        Set Variable [ $sql ; Value: "DROP TABLE  \""&GetValue($tos;$n) &"\"" ]
        Perform Script [ Specified: From list ; "Execute BE_FileMaker_SQL"  ; Parameter: $sql ]
        Set Variable [ $error ; Value: Get(ScriptResult) ]
        Loop
            Exit Loop If [  @to_exists ( $to )=0 or $error ]
            # check action has completed before continuing
            Pause/Resume Script [ Duration (seconds): .01  ]
        End Loop
        If [ $error ]
            Show Custom Dialog [ "An error has occurred:¶¶"&Tables::BE_errors ]
            Exit Script [ Text Result:  ]
        End If
        Set Variable [ $n ; Value: 1+$n ]
        Exit Loop If [ $n>$to_count ]
    End Loop
    Refresh Window [ ]
    Show Custom Dialog [ "Done" ]
Else
    Show Custom Dialog [ "Table Not Yet Created" ]
End If
```

Base Tables and Table Occurrences

FileMaker Pro "Base Tables" are the objects that contain field definitions and data. Each Base Table is represented by at least one Table Occurrence ("TO").

When first creating a table using "Manage Database" tool, FMP creates a Base Table and a TO with the same name (unless that TO name already exists - then " Copy" will be appended to it), and a new layout to enter data into it.

The TO name can be changed and additional TOs for the Base Table can be added to the schema.

The SQL API refers to a `TABLE`. I did some research to make sure I understood if and/or when `TABLE` refers to a BaseTable vs. a TO:

> `CREATE TABLE table_name` creates Base Table and a TO with the same name, unless that TO name already exists in which case the command will fail.
>
> It will also fail if a Base Table with that name already exists, no matter how its TOs are named. (The SQL "`IF NOT EXISTS`" option is not available.)
>
> `DROP TABLE table_name` will remove a TO named for the `table_name`.
>
> - If there is the only one TO for its Base Table, that Base Table, no matter what its name, will also be removed.
> - If there is more than one TO for that Base Table, then the Base Table will not be removed
> - If there is no TO with that name the command will fail, even there is a Base Table with that name
>
> To ensure a specific BaseTable and all its TO's are removed, your script needs to loop through and `DROP` each TO.
>
> `INSERT INTO table_name` requires a TO with the `table_name` name . So to insert data into a specific BaseTable your script must use the name of one of its TO's.

The scripting in the demo file takes all the above into account to perform `DROP`, `CREATE and INSERT` operations in succession

SQL vs. FMP data types

FMP just has Text, Number, Time, Date, Time, Timestamp, Container data types. SQL API requires SQL data types, but not all SQL data types are recognized

The SQL API only allows:

`NUMERIC, DECIMAL, INT`

all these will result in a Number field being created in FMP

For `NUMERIC` and `DECIMAL`, you can specify the precision and scale. For example: `DECIMAL(10,0)`

`VARCHAR, CHARACTER VARYING`

for FMP Text Fields

For these you can specify the length of the string

e.g., `VARCHAR(255)`

Note that `TEXT` is, ironically, not allowed. Nor is `CHAR`.

`DATE, TIME, TIMESTAMP,`

For these, you can specify the precision. For example: TIMESTAMP(6)

`BLOB, VARBINARY, LONGVARBINARY, or BINARY VARYING`.

for container fields

The demo file uses `Varchar` for Text Fields and `Decimal` for Number fields

Primary Key Fields

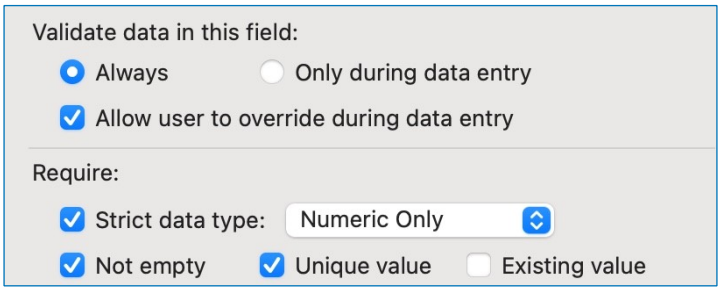
This statement:

```
CREATE TABLE "My Table"
( a_pk INT PRIMARY KEY,
"Name" varchar,
"Date of Birth" Date,
```

```
"Height" decimal,

"Appointment Date-Time" Timestamp)
```

Will create a field "a_pk" with these validation requirements:



However, the SQL "`Auto Increment`" option is not available in the API, and automatic entry of a UUID cannot be specified via a `CREATE TABLE` statement.

Names of Tables and Columns in API Commands

The API does not recognize the backtick (`) character . Enclose column and table names in double quotes to avoid errors caused by problematic names.

Case Sensitivity for API Commands

I have found that for the API SQL command names, keywords, and data types are not case sensitive:

e.g., `droP tABle "My Table"` works.

Data Values for INSERT statements

Date, Time and Timestamp values must be in SQL format.

(The demo file includes some custom functions to convert FileMaker values to SQL formats.)

Data values for all non-numeric data types must enclosed in single quotes Date, Time and Timestamp values must be preceded by the data type" e.g.,

```
VALUES (1, 'John',DATE '2000-01-01',72.5,TIMESTAMP '2021-02-21
10:00:00')
```

Note that Date and Timestamp values are preceded by a label: `DATE` or `TIMESTAMP`

The FMP SQL guide suggests that "*Each `INSERT` statement adds one record to the database table.*" (Page18.)

However, I have that found that a single `INSERT` statement can add multiple rows, e.g.:

```
INSERT INTO "MyTable"

 ("a_pk","Name","Date of Birth","Height","Appointment Date-Time")

 VALUES

(1, 'John',DATE '2000-01-01',72.5,TIMESTAMP '2021-02-21 10:00:00'),

(2, 'Mary',DATE '1996-02-03',67,TIMESTAMP '2021-02-22 03:00:00')
```