From JSON Object to Table

March 19, 2024 Condor Informatique – Turin

Disclaimer

El Condor – CONDOR INFORMATIQUE – Turin

Comments, suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`. The optional parties of syntactic explanation are contained between `[square parentheses]`, alternatives are separated by `|` and the variable parties are in *`italics`*.

Contents

1 Generate Data Base fields from a JSON object

In an application there is often a need to store information which, despite being classifiable under a common typology, nonetheless contain different data, for example a table of events that contains the information like logging, particular activities on data and so on; a minimal solution can be a table with three fields:

- event type,
- JSON data object,
- timestamp (if not in the above field).

However, with this solution, the problem arises of recovering such data perhaps in a form compatible with relational tools id est a table, hence this work, in fact, illustrates a possible solution to this problem using the PHP Data Objects (PDO) extension.

2 JSON object into Data Base tables

2.1 SQLite

The JSON functions and operators were introduced in the SQLite release 3.37.2 and are built by default from the version 3.38.0 (2022-02-22)[1].

The PHP Class described in this document does not use the features offered by SQLite and can therefore be used with versions of SQLite lower than those mentioned above.

2.2 MySql and MariaDB

The Class has been tested with MariaDB and is therefore Mysql compatible.

3 The tool

The script `json2table.php` contains the class `J2t` that has three public functions:

- `json2table`
- `execSQL`
- `extractFields`

The first two functions assume that the database has already been opened (i.e. by creation of a new PDO object).

The scope of the first one function is to create a temporary table using an array of Jason Objects or an SQL statement, the second function is also used internally; the third function returns data extracted from an array, presumably a row of data with any JSON fields.

3.1 Table generation

3.1.1 The `json2table` function

The `json2table` function structure is:

```
$answer = json2table($dbh,$data,$tableName="TemporaryTable",$flds = array())
```

where:

- `$dbh` is an instances of the PDO base class,
- `$data` is an array of JSON objects or an SQL SELECT statement that can contain JSON fields,
- `$tableName` is the name of the temporary table created, ☞ the table isn't generate if it exists in the database,
- `$flds` is an optional array of field names to extract from input for the table to be created.

`json2table` returns an empty string if it succeed otherwise return an error message (see par. 4 Errors).

---

[1] See https://www.sqlite.org/json1.html#jptr

3.1.2 Data from SQL SELECT

The second parameter of the `json2table` method can be a SQL SELECT command where one or more fields can contain a JSON object.

The JSON field can be an *object* i.e. a set of name/value pairs or an array of key value data, see below.

```
{"Author":"Alan Perlis","Source":"","Quote":"Syntactic sugar causes cancer of the
semicolon."}

[{"_id":"9sgo4-hjyyq","author":"The Buddha","content":"Radiate boundless love
towards the entire world — above, below, and across — unhindered, without ill
will, without enmity.","tags":["Wisdom","Love"],"authorSlug":"the-
buddha","length":122,"dateAdded":"2023-03-30","dateModified":"2023-04-14"}]
```

A value can be an array (see above the name `tags`), whose elements are transformed into comma-separated strings.

Note that the SQL command can also contain non JSON fields which will be included along with the JSON generated fields, see parag. 3.1.4.1 Dealing with equal name fields for possibly field name collision.

Example 1: JSON from `SELECT` statement

```
$tTable = "tTable";
$table = ($Table == "Q") ? "Quotes" : "ITQuotes";
...
J2t::json2table($db,"SELECT Count, Quote, SUBSTR(Timestamp,1,10) Date FROM
$table",$tTable);
...
Array
(
    [0] => Array
        (
            [Count] => 0
            [Quote] => {"Author":"Alan Perlis","Source":"","Quote":"Syntactic
sugar causes cancer of the semicolon."}
            [Date] => 2023-05-09
        )
    [1] => Array
        (
            [Count] => 1
            [Quote] => {"Author":"Edsger Dijkstra","Source":"","Quote":"Object-
oriented programming is an exceptionally bad idea which could only have originated
in California."}
            [Date] => 2023-05-09
        )
...
```

3.1.3 Data from array of JSON fields

The second parameter of the `json2table` function can be a simple array such as those obtained by fetching a single column or an array where the items are array containing associative data[2].

---

[2] When the function is invoked with SQL SELECT, data are in this form.

Example 2: JSON fields array

```
$tTable = "tTable";
$table = ($Table == "Q") ? "Quotes" : "ITQuotes";
...
$sql = "SELECT Quote FROM $table";
$sth = $db->prepare($sql);
$sth->execute();
$result = $sth->fetchAll(PDO::FETCH_COLUMN,0);
J2t::json2table($db,$result,$tTable);
...
Array
(
    [0] => {"Author":"Alan Perlis","Source":"","Quote":"Syntactic sugar causes
cancer of the semicolon."}
    [1] => {"Author":"Edsger Dijkstra","Source":"","Quote":"Object-oriented
programming is an exceptionally bad idea which could only have originated in
California."}
    [2] => {"Author":"Edsger Dijkstra","Source":"","Quote":"It is practically
impossible to teach good programming to students that have had a prior exposure to
BASIC: as potential programmers they are mentally mutilated beyond hope of
regeneration."}
    [3] => {"Author":"Edsger Dijkstra","Source":"","Quote":"When FORTRAN has been
called an infantile disorder, full PL\/1, with its growth characteristics of a
dangerous tumor, could turn out to be a fatal disease."}
...
```

Example 3: Use whit array of data and call the class function

```
...
include 'common/json2table.php';
$dbh = new PDO("sqlite:DataBaseFile");
...
$sql = "SELECT Data FROM Loggin WHERE Type = 'MoveDrugs';
$sth = $dbh->prepare($sql);
$sth->execute();
$result = $sth->fetchAll(PDO::FETCH_COLUMN, 0);
$table = "tTable";
J2t::json2table($dbh,$result,$table);
...
```

Example 4: Use whit SQL and call of class method

```
...
include 'common/json2table.php';
$dbh = new PDO("sqlite:DataBaseFile");
...
$sql = "SELECT Data FROM Loggin WHERE Type = 'MoveDrugs'";
$y2t = new J2t;
$answer = $y2t->json2table($dbh,$sql,"t");
if ($answer != "") exit($answer);         // Something went wrong
$result = $y2t->execSql($dbh,"SELECT Count(*) Count, SUM(Qty) Qty FROM t");
if (gettype($result) == "string") echo $result;  // SQL problem
else print_r($result→fetchAll(PDO::FETCH_ASSOC));
$sql = "SELECT * FROM t WHERE Qty > ?";
```

```
print_r($y2t->execSql($dbh,,Array(100))->fetchAll(PDO::FETCH_ASSOC));
...
```

3.1.4 The data extracted

The fields of the generated table will be those of the first element of the input data set or those indicated in the possible fourth parameter of the method, this parameter is an array of field names and possibly data type:

Example 5: Choice of some data

```
include 'json2table.php';
$DBname = "json.sqlite";
$db = new PDO("sqlite:$DBname");
$j2table = new J2t;
...
$sql = "SELECT * FROM Advanced";
$ans = $j2table→json2table($db,$sql,$tTable,array("Store","Qty","Box"=>"TEXT");
```

The field data type is inferred from the input data unless indicated in the fourth parameter; the value accepted are `TEXT`, `INTEGER` and `REAL`.

The usefulness of the fourth parameter is in being able:

- to obtain a subset of the data present,
- to possibly indicate the type of data, for example, treating data stored as character as numeric,
- to indicate field(s) not present in the first row.

3.1.4.1 Dealing with equal name fields

If a field name also appears in JSON object(s), the name assigned to the latter will have the suffix _ followed by the name of the JSON object; ☞ if the field name is an ordinary table field it must appears before the JSON fields (see the SELECT of the example below).
Suppose the row obtained by `SELECT Date, Data, Data2 FROM Advanced` contains:

`Date`	`2024-02-17 08:07:34`
`Data`	`{"Store":"Georgia","ToStore":"Butea","Qty":"711","Items":0.5,"Lot":"12","Box":"2758,2761"}`
`Data2`	`{"Store":"Piscina","AtStore":"Georgia","Date":"March 17","Qty":21,"Items":5.0,"Lot":"19","Box":"16"}`

And the requested fields of the example are (the fourth parameter):

```
Date, Store, AtStore, Box => TEXT, Qty => INTEGER
```

The fields of the temporary table are:

```
Date TEXT, Store TEXT, AtStore TEXT, Box TEXT, Qty INTEGER, Date_Data2 TEXT,
Store_Data2 TEXT, Box_Data2 TEXT, Qty_Data2 INTEGER
```

3.2 Execute SQL

The function `execSQL` is used by the `json2table` function but it is also a method and can be used for execute SQL commands:

```
execSQL($dbh,$sql,$parms=[])
```

The optional `$parms` is an array of values with the elements that are bound in the SQL statement being executed, see the *Example 4: Use whit SQL and call of class method* above.

The function returns a `PDOStatement` object or an error if the SQL isn’t correct, in order to obtain the error the function sets the error mode to `SILENT`, the previous mode is restored after the command execution.

Example 6: ExecSQL function

```
public static function execSQL($dbh,$sql,$parms=[]) {
      $errMode = $dbh→getAttribute(PDO::ATTR_ERRMODE); // save previous
ATTR_ERRMODE
      $dbh->setattribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_SILENT); // disable crash
      $sth = $dbh->prepare($sql);
      $a = $dbh->errorInfo();
      $dbh->setattribute(PDO::ATTR_ERRMODE,$errMode);// restore previous
ATTR_ERRMODE
      if (!$sth) {return "\n<br>$sql\n<br>".$a[2];}
      $sth->execute($parms);
      return $sth;
}
```

4 Errors

`No data!`

`Table` *`table`* `exists` If the temporary table exists (SQLite, Mysql and Maria DB)

Others error like errors on JSON data, are inserted in a temporary table with name *`table`*`_Errors`.

5 Compatibility

Tried with:

PHP	SQLite	MariaDB
5.6.40	3.8.10.2	
8.1.4	3.36.0	11.3.2

6 History

March 2023	First version
0.2.0 8 May 2023	• Input data from SQL SELECT can contain normal fields and one or more JSON field(s). • an item of JSON field can be an array.
0.3.0 March 2024	• Choice on fields, • enhanced type management, • enhanced error management.