Transforming JSON file to relational form

Authors	Specialization	Project receiver
Tomasz Gil	Data Processing Technology	Roche
Łukasz Kobyłecki	Project, Thursday 11:45	

Description

In most conventional databases and data warehouses the typical method of storing, analyzing and processing data is using a relational data form. It’s largely supported and gives a very strict, coherent and often normalized way of presenting data. Obviously, there are other data representation methods that are also widely used, depending on the environment capabilities and requirements, which largely differ from the relational form in many fields. One of them is JSON, used in network communication, supporting loosely structured data.

Taking into consideration the above, there is no concrete or straightforward way of converting JSON to relational form, especially because the very different nature of the data representation. Relational form is fully structured while JSON is semi-structured. Converting from the weakly structured JSON to more demanding relational form requires certain assumptions which will largely affect the data which one will obtain after conversion process.

Nevertheless, in certain scenarios we need to convert JSON data to relational form. That could be due to data storage or analysis requirements, platform support or business applications. For that reason, we created a console application that solves that very problem, with minimal assumptions.

Assumptions

1. Application should be written in Node.js or Python. It has to be a console application.
2. There are two variants of application output. It depends on application mode.
   a. .sql file which contains CREATE TABLE statements of all transformed tables
   b. set of .csv files, each file name corresponds to table name and each file content corresponds to table data
3. Application must have command line interface, for example can handle parameters like:

   a. -i, --input *path/to/file.json*
   b. -m, --mode *sql* | *csv*
   c. -o, --output */path/to/directory*
4. Application must handle references between tables. It has to create valid foreign key constraints.
5. Application must create logs. Logging and its details like log level or log file should be controlled via configuration file or command line parameter.
6. Application must have configuration file. Application should not contain hardcoded values.
7. Application must be installable with standard tools of used programming language.
8. Application must contain a README.md file which describes the purpose of use and describes how to use it.
9. Application name should be invented by the team.

Work progress

The entire project was held during 3 month period in close collaboration with project owner from Roche. Initially we splitted known requirement into small, manageable tasks. Timetable with milestone dates (delivery of the most substantial parts of the application) can be seen below.

7.03.2019

- Meeting with project owner from Roche
- Setting project requirements
- Input data analysis
- Creating code repository
- Creating tasks and work planning

12.03.2019

- Programming language comparison - Node.js vs Python
- Environment setup
- First feedback phase from project owner

2.04.2019

- Command line interface
- Loading and parsing input file
- Creating json schema
- Sql mode transformation
- Configuration file
- Logging
- Basic README

9.04.2019

- Second feedback phase from project owner
- Application testing

25.04.2019

- Sql mode transformation update
- Input json preprocessing
- Csv mode transformation
- README improvement

6.05.2019

- Refining requirements with project owner from Roche

18.05.2019

- General improvements and refactoring
- First release
- Documentation

Documentation

Application package is available via NPM (node package manager), which is a global package repository for JavaScript.
https://www.npmjs.com/package/json-to-rel

Software is open-source, the code is available in github repository.
https://github.com/tomaszgil/json-to-rel

Installation

Prerequisites

In order to successfully install and run the software, your system has to fulfill given criteria:

- Node.js environment installed (version 10.15.3 or newer)
- NPM command line interface (version 6.9.0 or newer)

Installation process

You can install the application using NPM command line interface:

```
npm install --global json-to-rel
```

Alternatively, you can use other dependency management tools like Yarn:

```
yarn global add json-to-rel
```

Application usage

You can run the application using a command line with a command:

```
json-to-rel -i input-file.json -m sql -o output-dir/
```

Full collection of parameters, that can be passed to our application through command line, can be seen in a table below.

Parameter	Description	Short flag	Long flag	Possible value	Required
Input file	Path to a file with input JSON object	`-i`	`--input`	`path/to/file.json`	`Yes`
Output directory	Path to a directory in which the application will store files with the results of the processing	`-o`	`--output`	`path/to/directory`	`Yes`
Mode	Type of data and files	`-m`	`--mode`	`sql | csv`	`Yes`

	that will be generated				
Logging	Activating optional logging to a file with log's level of detail (the higher the value, the more detailed the log messages)	`-l`	`--logging`	`0 | 1 | 2`	`No`
Configuration file	Path to file with additional configuration JSON object	`-c`	`--config`	`path/to/file.json`	`No`

You can optionally pass your own configuration in form of a JSON file. If the provided values are valid, the application will override the default configuration with passed properties. Example configuration file can be seen below.

```
{
  "inputFileEncoding": "utf8",
  "outputFileName": "tables.sql",
  "rootTableName": "GENERATED",
  "surrogatePrimaryKeyName": "__ID",
  "logFile": "json2rel.log",
  "csvDelimiters": {
    "col": ";",
    "row": "\n"
  },
  "generatedAttributeName": "value",
  "truncateTableName": false
}
```

Meaning of the parameters with possible values can be seen in a table below (all keys are optional).

Parameter	**Key**	**Value type**
Input file encoding	`inputFileEncoding`	`string`
Output file name	`outputFileName`	`string`

Table name, that will be given to the top most table	`rootTableName`	`string`
Suffix, that will be appended to the surrogate primary key column	`surrogatePrimaryKeyName`	`string`
Log file name	`logFile`	`string`
Column delimiter in output CSV files	`csvDelimiters.col`	`string`
Row delimiter in output CSV files	`csvDelimiters.row`	`string`
Column name for attributes without deducible name (arrays of simple types)	`generatedAttributeName`	`string`
Activating a function that shortens table names	`truncateTableName`	`boolean`

Output

The application offers two types of output, based on which application processing and the output information slightly varies. For both output modes table names reflect input JSON file structure.

1. **SQL**. Output is a single SQL file with a set of `create table` statements representing relational tables mapped from input JSON object. Each table features a surrogate primary key. Tables are connected with foreign key constraints. Syntax is compliant with Sqlite 3.
2. **CSV**. Output is a collection of CSV files with file names corresponding to table names. Each file contains a header with table column names and rows which correspond to table records with values from input json object.

Development

For developing application the same initial prerequisites have to be met. We also recommend installing `git` client.

1. Clone repository (if you don't have `git` client installed, download the repository from github link given above)

   ```
   git clone https://github.com/tomaszgil/json-to-rel.git
   ```
2. Install project dependencies

   ```
   npm install
   ```
3. Run application (pay attention to the way arguments are passed)

   ```
   npm start -- --input /path/to/input --mode sql --output
   /path/to/output
   ```

Building and testing

If you want to test and build the application you can do it by simulating package installation and running the application

1. Simulate package installation. This will rebuild and make the application accessible globally.

   ```
   npm link
   ```
2. Run the application.

   ```
   json-to-rel -i input-file.json -m sql -o output-dir/
   ```
3. Once you're done, remove the application.

   ```
   npm unlink
   ```