MIT World Peace University

Database Management Systems
Second Year B. Tech, Semester 4

Basics of JSON

Assignment No. 9

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

May 4, 2023

Contents

1 Aim

Create a json document and write the json query to displayt the data.

2 Objectives

1. To understand the key structure elements of a json file: object names and values.
2. To study the core data types that json files can store including: boolean, numeric and string; hierarchical json structures including: objects, arrays and data elements.
3. To use MySQL JSON data type to store JSON documents in the database and perform CRUD operations.

3 Problem Statement

Create tables and solve given queries.

4 Theory

4.1 Introduction to JSON

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. It is a text format that is easy to read and write and is often used for transmitting data between a server and a web application. JSON has become popular because of its simplicity and flexibility in handling data structures, making it a popular choice for data exchange between systems.

JSON is based on two basic structures: key-value pairs and arrays. Key-value pairs consist of a key and a value, separated by a colon, and are enclosed in curly braces. Arrays are lists of values and are enclosed in square brackets. JSON is human-readable, which means that it can be easily understood by people, and it is also machine-readable, which means that computers can easily parse and generate JSON data.

4.2 Reading and Writing Files in Python

Python provides built-in support for reading and writing files. To open a file in Python, you can use the open() function, which takes two arguments: the file name and the mode in which you want to open the file. The mode can be "r" for reading, "w" for writing, "a" for appending, and "x" for exclusive creation. Once you have opened a file, you can read or write data to it using the file object's methods.

To read data from a file, you can use the read() method, which reads the entire contents of the file as a string. You can also use the readline() method to read a single line at a time or the readlines() method to read all the lines into a list. To write data to a file, you can use the write() method, which writes a string to the file, or the writelines() method, which writes a list of strings to the file.

Syntax for reading a file in Python:

```
with open("file.txt", "r") as f:
    data = f.read()
```

Syntax for writing to a file in Python:

```
with open("file.txt", "w") as f:
    f.write("Hello, world!")
```

4.3 Writing to a JSON File

Python provides a built-in module called json for working with JSON data. To write JSON data to a file, you can use the json.dump() function, which takes two arguments: the data you want to write and the file object you want to write it to. The json.dump() function automatically converts the data to JSON format and writes it to the file.

Syntax for writing JSON data to a file in Python:

```
import json

data = {"name": "John", "age": 30, "city": "New York"}

with open("data.json", "w") as f:
json.dump(data, f)
```

4.4 Introduction to MySQL and JSON data type

MySQL is a popular open-source relational database management system that provides a way to store and manage structured data. MySQL supports a JSON data type, which allows you to store JSON data in a column of a table. The JSON data type is a flexible and efficient way to store and manipulate data that has a variable structure.

When you store JSON data in a MySQL database, you can use SQL to query and manipulate the data just like you would with any other data type. MySQL provides a set of functions and operators for working with JSON data, such as JSONEXTRACT() for extracting data from a JSON object, JSONARRAY() for creating a JSON array, and JSONOBJECT() for creating a JSON object.

Syntax for creating a table with a JSON column in MySQL:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    json_column JSON,
    ...
);
```

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Interpreters Used: Python 3.11

6 Input

1. Read CSV file and convert it into json file with python
2. Read text file and convert it into json file with python
3. Database with JSON Data field

7 Outputs

```python
## read a csv file and convert it into json file
import csv
import json

# Open the CSV file and read its contents
with open('data.csv', newline='') as csv_file:
    csv_reader = csv.DictReader(csv_file)

    # Create an empty list to hold the JSON objects
    json_objects = []

    # Loop through each row in the CSV file
    for row in csv_reader:
        # Convert the row to a JSON object and append it to the list
        json_objects.append(row)

# Write the JSON objects to a file with indentation
with open('data.json', 'w') as json_file:
    json.dump(json_objects, json_file, indent=4)
```

Listing 1: Python Code

```
ID, NAME, AGE, CITY
1, John, 20, New York
2, Mary, 25, Boston
3, Peter, 30, Chicago
4, John, 35, New York
5, Mary, 40, Boston
6, Peter, 45, Chicago
7, John, 50, New York
8, Mary, 55, Boston
9, Peter, 60, Chicago
10, Randy, 65, Los Angeles
```

Listing 2: CSV Input

```
[
    {
        "ID": "1",
        " NAME": " John",
        " AGE": " 20",
        " CITY": " New York"
    },
    {
        "ID": "2",
        " NAME": " Mary",
        " AGE": " 25",
        " CITY": " Boston"
    },
    {
        "ID": "3",
        " NAME": " Peter",
        " AGE": " 30",
        " CITY": " Chicago"
    },
    {
        "ID": "4",
```

```
        " NAME": " John",
        " AGE": " 35",
        " CITY": " New York"
    },
    {
        "ID": "5",
        " NAME": " Mary",
        " AGE": " 40",
        " CITY": " Boston"
    },
    {
        "ID": "6",
        " NAME": " Peter",
        " AGE": " 45",
        " CITY": " Chicago"
    },
    {
        "ID": "7",
        " NAME": " John",
        " AGE": " 50",
        " CITY": " New York"
    },
    {
        "ID": "8",
        " NAME": " Mary",
        " AGE": " 55",
        " CITY": " Boston"
    },
    {
        "ID": "9",
        " NAME": " Peter",
        " AGE": " 60",
        " CITY": " Chicago"
    },
    {
        "ID": "10",
        " NAME": " Randy",
        " AGE": " 65",
        " CITY": " Los Angeles"
    }
]
```


Listing 3: JSON Output

8 Conclusion

Thus, we have learned about the data types, libraries and methods to create JSON documents for storing and retrieving the data.

9 FAQ

1. What is JSON?

   JSON stands for JavaScript Object Notation, which is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is often used for transmitting data between a server and a web application.

2. Mention what is the rule for JSON syntax rules? Give an example of JSON object?

   JSON syntax follows a set of rules that determine how data is represented in a JSON object. Some of the key rules for JSON syntax are:

   (a) JSON data is represented as key-value pairs, enclosed in curly braces .

   (b) Each key in a JSON object must be a string enclosed in double quotes, followed by a colon, and then its value.

   (c) Multiple key-value pairs in a JSON object are separated by commas.

   Here's an example of a JSON object that represents information about a person:

```
{
"name": "John Doe",
"age": 30,
"address": {
"street": "123 Main St",
"city": "Anytown",
"state": "CA",
"zip": "12345"
},
"phone": [
{
  "type": "home",
  "number": "555-1234"
},
{
"type": "work",
"number": "555-5678"
}
]
}
```

   In this example, the JSON object represents a person named John Doe, including their name, age, address, and phone numbers.

3. Mention what are the data types supported by JSON?
   JSON supports a limited set of data types, including:

   (a) string: a sequence of characters enclosed in double quotes

   (b) number: an integer or floating-point value

   (c) boolean: either true or false

   (d) null: a special value representing "no value"

   (e) array: an ordered list of values, enclosed in square brackets [ ]

(f) object: an unordered collection of key-value pairs, enclosed in curly braces

4. List out the uses of JSON?

   JSON is a widely used format for data interchange between systems, and it has many applications, including:

   (a) Web APIs: JSON is often used as the data format for APIs that provide access to web services and data.

   (b) Configuration files: JSON can be used to store configuration data for applications and systems.

   (c) Data storage: JSON can be used to store data in databases or file systems.

   (d) Data exchange: JSON can be used to exchange data between different programming languages and platforms.

   (e) Front-end development: JSON can be used to represent data in web applications and JavaScript programs.