MySQL JSON

JSON abbreviated as **JavaScript Object Notation**. It is a **lightweight data-interchange format** similar to other data types and can be easily read and write by humans. It can also be parsed and generate by machines easily.
Generally, the JSON data type supports two structures:

- A collection of name/value pairs chain, which acts as a data array.
- An ordered list of values.

Since it manages the individual values in a **name-value pair chain** that acts as a data array, we can retrieve the whole field using a single command. This useful feature allows us to retrieve the data in a large system quickly.

Why we use JSON?

JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications
e.g., Sending some data from the server to the client, so it can be displayed on a web page, or vice versa.

Advantages Of JSON :
The JSON data type provides certain advantages over storing JSON-format strings in a string column:

1. Automatic content validation: When you're adding JSON data to MySQL, the database automatically confirms that the data format fits the data and doesn't allow you to save it if it doesn't match.
2. Faster data transfer: All calls from other clients require data conversion from record to JSON. Saving data directly in JSON makes data transfer more efficient. In addition, JSON can have a substantially lower character count, reducing the overhead in data transfers.
3. Readability: Since JSON data is saved in text format, not XML, it's more easily readable by the human eye.
4. Easy exchange: JSON is helpful when it comes to data exchange between heterogeneous systems since you can also use the JSON format with other programming languages.
5. Potential to combine and store: JSON documents with multiple keys and value attributes can be saved in the same document.

Disadvantages Of JSON :

1. Indexing: MySQL doesn't support indexing of JSON columns, which means that if you want to search through your JSON documents, you could trigger a full table scan. A JSON column cannot be indexed directly. However, if you wish to perform more efficient searches, you can generate a column with values extracted from the JSON column, on which you can create an index.
2. Limited storage: JSON documents stored in MySQL can only reach a theoretical maximum size of 1GB.

3. Inefficient storage: JSON could be more storage efficient. If your priority is optimizing data architecture by prioritizing storage efficiency in your database schema, you may be better off with more traditional data types such as INT, CHAR, VARCHAR, and the like.

The storage space required for the JSON document is roughly the same as the storage requirements for **LONGBLOB** and **LONGTEXT**.

We can define the JSON data type column in the MySQL table using the following syntax :

```
CREATE TABLE table_name(
        ...
        json_column_nameJSON,
        ...
        );
```

Example to create the table :

```
CREATE TABLE events
(
  id int auto_increment primary key,
  event_name varchar(10),
  visitor varchar(10),
  properties json,
  browser json
)
```

Example to insert the data:

```
INSERT INTO events(event_name, visitor,properties, browser)
VALUES (
  'pageview',
   '1',
   '{ "page": "/" }',
   '{ "name": "Safari", "os": "Mac", "resolution": { "x": 1920, "y": 1080 } }'
),
('pageview',
  '2',
  '{ "page": "/contact" }',
  '{ "name": "Firefox", "os": "Windows", "resolution": { "x": 2560, "y": 1600 } }'
),
(
  'pageview',
  '1',
  '{ "page": "/products" }',
  '{ "name": "Safari", "os": "Mac", "resolution": { "x": 1920, "y": 1080 } }'
),

);
```

Reading Data

1. We can run a simple SELECT statement to see the data in the table.

```
Select * from tablename;
```

2. To pull values out of the JSON columns, you use the column path operator (->).

Example :

```
SELECT id, browser->'$.name' browser
  FROM events;
```

3. To remove the quote marks, you use the inline path operator (->>) as follows:

Example :

```
SELECT id, browser->>'$.name' browser
FROM events;
```

or

for multiple columns :

```
SELECT id, browser ->> '$.name' field1, browser ->> '$.os' filed2 FROM events;
```

or

for multiple columns with where condition from JSON field:

```
SELECT id, browser ->> '$.name' field1, browser ->> '$.os' field2 FROM events
where  browser ->> '$.name' = 'Chrome';
```

or

Select with where contion

```
SELECT id, browser  FROM events where browser ->> '$.name' = 'Chrome';
```

4. When used in combination with the JSON_EXTRACT(), you can retrieve the values for the specified column.

Example:

```
SELECT event_name, JSON_EXTRACT(browser, '$.name') FROM events;
```

Update JSON Data in MySQL

We can update data in JSON fields with the JSON_INSERT(), JSON_REPLACE(), and JSON_SET() functions. These functions also require a path expression to specify which parts of the JSON object to modify. The output of these functions is a valid JSON object with the changes applied.

Requirement	Function
Add a new key and value	JSON_INSERT
Update a value for an existing key	JSON_REPLACE
Add a new key and value or update an existing key’s value	JSON_SET

JSON_INSERT
A member not present in an existing object. The member is added to the object and associated with the new value.

```
-- update events set browser = JSON_INSERT(browser,'$.version','4.0') where id =1;
-- update events set browser = JSON_INSERT(browser,'$.version','4.0'  , '$.year','2000' )
      where id=1;
--  update events set browser = JSON_INSERT(browser,'$.resolution.z','4000') where id =1;
```

JSON_REPLACE
Replaces existing values in a JSON document and returns the result.

```
--  update events set browser = JSON_Replace(browser ,'$.os','Macc') where id = 1;
```

JSON_SET
Inserts or updates data in a JSON document and returns the result.

```
-- update events set browser = json_set(browser,'$.os','Mac','$.rating','5') where id =1;
```

Deleting the records

- We can delete data in JSON fields with the JSON_REMOVE() function and DELETE.
- JSON_REMOVE() : function returns the updated JSON after removing the specified key based on the path expression.
- Using JSON_REMOVE function, it is possible to remove the screen key/value pairs from product:
- There are two DELETE operations you can do when working with JSON fields:
    -- delete an attribute from a JSON field
    -- delete a row from your table

1. delete a row from your table

```
--- delete from events where json_extract(browser, '$.os') = 'Mac';
```

2. delete an attribute from a JSON field

```
-- update events set browser = json_remove(browser,'$.rating') where id =3;
```

JSON FUNCTIONS

- **JSON_APPEND -**
  - Append data to JSON document
  - Appends values to the end of the indicated arrays within a JSON document and returns the result. This function was renamed to JSON_ARRAY_APPEND()

    Example :

    select json_array_append(browser,'$.name',"Firefox") from events where id =3;

- **JSON_INSERT**

  Example:

  update events set browser = JSON_INSERT(browser,'$.resolution.z','4000') where id =1;
- **JSON_LENGTH**

  Example:

  select json_length(browser) from events where id =3;
- **JSON_MERGE()**

  Example:

  select json_merge(browser,properties) from events where id =3;
- **JSON_PRETTY()**

  Example:

  select json_pretty(browser) from events where id =3;
- **JSON_REMOVE()**

  Example:

  update events set browser = json_remove(browser,'$.rating') where id =3;
- **JSON_REPLACE()**

  Example:

  update events set browser = JSON_Replace(browser ,'$.os','Macc') where id = 1;