SQL databases tend to be rigid in design. By its nature, the structured query language enforces data type and size constraints.

In comparison, NoSQL databases encourage flexibility in design.

The JSON data type in MySQL grants the strengths of both of these systems. It allows to structure some parts of the database and leave others to be flexible.

MySQL supports the native JSON data type since version 5.7.8.

To define a column whose data type is JSON, you use the following syntax:

CREATE TABLE table\_name (

...

json\_column\_name JSON,

...

);

Notice that a JSON column cannot have a default value. In addition, a JSON column cannot be [indexed](https://www.mysqltutorial.org/mysql-index/mysql-create-index/) directly.

Suppose, we have to track the visitors and their actions on our website. Some visitors may just view the pages and other may view the pages and buy the products.

CREATE TABLE events(

id int auto\_increment primary key,

event\_name varchar(255),

visitor varchar(255),

properties json,

browser json

);

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 } }'

),

(

'purchase',

'3',

'{ "amount": 200 }',

'{ "name": "Firefox", "os": "Windows", "resolution": { "x": 1600, "y": 900 } }'

),

(

'purchase',

'4',

'{ "amount": 150 }',

'{ "name": "Firefox", "os": "Windows", "resolution": { "x": 1280, "y": 800 } }'

),

(

'purchase',

'4',

'{ "amount": 500 }',

'{ "name": "Chrome", "os": "Windows", "resolution": { "x": 1680, "y": 1050 } }'

);

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

SELECT id, browser->'$.name' browser

FROM events;

+----+-----------+

| id | browser |

+----+-----------+

| 1 | "Safari" |

| 2 | "Firefox" |

| 3 | "Safari" |

| 4 | "Firefox" |

| 5 | "Firefox" |

| 6 | "Chrome" |

+----+-----------+

6 rows in set (0.00 sec)

Notice that data in the browser column is surrounded by quote marks. To remove the quote marks, you use the inline path operator (->>) as follows:

SELECT id, browser->>'$.name' browser

FROM events;

+----+---------+

| id | browser |

+----+---------+

| 1 | Safari |

| 2 | Firefox |

| 3 | Safari |

| 4 | Firefox |

| 5 | Firefox |

| 6 | Chrome |

+----+---------+

6 rows in set (0.00 sec)

SELECT browser->>'$.name' browser,

count(browser)

FROM events

GROUP BY browser->>'$.name';

The output of the query is as follows:

+---------+----------------+

| browser | count(browser) |

+---------+----------------+

| Safari | 2 |

| Firefox | 3 |

| Chrome | 1 |

+---------+----------------+

3 rows in set (0.02 sec)

SELECT visitor, SUM(properties->>'$.amount') revenue

FROM events

WHERE properties->>'$.amount' > 0

GROUP BY visitor;

Here is the output:

+---------+---------+

| visitor | revenue |

+---------+---------+

| 3 | 200 |

| 4 | 650 |

+---------+---------+

2 rows in set (0.00 sec)