MySQL8.0 json 使用 类型 查询 函数

一,对记录的操作

1.创建有json字段的表

```
-- 创建表
CREATE TABLE t_json(id INT PRIMARY KEY, sname VARCHAR(20) , info  JSON);
```

2.插入记录

```
-- 插入含有json数组的记录
INSERT INTO t_json(id,sname,info) VALUES( 1, 'name1', JSON_ARRAY(1, "abc", NULL, TRUE, CURTIME()));
-- 插入含有json对象的记录
INSERT INTO t_json(id,sname,info) VALUES( 2, 'name2', JSON_OBJECT("age", 20, "time", now()));
 INSERT INTO t_json(id,sname,info) VALUES( 3, 'name3', '{"age":20, "time":"2018-07-14 10:52:00"}');
```

3.查询记录

```
-- 查询记录
SELECT sname,JSON_EXTRACT(info,'$.age') FROM t_json;
SELECT sname,info->'$.age' FROM t_json;
-- 查询key
SELECT id,json_keys(info) FROM t_json;
```

4.修改记录

```
-- 增加键
UPDATE t_json SET info = json_set(info,'$.ip','192.168.1.1') WHERE id = 2;

-- 变更值
UPDATE t_json SET info = json_set(info,'$.ip','192.168.1.2') WHERE id = 2;

-- 删除键
UPDATE t_json SET info = json_remove(info,'$.ip') WHERE id = 2;
```

二,创建json值函数

1.JSON_ARRAY 生成json数组

```
-- JSON_ARRAY(val1,val2,val3...)
-- 生成一个包含指定元素的json数组。
SELECT JSON_ARRAY(1, "abc", NULL, TRUE, CURTIME()); -- [1, "abc", null, true, "10:37:08.000000"]
```

2.JSON_OBJECT 生成json对象

```
-- JSON_OBJECT(key1,val1,key2,val2...)
-- 生成一个包含指定K-V对的json object。如果有key为NULL或参数个数为奇数，则抛错。
SELECT JSON_OBJECT('age', 20, 'time', now()); -- {"id": 87, "name": "carrot"}
```

3.JSON_QUOTE 加"号

```
-- JSON_QUOTE(json_val)
-- 将json_val用"号括起来。
SELECT JSON_QUOTE('[1,2,3]'); -- "[1,2,3]"
```

三,搜索json值函数

1.JSON_CONTAINS 指定数据是否存在

```
set @j = '{"a": 1, "b": 2, "c": {"d": 4}}';
-- JSON_CONTAINS(json_doc, val[, path])
-- 查询json文档是否在指定path包含指定的数据，包含则返回1，否则返回0。如果有参数为NULL或path不存在，则返回NULL。
SELECT JSON_CONTAINS(@j, '4', '$.c.d'); -- 1
```

2.JSON_CONTAINS_PATH 指定路径是否存在

```
-- JSON_CONTAINS_PATH(json_doc, one_or_all, path[, path] ...)
-- 查询是否存在指定路径，存在则返回1，否则返回0。如果有参数为NULL，则返回NULL。
-- one_or_all只能取值"one"或"all"，one表示只要有一个存在即可；all表示所有的都存在才行。
SELECT JSON_CONTAINS_PATH(@j, 'one', '$.a', '$.e'); -- 1
SELECT JSON_CONTAINS_PATH(@j, 'all', '$.a', '$.c.d'); -- 1
```

3.JSON_EXTRACT 查找所有指定数据

```
-- JSON_EXTRACT(json_doc, path[, path] ...)
-- 从json文档里抽取数据。如果有参数有NULL或path不存在，则返回NULL。如果抽取出多个path，则返回的数据封闭在一个json array里。
set @j2 = '[10, 20, [30, 40]]';
SELECT JSON_EXTRACT('[10, 20, [30, 40]]', '$[1]'); -- 20
SELECT JSON_EXTRACT('[10, 20, [30, 40]]', '$[1]', '$[0]'); -- [20, 10]
SELECT JSON_EXTRACT('[10, 20, [30, 40]]', '$[2][*]'); -- [30, 40]
```

4.JSON_KEYS 查找所有指定键值

```
-- JSON_KEYS(json_doc[, path])
-- 获取json文档在指定路径下的所有键值，返回一个json array。如果有参数为NULL或path不存在，则返回NULL。
SELECT JSON_KEYS('{"a": 1, "b": {"c": 30}}'); -- ["a", "b"]
SELECT JSON_KEYS('{"a": 1, "b": {"c": 30}}', '$.b'); -- ["c"]
SELECT id,json_keys(info) FROM t_json;
```

5.JSON_SEARCH 查找所有指定值的位置

```
-- JSON_SEARCH(json_doc, one_or_all, search_str[, escape_char[, path] ...])
-- 查询包含指定字符串的paths，并作为一个json array返回。如果有参数为NUL或path不存在，则返回NULL。
-- one_or_all："one"表示查询到一个即返回；"all"表示查询所有。
-- search_str：要查询的字符串。 可以用LIKE里的'%'或‘_’匹配。
-- path：在指定path下查。
SET @j3 = '["abc", [{"k": "10"}, "def"], {"x":"abc"}, {"y":"bcd"}]';
SELECT JSON_SEARCH(@j3, 'one', 'abc'); -- "$[0]"
SELECT JSON_SEARCH(@j3, 'all', 'abc'); -- ["$[0]", "$[2].x"]
SELECT JSON_SEARCH(@j3, 'all', 'abc', NULL, '$[2]'); -- "$[2].x"
SELECT JSON_SEARCH(@j3, 'all', '10'); -- "$[1][0].k"
SELECT JSON_SEARCH(@j3, 'all', '%b%'); -- ["$[0]", "$[2].x", "$[3].y"]
SELECT JSON_SEARCH(@j3, 'all', '%b%', NULL, '$[2]'); -- "$[2].x"
```

四,修改json值函数

1.JSON_ARRAY_APPEND 指定位置追加数组元素

```
-- JSON_ARRAY_APPEND(json_doc, path, val[, path, val] ...)
-- 在指定path的json array尾部追加val。如果指定path是一个json object，则将其封装成一个json array再追加。如果有参数为NULL，则返回N
SET @j4 = '["a", ["b", "c"], "d"]';
-- SELECT JSON_ARRAY_APPEND(@j4, '$[1][0]', 3); -- ["a", [["b", 3], "c"], "d"]
SET @j5 = '{"a": 1, "b": [2, 3], "c": 4}';
SELECT JSON_ARRAY_APPEND(@j5, '$.b', 'x'); -- {"a": 1, "b": [2, 3, "x"], "c": 4}
SELECT JSON_ARRAY_APPEND(@j5, '$.c', 'y'); -- {"a": 1, "b": [2, 3], "c": [4, "y"]}
SELECT JSON_ARRAY_APPEND(@j5, '$', 'z'); -- [{"a": 1, "b": [2, 3], "c": 4}, "z"]
```

2.JSON_ARRAY_INSERT 指定位置插入数组元素

```
-- JSON_ARRAY_INSERT(json_doc, path, val[, path, val] ...)
-- 在path指定的json array元素插入val，原位置及以右的元素顺次右移。如果path指定的数据非json array元素，则略过此val；如果指定的元素
SET @j6 = '["a", {"b": [1, 2]}, [3, 4]]';
SELECT JSON_ARRAY_INSERT(@j6, '$[1]', 'x'); -- ["a", "x", {"b": [1, 2]}, [3, 4]]
SELECT JSON_ARRAY_INSERT(@j6, '$[100]', 'x'); -- ["a", {"b": [1, 2]}, [3, 4], "x"]
SELECT JSON_ARRAY_INSERT(@j6, '$[1].b[0]', 'x'); -- ["a", {"b": ["x", 1, 2]}, [3, 4]]
SELECT JSON_ARRAY_INSERT(@j6, '$[0]', 'x', '$[3][1]', 'y'); -- ["x", "a", {"b": [1, 2]}, [3, "y", 4]]
```

3.JSON_INSERT 指定位置插入

```
-- JSON_INSERT(json_doc, path, val[, path, val] ...)
-- 在指定path下插入数据，如果path已存在，则忽略此val（不存在才插入）。
SET @j7 = '{ "a": 1, "b": [2, 3]}';
SELECT JSON_INSERT(@j7, '$.a', 10, '$.c', '[true, false]'); -- {"a": 1, "b": [2, 3], "c": "[true, false]"}
```

4.JSON_REPLACE 指定位置替换

```
-- JSON_REPLACE(json_doc, path, val[, path, val] ...)
-- 替换指定路径的数据，如果某个路径不存在则略过（存在才替换）。如果有参数为NULL，则返回NULL。
SELECT JSON_REPLACE(@j7, '$.a', 10, '$.c', '[true, false]'); -- {"a": 10, "b": [2, 3]}
```

5.JSON_SET 指定位置设置

```
-- JSON_SET(json_doc, path, val[, path, val] ...)
-- 设置指定路径的数据（不管是否存在）。如果有参数为NULL，则返回NULL。
SELECT JSON_SET(@j7, '$.a', 10, '$.c', '[true, false]'); -- {"a": 10, "b": [2, 3], "c": "[true, false]"}
```

6.JSON_MERGE 合并

```
-- JSON_MERGE(json_doc, json_doc[, json_doc] ...)
-- merge多个json文档。规则如下：
-- 如果都是json array，则结果自动merge为一个json array；
-- 如果都是json object，则结果自动merge为一个json object；
-- 如果有多种类型，则将非json array的元素封装成json array再按照规则一进行mege。
SELECT JSON_MERGE('[1, 2]', '[true, false]'); -- [1, 2, true, false]
SELECT JSON_MERGE('{"name": "x"}', '{"id": 47}'); -- {"id": 47, "name": "x"}
SELECT JSON_MERGE('1', 'true'); -- [1, true]
SELECT JSON_MERGE('[1, 2]', '{"id": 47}'); -- [1, 2, {"id": 47}]
```

7.JSON_REMOVE 指定位置移除

```
-- JSON_REMOVE(json_doc, path[, path] ...)
-- 移除指定路径的数据，如果某个路径不存在则略过此路径。如果有参数为NULL，则返回NULL。
SET @j8 = '["a", ["b", "c"], "d"]';
SELECT JSON_REMOVE(@j8, '$[1]'); -- ["a", "d"]
```

8.JSON_UNQUOTE 去"号

```
-- JSON_UNQUOTE(val)
-- 去掉val的引号。如果val为NULL，则返回NULL。
SELECT JSON_UNQUOTE("\"123\""); -- 123
```

五,返回json值属性的函数

1.JSON_DEPTH 深度

```
-- JSON_DEPTH(json_doc)
-- 获取json文档的深度。如果参数为NULL，则返回NULL。
-- 空的json array、json object或标量的深度为1。
SELECT JSON_DEPTH('{}'), JSON_DEPTH('[]'), JSON_DEPTH('true'); -- 1 1 1
SELECT JSON_DEPTH('[10, 20]'), JSON_DEPTH('[[], {}]'); -- 2 2
SELECT JSON_DEPTH('[10, {"a": 20}]'); -- 3
```

2.JSON_LENGTH 长度

```
-- JSON_LENGTH(json_doc[, path])
-- 获取指定路径下的长度。如果参数为NULL，则返回NULL。
-- 长度的计算规则：
-- 标量的长度为1；
-- json array的长度为元素的个数；
-- json object的长度为key的个数。
SELECT JSON_LENGTH('[1, 2, {"a": 3}]'); -- 3
SELECT JSON_LENGTH('{"a": 1, "b": {"c": 30}}'); -- 2
SELECT JSON_LENGTH('{"a": 1, "b": {"c": 30}}', '$.b'); -- 1
```

3.JSON_TYPE 类型

```
-- JSON_TYPE(json_val)
-- 获取json文档的具体类型。如果参数为NULL，则返回NULL。
select JSON_TYPE('[1,2]'); -- ARRAY
```

4.JSON_VALID 是否有效json格式

```
-- JSON_VALID(val)
-- 判断val是否为有效的json格式，是为1，不是为0。如果参数为NUL，则返回NULL。
SELECT JSON_VALID('{"a": 1}'); -- 1
SELECT JSON_VALID('hello'), JSON_VALID('"hello"'); -- 1
```

附录:

```
JSON_ARRAY 生成json数组
JSON_OBJECT 生成json对象
```

JSON_QUOTE 加"号
JSON_CONTAINS 指定数据是否存在
JSON_CONTAINS_PATH 指定路径是否存在
JSON_EXTRACT 查找所有指定数据
JSON_KEYS 查找所有指定键值
JSON_SEARCH 查找所有指定值的位置
JSON_ARRAY_APPEND 指定位置追加数组元素
JSON_ARRAY_INSERT 指定位置插入数组元素
JSON_INSERT 指定位置插入
JSON_REPLACE 指定位置替换
JSON_SET 指定位置设置
JSON_MERGE 合并
JSON_REMOVE 指定位置移除
JSON_UNQUOTE 去"号
JSON_DEPTH 深度
JSON_LENGTH 长度
JSON_TYPE 类型
JSON_VALID 是否有效json格式

函数名	描述
JSON_APPEND()（废弃的5.7.9）	JSON文件追加数据
JSON_ARRAY()	创建JSON数组
JSON_ARRAY_APPEND()	JSON文件追加数据
JSON_ARRAY_INSERT()	插入JSON数组
->	在评估路径返回JSON列值；相当于json_extract()。
JSON_CONTAINS()	是否包含特定对象的JSON文档路径
JSON_CONTAINS_PATH()	无论是JSON文件包含任何数据路径
JSON_DEPTH()	JSON文档的最大深度
JSON_EXTRACT()	从JSON文档返回数据
->>	在评估路径和结束引语结果返回JSON列值；相当于json_unquote（json_extract()）。
JSON_INSERT()	将数据插入到JSON文档
JSON_KEYS()	从JSON文件密钥数组
JSON_LENGTH()	在JSON文档中的元素数
JSON_MERGE()（废弃的5.7.22）	合并的JSON文件，保存重复键。不json_merge_preserve()的同义词
JSON_MERGE_PATCH()	合并的JSON文件，免去重复键的值
JSON_MERGE_PRESERVE()	合并的JSON文件，保存重复键
JSON_OBJECT()	创建JSON对象
JSON_PRETTY()	版画在人类可读的格式JSON文档，每个数组元素或对象成员打印在新的行中，缩进两个空格就其母。
JSON_QUOTE()	引用JSON文档
JSON_REMOVE()	从JSON文件中删除数据
JSON_REPLACE()	在JSON文件的值替换
JSON_SEARCH()	在JSON文件价值路径
JSON_SET()	将数据插入到JSON文档

函数名	描述
JSON_STORAGE_SIZE()	用于一个JSON文件的二进制表示形式存储空间；一个JSON柱，空间时使用的文档插入到任何部分更新之前
JSON_TYPE()	JSON值类型
JSON_UNQUOTE()	JSON值而言
JSON_VALID()	JSON值是否是有效的

mysql官方文档:https://dev.mysql.com/doc/refman/5.7/en/json-utility-functions.html