MySQL学习笔记（Day009：JSON）

MySQL学习

一. MySQL JSON类型

1. JSON介绍

- **JSON**（ J ava S cript O bject N otation）是一种轻量级的数据交换语言，并且是独立于语言的文本格式。
- 一些 NoSQL数据库 选择 JSON 作为其数据存储格式，比如：MongoDB、CouchDB等。
- MySQL 5.7.x 开始支持**JSON**数据类型。

> 官方文档(JSON类型)

2. JSON格式示例

```
--
-- 摘自 维基百科
--

{
     "firstName": "John",     -- Key : Value 格式
     "lastName": "Smith",
     "sex": "male",
     "age": 25,
     "address":               -- Key : Value ; 其中 Value 也是一个 Key-Value 的结构
     {
         "streetAddress": "21 2nd Street",
         "city": "New York",
         "state": "NY",
         "postalCode": "10021"
     },
     "phoneNumber":
     [
         {
           "type": "home",
           "number": "212 555-1234"
         },
         {
           "type": "fax",
           "number": "646 555-4567"
         }
     ]
}
```

3. JSON VS BLOB

- **JSON**
  - JSON数据可以做有效性检查;
  - JSON使得查询性能提升;
  - JSON支持部分属性索引，通过虚拟列的功能可以对JSON中的部分数据进行索引;
- **BLOB**
  - BLOB类型无法在数据库层做约束性检查;
  - BLOB进行查询，需要遍历所有字符串;
  - BLOB做只能做指定长度的索引;

> 5.7之前，只能把JSON当作BLOB进行存储。数据库层面无法对JSON数据做一些操作，只能由应用程序处理。

4.结构化和非结构化

- **结构化**
  - 二维表结构（行和列）
  - 使用SQL语句进行操作
- **非结构化**
  - 使用Key-Value格式定义数据，无结构定义
  - Value可以嵌套Key-Value格式的数据
  - 使用JSON进行实现

```
--
-- SQL创建User表
--
create table user (
    id bigint not null auto_increment,
    user_name varchar(10),
    age int,
    primary key(id)
);


#
# JSON定义的User表
#

db.user.insert({
    user_name:"tom",
    age:30
})

db.createCollection("user")
```

5. JSON操作示例

5.1 JSON入门

```
--
-- 创建带json字段的表
--
mysql> create table user (
    -> uid int auto_increment,
    -> data json,
    -> primary key(uid)
    -> );
Query OK, 0 rows affected (0.11 sec)

--
-- 插入json数据
--
mysql> insert into user values (
    -> null,  -- 自增长数据，可以插入null
    -> '{
    '> "name":"tom",
    '> "age":18,
    '> "address":"SZ"
    '> }'
    -> );
Query OK, 1 row affected (0.03 sec)

mysql> insert into user values (
    -> null,
    -> '{
    '> "name":"jim",
    '> "age":28,
    '> "mail":"jim@163.com"
    '> }'
    -> );
Query OK, 1 row affected (0.02 sec)

mysql> insert into user values ( null, "can you insert it?");  -- 无法插入，因为是JSON类型
ERROR 3140 (22032): Invalid JSON text: "Invalid value." at position 0 in value (or column) can you insert it?.  -- 这短话有单引号，但是渲染有问题，所以这里去掉了

mysql> select * from user;
+-----+-------------------------------------------------+
| uid | data                                            |
+-----+-------------------------------------------------+
|   1 | {"age": 18, "name": "tom", "address": "SZ"}     |  -- 这个json中有address字段
|   2 | {"age": 28, "mail": "jim@163.com", "name": "jim"} |  -- 这个json中有mail字段
+-----+-------------------------------------------------+
2 rows in set (0.00 sec)
```

5.2 JSON常用函数介绍

```
--
-- 使用json_extract提取数据
-- 原型 : JSON_EXTRACT(json_doc, path[, path] ...)
--
mysql> select json_extract('[10, 20, [30, 40]]', '$[1]');
+-------------------------------------------+
| json_extract('[10, 20, [30, 40]]', '$[1]') |
+-------------------------------------------+
| 20                                        |  -- 从list中抽取 下标 为1的元素（下标从0开始）
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> select
    -> json_extract(data, '$.name'),    -- 提起name字段的数据
    -> json_extract(data, '$.address')  -- 提取address字段的数据
    -> from user;
+-------------------------------+---------------------------------+
| json_extract(data, '$.name') | json_extract(data, '$.address') |
+-------------------------------+---------------------------------+
| "tom"                         | "SZ"                            |
| "jim"                         | NULL                            |  -- jim 没有address字段，填充了NULL
+-------------------------------+---------------------------------+
2 rows in set (0.00 sec)

--
-- json_object 将list(K-V对)封装成json格式
-- 原型 : JSON_OBJECT([key, val[, key, val] ...])
--
mysql> select json_object("name", "jery", "email", "jery@163.com", "age",33);
+-----------------------------------------------------------------+
| json_object("name", "jery", "email", "jery@163.com", "age",33) |
+-----------------------------------------------------------------+
| {"age": 33, "name": "jery", "email": "jery@163.com"}            |  -- 封装成了K-V对
+-----------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> insert into user values (
    -> null,
    -> json_object("name", "jery", "email", "jery@163.com", "age",33)  -- 进行封装
    -> );
Query OK, 1 row affected (0.03 sec)

mysql> select * from user;
+-----+-----------------------------------------------------------+
| uid | data                                                      |
+-----+-----------------------------------------------------------+
|   1 | {"age": 18, "name": "tom", "address": "SZ"}               |
|   2 | {"age": 28, "mail": "jim@163.com", "name": "jim"}         |
|   4 | {"age": 33, "name": "jery", "email": "jery@163.com"}      |
+-----+-----------------------------------------------------------+
3 rows in set (0.00 sec)


--
-- json_insert 插入数据
-- 原型 : JSON_INSERT(json_doc, path, val[, path, val] ...)
--
mysql> set @j = '{ "a": 1, "b": [2, 3]}';
Query OK, 0 rows affected (0.00 sec)

mysql> select json_insert(@j, '$.a', 10, '$.c', '[true, false]');
+-----------------------------------------------------+
| json_insert(@j, '$.a', 10, '$.c', '[true, false]') |
+-----------------------------------------------------+
| {"a": 1, "b": [2, 3], "c": "[true, false]"}         |  -- a还是=1，存在的被忽略，不受影响
+-----------------------------------------------------+  -- c之前不存在，则插入
1 row in set (0.00 sec)

mysql> update user set data = json_insert(data, "$.address_2", "BJ") where uid = 1;  -- 插入 addres_2
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from user;
+-----+-------------------------------------------------------------------+
| uid | data                                                              |
+-----+-------------------------------------------------------------------+
|   1 | {"age": 18, "name": "tom", "address": "SZ", "address_2": "BJ"} |  -- 增加了addres_2 : "BJ"
|   2 | {"age": 28, "mail": "jim@163.com", "name": "jim"}               |
|   4 | {"age": 33, "name": "jery", "email": "jery@163.com"}            |
+-----+-------------------------------------------------------------------+
3 rows in set (0.00 sec)

--
-- json_merge 合并数据并返回。注意：原数据不受影响
-- 原型 : JSON_MERGE(json_doc, json_doc[, json_doc] ...)
--
mysql> select json_merge('{"name": "x"}', '{"id": 47}');    -- 原来有两个JSON
+-------------------------------------------+
| json_merge('{"name": "x"}', '{"id": 47}') |
+-------------------------------------------+
| {"id": 47, "name": "x"}                   |  -- 合并多个JSON
+-------------------------------------------+
1 row in set (0.00 sec)

mysql> select
    -> json_merge(
    -> json_extract(data, '$.address'),      -- json 1
    -> json_extract(data, '$.address_2'))    -- jons 2
    -> from user where uid = 1;
+-------------------------------------------------------------------------------+
| json_merge( json_extract(data, '$.address'), json_extract(data, '$.address_2')) |
+-------------------------------------------------------------------------------+
| ["SZ", "BJ"]                                                                  |  -- 合并成一个json
+-------------------------------------------------------------------------------+
1 row in set (0.00 sec)

--
-- json_array_append 追加数据
-- 原型 : JSON_ARRAY_APPEND(json_doc, path, val[, path, val] ...)
-- json_append 在5.7.9 中重命名为 json_array_append
--
mysql> set @j = '["a", ["b", "c"], "d"]';     -- 下标为1的元素中只有["b", "c"]
Query OK, 0 rows affected (0.00 sec)

mysql> select json_array_append(@j, '$[1]', 1);
+----------------------------------+
| json_array_append(@j, '$[1]', 1) |
+----------------------------------+
| ["a", ["b", "c", 1], "d"]        |    --  现在插入了 数字 1
+----------------------------------+
1 row in set (0.00 sec)
mysql> update user set data = json_array_append(
    -> data,
    -> '$.address',
    -> json_extract(data, '$.address_2'))
    -> where uid = 1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from user;
+-----+-------------------------------------------------------------------------------+
| uid | data                                                                          |
+-----+-------------------------------------------------------------------------------+
|   1 | {"age": 18, "name": "tom", "address": ["SZ", "BJ"], "address_2": "BJ"} | --address_2追加到address
|   2 | {"age": 28, "mail": "jim@163.com", "name": "jim"}                   |
|   4 | {"age": 33, "name": "jery", "email": "jery@163.com"}                |
+-----+-------------------------------------------------------------------------------+
3 rows in set (0.00 sec)


--
-- json_remove 从json记录中删除数据
-- 原型 : JSON_REMOVE(json_doc, path[, path] ...)
--
mysql> set @j = '["a", ["b", "c"], "d"]';
Query OK, 0 rows affected (0.00 sec)

mysql> select json_remove(@j, '$[1]');
+-------------------------+
| json_remove(@j, '$[1]') |
+-------------------------+
| ["a", "d"]              |  -- 删除了下标为1的元素["b", "c"]
+-------------------------+
1 row in set (0.00 sec)

mysql> update user set data = json_remove(data, "$.address_2") where uid = 1;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from user;
+-----+-----------------------------------------------------------+
| uid | data                                                      |
+-----+-----------------------------------------------------------+
|   1 | {"age": 18, "name": "tom", "address": ["SZ", "BJ"]}  |  -- address_2 的字段删除了
|   2 | {"age": 28, "mail": "jim@163.com", "name": "jim"}    |
|   4 | {"age": 33, "name": "jery", "email": "jery@163.com"} |
+-----+-----------------------------------------------------------+
3 rows in set (0.00 sec)
```

> 官方文档(JSON函数)

5.3 JSON创建索引

JSON 类型数据本身 无法直接 创建索引，需要将需要索引的 JSON数据 重新 生成虚拟列(Virtual Columns) 之后，对 该列 进行 索引

> 官方文档–JSON创建索引

- **新建表时创建JSON索引**

```
mysql> create table test_inex_1(
    -> data json,
    -> gen_col varchar(10) generated always as (json_extract(data, '$.name')),  -- 抽取data中的name， 生成新的一列，名字为gen_col
    -> index idx (gen_col)  -- 将gen_col 作为索引
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> show create table test_index_1;
-- -----省略表格线-----
| test_index_1 | CREATE TABLE `test_index_1` (
  `data` json DEFAULT NULL,
  `gen_col` varchar(10) GENERATED ALWAYS AS (json_extract(data, '$.name')) VIRTUAL,
  KEY `idx` (`gen_col`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
-- -----省略表格线-----
1 row in set (0.00 sec)

mysql> insert into test_index_1(data) values ('{"name":"tom", "age":18, "address":"SH"}');
Query OK, 1 row affected (0.04 sec)

mysql> insert into test_index_1(data) values ('{"name":"jim", "age":28, "address":"SZ"}');
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_index_1;
+-----------------------------------------+---------+
| data                                    | gen_col |
+-----------------------------------------+---------+
| {"age": 18, "name": "tom", "address": "SH"} | "tom"   |
| {"age": 28, "name": "jim", "address": "SZ"} | "jim"   |
+-----------------------------------------+---------+
2 rows in set (0.00 sec)

mysql> select json_extract(data,"$.name") as username from test_index_1 where gen_col="tom";  -- 如果这样做，为空，原因如下
Empty set (0.00 sec)

mysql> select hex('"');
+----------+
| hex('"') |
+----------+
| 22       |  -- 双引号的 16进制
+----------+
1 row in set (0.00 sec)

mysql> select hex(gen_col) from test_index_1;
+--------------+
| hex(gen_col) |
+--------------+
| 226A696D22   |  -- 双引号本身也作为了存储内容
| 22746F6D22   |
+--------------+
2 rows in set (0.00 sec)

mysql> select json_extract(data,"$.name") as username from test_index_1 where gen_col='"tom"';  -- 使用'"tome"',用单引号括起来
+----------+
| username |
+----------+
| "tom"    |  -- 找到了对应的数据
+----------+
1 row in set (0.00 sec)

mysql> explain select json_extract(data,"$.name") as username from test_index_1 where gen_col='"tom"'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test_index_1
   partitions: NULL
         type: ref
possible_keys: idx    -- 使用了 key idx
          key: idx
      key_len: 43
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0.00 sec)

---
--- 建立表的时候去掉双引用
---

mysql> create table test_index_2 (
    -> data json,
    -> gen_col varchar(10) generated always as (
    ->    json_unquote(    -- 使用json_unquote函数进行去掉双引号
    ->             json_extract(data, "$.name")
    ->    )),
    -> key idx(gen_col)
    -> );
Query OK, 0 rows affected (0.13 sec)

mysql> show create table test_index_2;
-- -----省略表格线-----
| test_index_2 | CREATE TABLE `test_index_2` (
  `data` json DEFAULT NULL,
  `gen_col` varchar(10) GENERATED ALWAYS AS (json_unquote(
            json_extract(data, "$.name")
   )) VIRTUAL,
  KEY `idx` (`gen_col`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
-- -----省略表格线-----
1 row in set (0.00 sec)

mysql> insert into test_index_2(data) values ('{"name":"tom", "age":18, "address":"SH"}');
Query OK, 1 row affected (0.03 sec)

mysql> insert into test_index_2(data) values ('{"name":"jim", "age":28, "address":"SZ"}');
Query OK, 1 row affected (0.02 sec)

mysql> select json_extract(data,"$.name") as username from test_index_2 where gen_col="tom";  -- 未加单引号
+----------+
| username |
+----------+
| "tom"    |  -- 可以找到数据
+----------+
1 row in set (0.00 sec)

mysql> explain select json_extract(data,"$.name") as username from test_index_2 where gen_col="tom"\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test_index_2
   partitions: NULL
         type: ref
possible_keys: idx   -- 使用了 key idx
          key: idx
      key_len: 43
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0.00 sec)
```

- **修改已存在的表创建JSON索引**

```
--
-- 使用之前的user表操作
--
mysql> show create table user;
-- -----省略表格线-----
| user  | CREATE TABLE `user` (
  `uid` int(11) NOT NULL AUTO_INCREMENT,
  `data` json DEFAULT NULL,
  PRIMARY KEY (`uid`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 |
-- -----省略表格线-----
1 row in set (0.00 sec)

mysql> select * from user;
+-----+-------------------------------------------------------+
| uid | data                                                  |
+-----+-------------------------------------------------------+
|   1 | {"age": 18, "name": "tom", "address": ["SZ", "BJ"]}  |
|   2 | {"age": 28, "mail": "jim@163.com", "name": "jim"}    |
|   4 | {"age": 33, "name": "jery", "email": "jery@163.com"} |
+-----+-------------------------------------------------------+

mysql> alter table user
    -> add user_name varchar(32)
    -> generated always as (json_extract(data,"$.name")) virtual;
Query OK, 0 rows affected (0.05 sec)
Records: 0  Duplicates: 0  Warnings: 0
-- virtual 关键字是不将该列的字段值存储，对应的是stored

mysql> select user_name from user;
+-----------+
| user_name |
+-----------+
| "tom"     |
| "jim"     |
| "jery"    |
+-----------+
3 rows in set (0.00 sec)

mysql> alter table user add index idx(user_name);
Query OK, 0 rows affected (0.13 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> select * from user where user_name='"tom"';  -- 加单引号
+-----+-------------------------------------------------------+-----------+
| uid | data                                                  | user_name |
+-----+-------------------------------------------------------+-----------+
|   1 | {"age": 18, "name": "tom", "address": ["SZ", "BJ"]} | "tom"     |
+-----+-------------------------------------------------------+-----------+
1 row in set (0.00 sec)

mysql> explain select * from user where user_name='"tom"'\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: user
   partitions: NULL
         type: ref
possible_keys: idx   -- 使用了 key idx
          key: idx
      key_len: 131
          ref: const
         rows: 1
     filtered: 100.00
        Extra: NULL
1 row in set, 1 warning (0.00 sec)

mysql> show create table user;
-- -----省略表格线-----
| user  | CREATE TABLE `user` (
  `uid` int(11) NOT NULL AUTO_INCREMENT,
  `data` json DEFAULT NULL,
  `user_name` varchar(32) GENERATED ALWAYS AS (json_extract(data,"$.name")) VIRTUAL,
  `user_name2` varchar(32) GENERATED ALWAYS AS (json_extract(data,"$.name")) VIRTUAL,
  PRIMARY KEY (`uid`),
  KEY `idx` (`user_name`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 |
-- -----省略表格线-----
1 row in set (0.00 sec)
```

二. 附录

```
--
-- 老师演示JSON的SQL
--
drop table if exists User;

CREATE TABLE User (
    uid BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(32) NOT NULL,
    email VARCHAR(256) NOT NULL,
    address VARCHAR(512) NOT NULL,
    UNIQUE KEY (name),
    UNIQUE KEY (email)
);

INSERT INTO User VALUES (NULL,'David','david@gmail','Shanghai ...');
INSERT INTO User VALUES (NULL,'Amy','amy@gmail','Beijing ...');
INSERT INTO User VALUES (NULL,'Tom','tom@gmail','Guangzhou ...');

SELECT * FROM User;

ALTER TABLE User ADD COLUMN address2 VARCHAR(512) NOT NULL;
ALTER TABLE User ADD COLUMN passport VARCHAR(64) NOT NULL;

DROP TABLE IF EXISTS UserJson;

CREATE TABLE UserJson(
    uid BIGINT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    data JSON
);

truncate table UserJson;

insert into UserJson
SELECT
    uid,JSON_OBJECT('name',name,'email',email,'address',address) AS data
FROM
    User;

SELECT * FROM UserJson;

SELECT uid,JSON_EXTRACT(data,'$.address2') from UserJson;

UPDATE UserJson
set data = json_insert(data,"$.address2","HangZhou ...")
where uid = 1;

SELECT JSON_EXTRACT(data,'$.address[1]') from UserJson;

select json_merge(JSON_EXTRACT(data,'$.address') ,JSON_EXTRACT(data,'$.address2'))
from UserJson;

begin;
UPDATE UserJson
set data = json_array_append(data,"$.address",JSON_EXTRACT(data,'$.address2'))
where JSON_EXTRACT(data,'$.address2') IS NOT NULL AND uid >0;
select JSON_EXTRACT(data,'$.address') from UserJson;
UPDATE UserJson
set data = JSON_REMOVE(data,'$.address2')
where uid>0;
commit;
```