JSON

ClustrixDB now provides Beta support for the JSON data type, including for SQL, many functions, and indexing.

JSON DDL

ClustrixDB supports the JSON data type, similar to the support included in MySQL 5.7. ClustrixDB stores JSON in a native JSON format, which allows for easy retrieval.

```
sql> create table files (id int primary key auto_increment, doc json);
```

JSON Functions

ClustrixDB supports the following JSON functions:

- JSON_ARRAY()
- JSON_CONTAINS_PATH()
- JSON_DEPTH()
- JSON_EXTRACT()
- -> (column path operator)
- ->> (inline path operator)
- JSON_KEYS()
- JSON_LENGTH()
- JSON_OBJECT()
- JSON_QUOTE()
- JSON_SEARCH()
- JSON_TYPE()
- JSON_UNQUOTE()
- JSON_VALID()

Indexing JSON

This example will use the following data and queries:

```
sql> insert into files (doc) values ('{"foo": {"bar": 1}, "baz": [1,2,3,4]}');
sql> insert into files (doc) values ('{"foo": {"bar": 2}, "baz": [3,4,5]}');

sql> select json_extract(doc, '$.baz') from files where json_extract(doc, '$.foo.
bar') = 1;
+----------------------------+
| json_extract(doc, '$.baz') |
+----------------------------+
| [1, 2, 3, 4]               |
+----------------------------+
1 row in set (0.01 sec)
```

ClustrixDB supports indexing raw JSON column values:

```
sql> alter table files add column foobar json;
```

But also, you can index a JSON attribute by creating a generated column and creating an index on that:

```
sql> alter table files add column foobar2 json generated always as (json_extract(doc, '$.foo.bar'));
sql> alter table files add index foobar2_id (foobar2);
```

This generated column + index can now be used to filter for values:

```
sql> explain select foobar2 from files where foobar2 = 1;
+-----------------------------------------------------+-----------+-----------+
| Operation                                           | Est. Cost | Est. Rows |
+-----------------------------------------------------+-----------+-----------+
| index_scan 1 := files.foobar2_id, foobar2 = param(0) |      4.61 |      1.01 |
+-----------------------------------------------------+-----------+-----------+
1 row in set (0.00 sec)
```

Note: ClustrixDB can not yet translate expressions in queries to match their equivalent indexed generated column. This SQL statement is equivalent to the one above, but cannot make use of its associated index:

```
sql> explain select json_extract(doc, '$.baz') from files where json_extract(doc, '$.foo.
bar') = 1;
+-----------------------------------------------------+-----------+-----------+
| Operation                                           | Est. Cost | Est. Rows |
+-----------------------------------------------------+-----------+-----------+
| stream_combine                                      |     13.54 |      0.91 |
|   compute expr0 := json_extract(1.doc, param(0))    |     12.45 |      0.91 |
|     filter (json_extract(1.doc, param(2)) = param(1)) |     12.43 |      0.91 |
|       index_scan 1 := files.__idx_files__PRIMARY    |     12.41 |      1.01 |
+-----------------------------------------------------+-----------+-----------+
4 rows in set (0.01 sec)
```

Caveats for JSON

When using JSON, the Query Results Cache (QRC) must be disabled.

```
sql> set global qrc_enabled = false;
```

ClustrixDB does not support:

- JSON_ARRAY_APPEND
- JSON_ARRAY_INSERT
- JSON_INSERT
- JSON_MERGE
- JSON_REMOVE
- JSON_REPLACE
- JSON_SET
- JSON_CONTAINS

Differences from MySQL

Inserting JSON

- ClustrixDB stores timestamps inside JSON as DATETIME.
- MySQL 5.7 sometimes limits the size of TIME values embedded in a JSON to 32 hours. ClustrixDB allows the full range of TIME values to be used.
- ClustrixDB converts input to JSON_QUOTE/UNQUOTE to strings, where MySQL only accepts string input.

Querying JSON

- ClustrixDB uses the same ordering for both the comparison operators and ORDER BY. MySQL uses different orderings for comparisons and ORD ER BY.
- ClustrixDB performs a depth first search for JSON_EXTRACT and JSON_SEARCH. MySQL performs a breadth first search, which results in a different ordering of results.

Due to some of the differences between ClustrixDB and MySQL, if you are replicating to MySQL using SBR, you may encounter errors.