USING JSON WITH CASSANDRA

Cassandra provides support for JSON. You can, of course, store JSON text into Cassandra text columns. But that is not what we mean here. Here we mean you can use JSON to write to Cassandra tables.

We illustrate with an example. First create a keyspace.

```
create keyspace json with REPLICATION = {'class' : 'SimpleStrategy',
'replication_factor' : 2 };
```

(This article is part of our Cassandra Guide. Use the right-hand menu to navigate.)

Simple JSON Example

Let's make a table with three columns. Then write data into it using the JSON keyword. That tells Cassandra to maps the JSON values to the columns names, just as you would expect with JSON.

```
create table books (isbn text primary key, title text, publisher text);

insert into books JSON '{"isbn": "123",
"title": "The Magic Mountain",
 "publisher": "Knofp"}';
```

Then we list the data. It looks like regular columnar data and not JSON. Because it's just regular data. We simply used JSON to add it, which is useful since data is often represented as JSON.

```
select * from books;
```

```
 isbn | publisher | title
------+-----------+--------------------
  123 |     Knofp | The Magic Mountain
```

Nested JSON

Here is a more complicated example. We have a customer record with a sales type that is of type list. Obviously a customer would have multiple sales. So this is a good way to represent that.

First we create the sale type. Then we create the customers table. We have to use the FROZEN keyword as each sales transaction cannot be changed, i.e., it is immutable. If you want to change a sales transaction you would have to delete it and then add it back.

```
use json;

CREATE type json.sale ( id int, item text, amount int );

CREATE TABLE json.customers ( id int  PRIMARY KEY, name text, balance int,
sales list> );
```

Now add values. Since the sales items are a list we can add more than one.

```
INSERT INTO json.customers (id, name, balance, sales)
VALUES (123, 'Greenville Hardware', 700,
 ) ;
```

Now list the items. Note that there is no question mark around the key value in the jSON. This is because it is a JSON object.

```
select * from customers;

 id  | balance | name                | sales
-----+---------+---------------------+-------------------------------------
----------------------------------------
 123 |     700 | Greenville Hardware |
```

If you want to convert it to a JSON string then use toJson()

```
select id, toJson(sales) from customers;

 id  | system.tojson(sales)
-----+-------------------------------------------------------------------------
---------------------
 123 |
```