Relax with Apache Couch DB

What is it?

- **C**luster **O**f **U**nreliable **C**ommodity **H**ardware
- Open source database created with an emphasis on ease of use. It “completely embraces the web.”
- NoSQL
- Document Store – JSON used to store data
- Query with JavaScript using MapReduce
- HTTP as an API

Data Model - Overview

- Web oriented, document based data system
  - Documents are stand alone, complex stores of information.
  - Documents can store files, functions, etc. but cannot store references
  - Each document has a URI – It is a resource and can be interacted with via HTTP according to the REST architecture.

Data Model – Basics

- Key-Value Construct is fundamental

Basic Implementation:

“name”: “John Doe”

Values can be any atomic data type: integer, floating-point, Boolean, and character strings. Non-string values do not need to be surrounded by quotations

“birthday” : 2020

Data Model – Complex

- Can create an *object* which is an unordered set of key-value pairs.

```
{“lastName”: “Doe”, “firstName”: “John”}

“customer” :{
        “lastName”: “Doe”
        “firstName”: “John”
        “birthday”: 2020
}
```

- Keys can only appear once and types may be distinct.
- An object can be the complex value of a key-value pair.

Data Model - Complex

- An array allows for an ordered collection of values that do not need to be of the same type:

  “things”: [“orange”, true, 42, 1.5, {“person”: “John Doe”}]

Data Model - Complex

- A *document* is an object that can be a nesting of array and objects.

```
{
    “lastPurchase”: “orange”
    “recentPurchases”: [“orange”, “apple”, “monster truck”]

        “customer” :{
            “lastName”: “Doe”
            “firstName”: “John”
            “birthday”: 2020}
}
```

Sample Interaction

- Accessing Server Information:
  - curl http://127.0.0.1:5984/

  Response:

```
{
  "couchdb": "Welcome",
  "version":"1.1.0"
}
```

Sample Interaction

- Create a DB named foo:

```
curl -X PUT http://127.0.0.1:5984/foo
```

Response:

```
{"ok": true}
```

Sample Interaction

- Put a resource into foo:

```
curl -X PUT http://127.0.0.1:5984/foo/Doc\
        -d '{“foo”:”bar”}'
```

Response:

```
{"ok":true, "id":"Doc", "rev":"<1-somenumbers>"}
```

Sample Interaction

- Update a resource in foo:
  - curl -X PUT http://127.0.0.1:5984/foo/Doc -d '{“_rev”:”<1-somenumbers>", “foo”:”bar”, “howMuch”:”totally”}

Response:

{"ok":true, "id":"Doc", "rev":"<1-somenumbers>"}

Sample Interaction

- Create a DB named foo again:

curl -X PUT http://127.0.0.1:5984/foo

Response:


```
{
  "error":"file_exists",
  "reason":"The database could not be created, the file already
   exists."
}
```

Sample Interaction

- Retrieve information about foo:

curl http://127.0.0.1:5984/foo

Response:


```
{
  "db_name": "foo",
  "doc_count": 0,
  "doc_del_count": 0,
  "update_seq": 0,
  "purge_seq": 0,
  "compact_running": false,
  "disk_size": 79,
  "instance_start_time": "1272453873691070",
  "disk_format_version": 5
}
```

Sample Interaction

- Delete foo:

```
url -X DELETE http://127.0.0.1:5984/foo
```

Response:

```
{"ok": true}
```

Sample Interaction

- Create a document and request a document id:

```
curl -X POST -H "Content-Type: application/json" --data \
'{ "text" : "Foo", "rating": 5 }' \
http://127.0.0.1:5984/foo
```

Response:

```
{
  "ok": true,
  "id": "123BAC",
  "rev": "946B7D1C"
}
```

Document Management

- Each document has an id and a revision number.
  - When you update a document the id remains the same but revision number will change.
- Validation functions can be implemented over a collection so that any document added or updated will be checked.
- Create new key-document collections called views with MapReduce.
- Documents can be replicated across instances of CouchDB

Creating Views

- Views result from a MapReduce sequence that returns a list of (key, value) pairs.
- They are materialized and indexed by key with a B+tree.
- We are essentially creating an index on some key.
- CouchDB is able to index views and update the indices as they are added, deleted, and updated.

Replication

- Supports distributed architecture
- Designed with two way replication / synchronization.
- Supports off-line operation.
  - “Eventual” synchronization once back on-line.
- Allows for multiple replicas to work on individual copies of the same data and sync at a a later time.
  - This allows for replication to devices like smartphones

ACID Semantics

- Atomicity, Consistency, Isolation, Durability
- Multi-Version Concurrency Control implements ACID semantics.
- This allows for high volume of concurrent readers and writers
    - Important for function of highly distributed architecture.

References

CouchDB: The Definitive Guide:

http://guide.couchdb.org/

CouchDB Wikipedia Page:

http://en.wikipedia.org/wiki/CouchDB