Getting Started with PouchDB - Part 5

The last four blog posts have introduced you to working with a PouchDB database. You learned to modify documents one at time, in bulk, and learned to query the data within that database. In this fifth part of our ongoing series on PouchDB, you learn to use reduce queries to provide summary data such as the sum or average cost data, minimum and maximum, and how to calculate an average of cost data.

Reduce Queries

A reduce query's purpose is to iterate over a set of elements in your database and reduce all values to single value. For example, you might want to calculate the sum of all costs, get the minimum cost or the maximum cost. These are all the things a reduce query can accomplish for you. You use the query() method to perform these reduce queries.

Count All Documents

The first example is to simply get a count of all documents. This is almost unnecessary, as I have already shown you a couple of ways to get the total number of documents in your database, but just for completeness, here is the code using a reduce query.

```
function countDocuments() {
  pouchDBSamplesCommon.hideMessageAreas();
  db.query({
    map: function (doc) {
      emit(doc._id);
    },
    reduce: '_count'
  },
    {
      reduce: true
    }).then(function (response) {
      pouchDBSamplesCommon.displayJSON(response);
    }).catch(function (err) {
      pouchDBSamplesCommon.displayMessage(err);
    });
}
```

The key to this code is to set the *reduce* property to the built-in string '**_count**'. The map function is just like what you saw in the last blog post. You write whatever code you want within the function to determine what to count. In the code above, no criteria is specified, so it gives you a total count of documents. If you wrapped an if statement around the emit() function, you can count a specific subset of documents. You don't even need to emit a property as the first parameter. You can pass a null as the first parameter, or even pass no parameters to the emit() function.

After running this query, the result returned looks like the following:

```
{
  "rows": [
    {
      "value": 10,
      "key": null
    }
  ]
}
```

Sum of All Costs

To sum a set of numbers in one of the properties across a set of documents, you write an if statement to restrict the number of documents that are to be summed. In the example below, you check for only those documents that have a *cost* property. When you call the emit() function, pass anything for the first parameter, and the second parameter is the property you wish to calculate the sum for. Set the reduce property to the built-in string '**_sum**'.

```
function sumCost() {
  pouchDBSamplesCommon.hideMessageAreas();
  db.query({
    map: function (doc) {
      if (doc.cost) {
        emit(null, doc.cost);
      }
    },
    reduce: '_sum'
  },
    {
      reduce: true
    }).then(function (response) {
      pouchDBSamplesCommon.displayJSON(response);
    }).catch(function (err) {
      pouchDBSamplesCommon.displayMessage(err);
    });
}
```

The output from running this query, looks like the following:

```
{
  "rows": [
    {
      "value": 360,
      "key": null
    }
  ]
}
```

Minimum and Maximum Costs

Another built-in reduce function is called '**_stats**'. This reduce query returns five pieces of data for a given numeric property in your documents; sum, min, max, count and sumsqr. This function is great if you need both a sum and an average of your *cost* property as using '**_stats**' is only one call as opposed to two. Look at the code below to see how to call the '**_stats**' reduce function.

```
function statsCost() {
  pouchDBSamplesCommon.hideMessageAreas();
  db.query({
    map: function (doc) {
      if (doc.cost) {
        emit(null, doc.cost);
      }
    },
    reduce: '_stats'
  },
    {
      reduce: true
    }).then(function (response) {
      pouchDBSamplesCommon.displayJSON(response);
    }).catch(function (err) {
      pouchDBSamplesCommon.displayMessage(err);
    });
}
```

The resulting output from the '_stats' reduce function looks like the following:

```
{
  "rows": [
    {
      "value": {
        "sum": 360,
        "min": 25,
        "max": 100,
        "count": 5,
        "sumsqr": 29100
      },
      "key": null
    }
  ]
}
```

NOTE: the *sumsqr* property is the sum of the squared elements. Each cost value is multiplied by itself, then all the squared values are summed to create the *sumsqr* property.

Average Costs

There is no average reduce function, but you have everything you need in the '**_stats**' function, namely, the *count* and *sum* properties. From these two properties you can calculate the average by dividing the *count* by the *sum* as shown in the code below.

```
function avgCost() {
  pouchDBSamplesCommon.hideMessageAreas();
  db.query({
    map: function (doc) {
      if (doc.cost) {
        emit(null, doc.cost);
      }
    },
    reduce: '_stats'
  },
    {
      reduce: true
    }).then(function (response) {
      // Calculate the average
      var avg = response.rows[0].value.sum /
response.rows[0].value.count;
      // Display the average
      pouchDBSamplesCommon.displayMessage('The average is ' + avg);
    }).catch(function (err) {
      pouchDBSamplesCommon.displayMessage(err);
    });
}
```

In the above code, after the average of the cost property is calculated, the average is passed to the displayMessage() method to be displayed. The result looks like this.

```
The average is 72
```

Average Costs on Filtered Data

Using the same reduce function above, you can also filter the data to get an average of cost data for only a subset of documents. In the example below, you can use the *startkey* and *endkey* properties to restrict the documents to only those that whose *_id* property starts with the letter 'C'. In this function it is important to emit the *_id* property value as that is what is used for filtering using *startkey* and *endkey*.

```
function avgCostForFilteredData() {
  pouchDBSamplesCommon.hideMessageAreas();
  db.query({
    map: function (doc) {
      if (doc.cost) {
        emit(doc._id, doc.cost);
      }
    },
    reduce: '_stats'
  },
    {
      startkey: 'C',
      endkey: 'C\ufff0',
      reduce: true
    }).then(function (response) {
      // Calculate the average
      var avg = response.rows[0].value.sum /
response.rows[0].value.count;
      // Display the average
      pouchDBSamplesCommon.displayMessage('The average is ' + avg);
    }).catch(function (err) {
      pouchDBSamplesCommon.displayMessage(err);
    });
}
```

The output from this method is displayed in the browser as the following:

```
The average is 87.5
```

Summary

In this blog post you learned to reduce a set of values from your documents into a single value such as the count, sum, average, minimum or maximum values. If all you want is a count, it is recommended to just use the allDocs() method as it is optimized to use the built-in index for your documents, whereas the query() method is not optimized. In the next blog post, you learn how to work with nested documents.

Sample Code

You can download the complete sample code at my website. http://www.pdsa.com/downloads. Choose "PDSA/Fairway Blog", then "Getting Started with PouchDB - Part 5" from the drop-down.