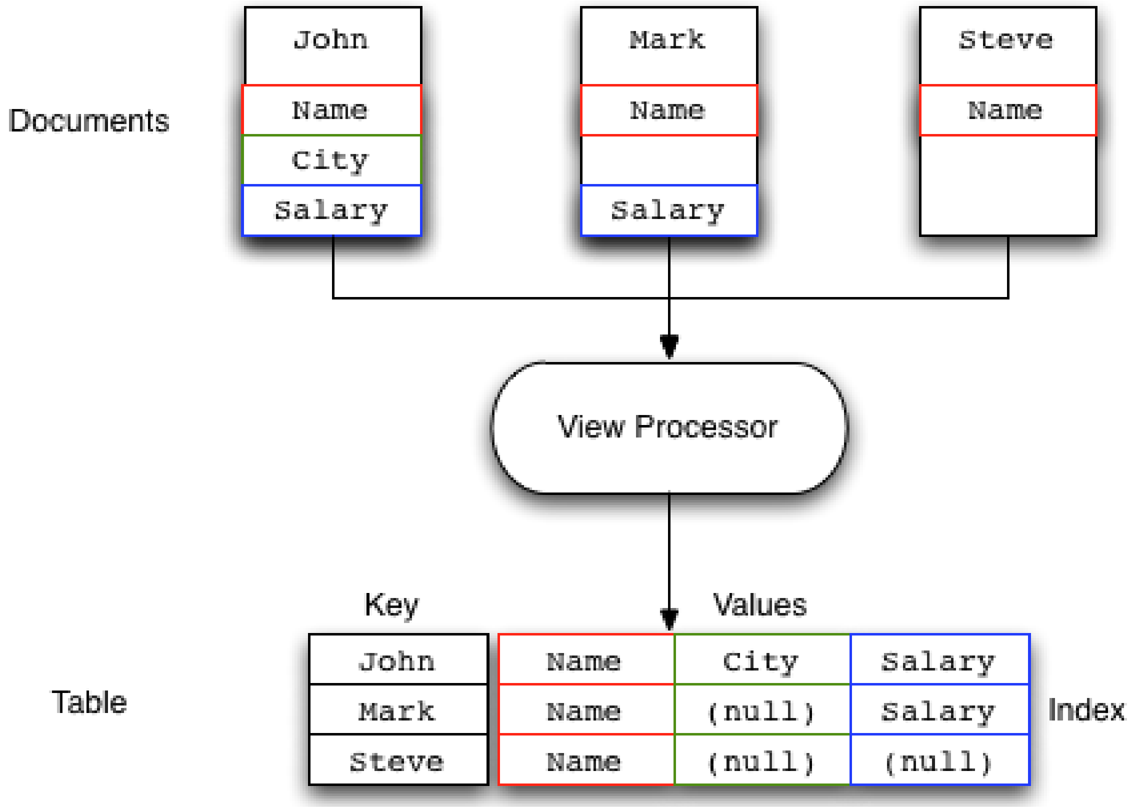
HP

Indexing - Views

What are Views?

- Extract fields from JSON documents and produce an index of the selected information



HP

Facts about Map/Reduce

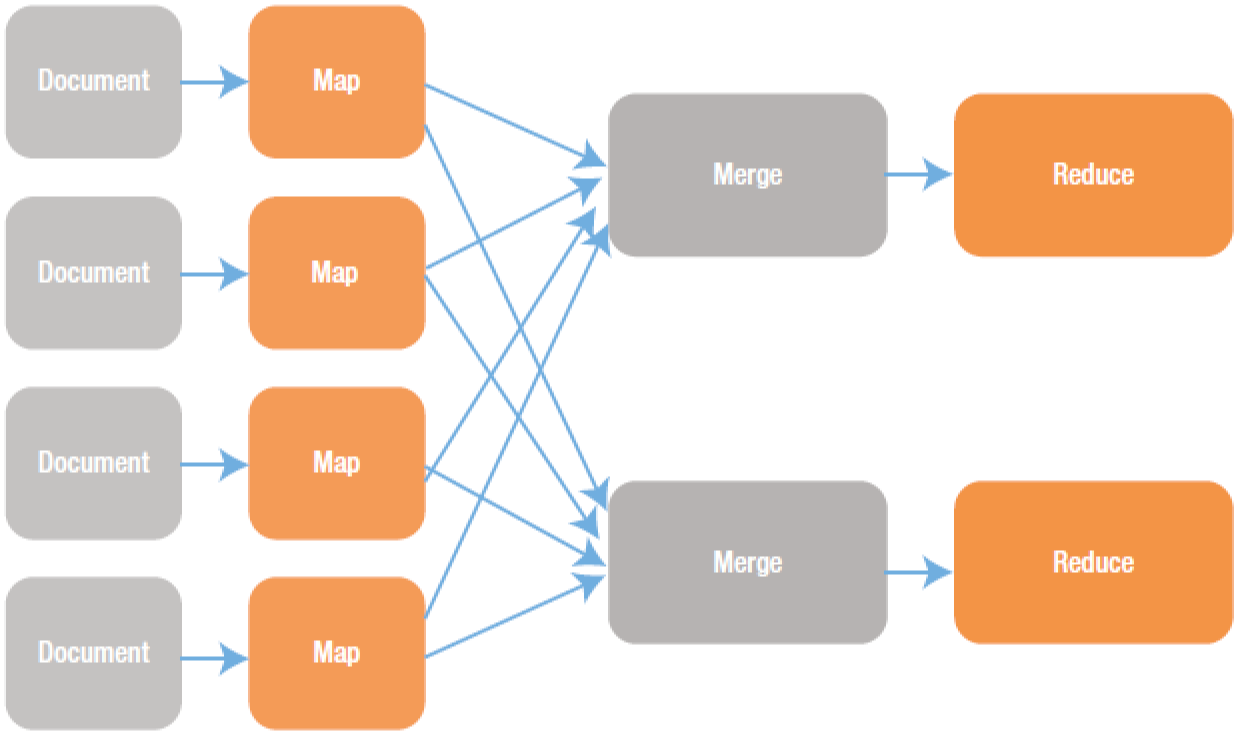
1. Programming paradigm, popularized and patented by Google
2. Great for parallel jobs
3. No Joins between documents
4. In Couchdatabase: Map/Reduce in JavaScript (default)
5. Also Possible with other languages

Workflow

1. Map function builds a list of key/value pairs
2. Reduce function reduces the list (to a single Value)

HP

MapReduce workflow



HP

Simple Map Example

- A List of Cars

Id: 1	Id: 2	Id: 3	Id: 4	Id: 5
make: Audi	make: Audi	make: VW	make: VW	make: VW
model: A3	model: A4	model: Golf	model: Golf	model: Polo
year: 2000	year: 2009	year: 2009	year: 2008	year: 2010
price: 5.400	price: 16.000	price: 15.000	price: 9.000	price: 12.000

- Step 1: Make a list, ordered by Price

```
Function(doc) {
  emit (doc.price, doc.id);
}
```

Key

Value

- Step 2: Result:

Key	**, Value**
5.400	1
9.000	4
12.000	5
15.000	3
16.000	2

Querying Maps

- Original Map

Key	, Value
5.400	, 1
9.000	, 4
12.000	, 5
15.000	, 3
16.000	, 2

- startkey=10.000 & endkey=15.500

Key	, Value
12.000	, 5
15.000	, 4

All keys from 10.000 to < 15.500

- key=10.000

Key	, Value

Exact key, so no result

- endkey=10.000

Key	, Value
5.400	, 1

All keys, less than 10.000

HP

Map Function

- Has one document as input
- Can emit all JSON-Types as key and value:

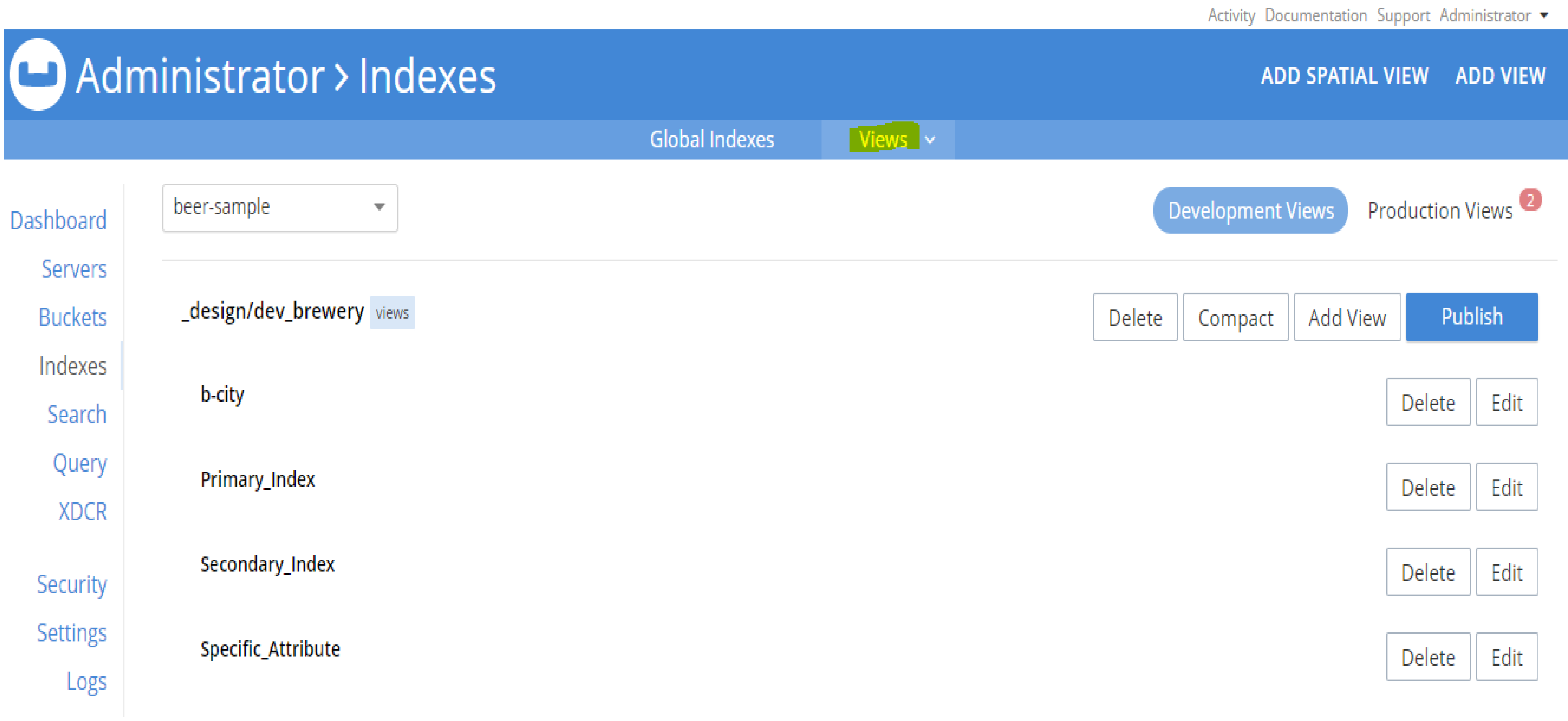
  - Special Values: null, true, false
  - Numbers: 1e-17, 1.5, 200
  - Strings : “+“, “1“, “Ab“, “Audi“
  - Arrays: [1], [1,2], [1,“Audi“,true]
  - Objects: {“price“:1300,“sold“:true}
- Results are ordered by key (or revers)
  (order with mixed types: see above)
- In Couchbase: Each result has also the doc._id

```
{"total_rows":5,"offset":0,
"rows":[
{"id":"1","key":"Audi","value":1},
{"id":"2","key":"Audi","value":1},
{"id":"3","key":"VW","value":1},
{"id":"4","key":"VW","value":1},
{"id":"5","key":"VW","value":1} ]}
```

HP

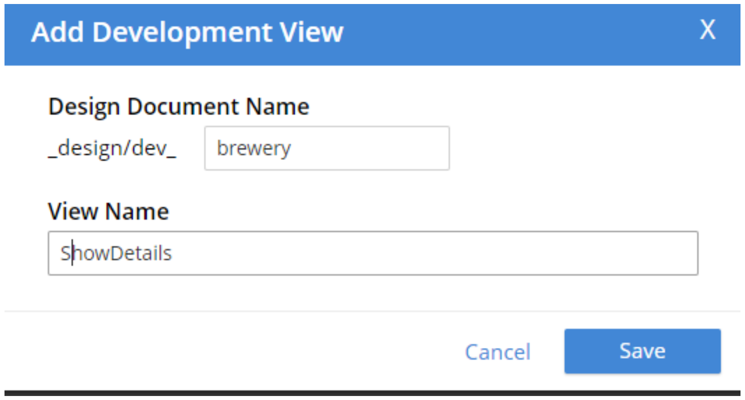
Views Editor

Indexes → Views → Add View



HP

Creating and Editing a View


Add Development View X

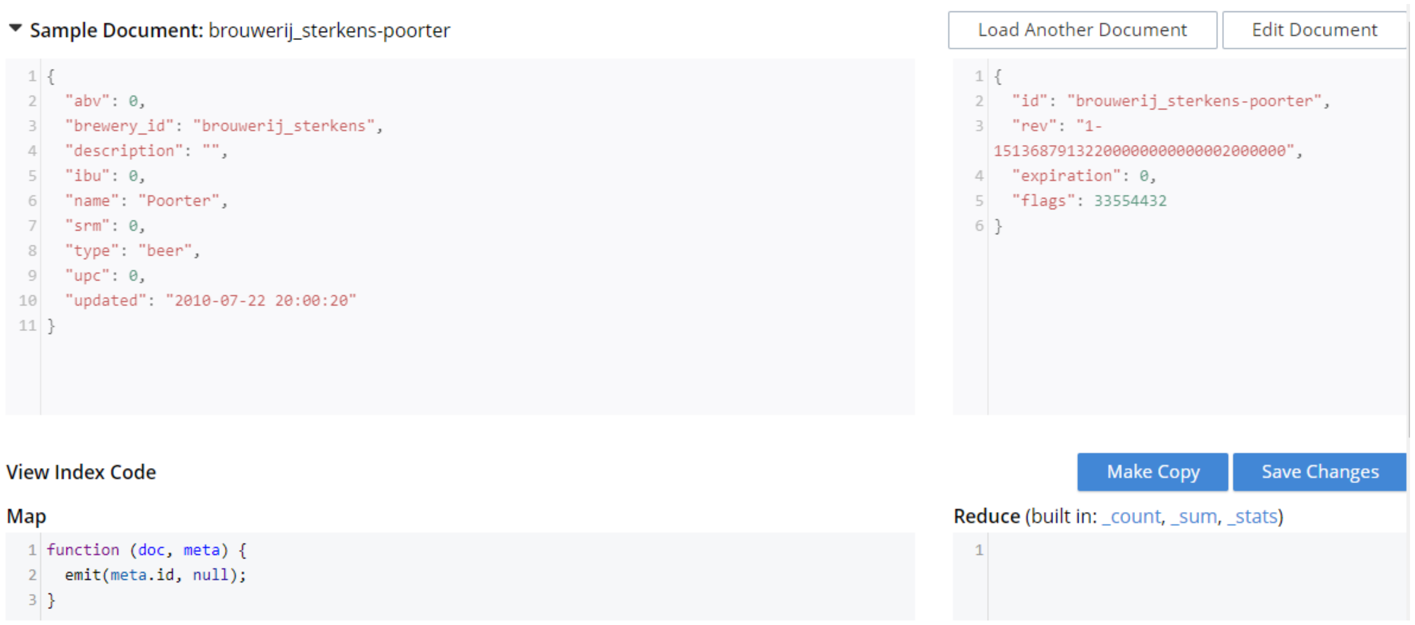
Design Document Name

design/dev brewery

View Name

ShowDetails

Cancel Save


Sample Document: brouwerij_sterkens-poorter

Load Another Document | Edit Document

```
{
  "abv": 0,
  "brewery_id": "brouwerij_sterkens",
  "description": "",
  "ibu": 0,
  "name": "Poorter",
  "srm": 0,
  "type": "beer",
  "upc": 0,
  "updated": "2010-07-22 20:00:20"
}
```

```
{
  "id": "brouwerij_sterkens-poorter",
  "rev": "1-
151368791322000000000000002000000",
  "expiration": 0,
  "flags": 33554432
}
```

View Index Code

Make Copy | Save Changes

Map

```
function (doc, meta) {
  emit(meta.id, null);
}
```

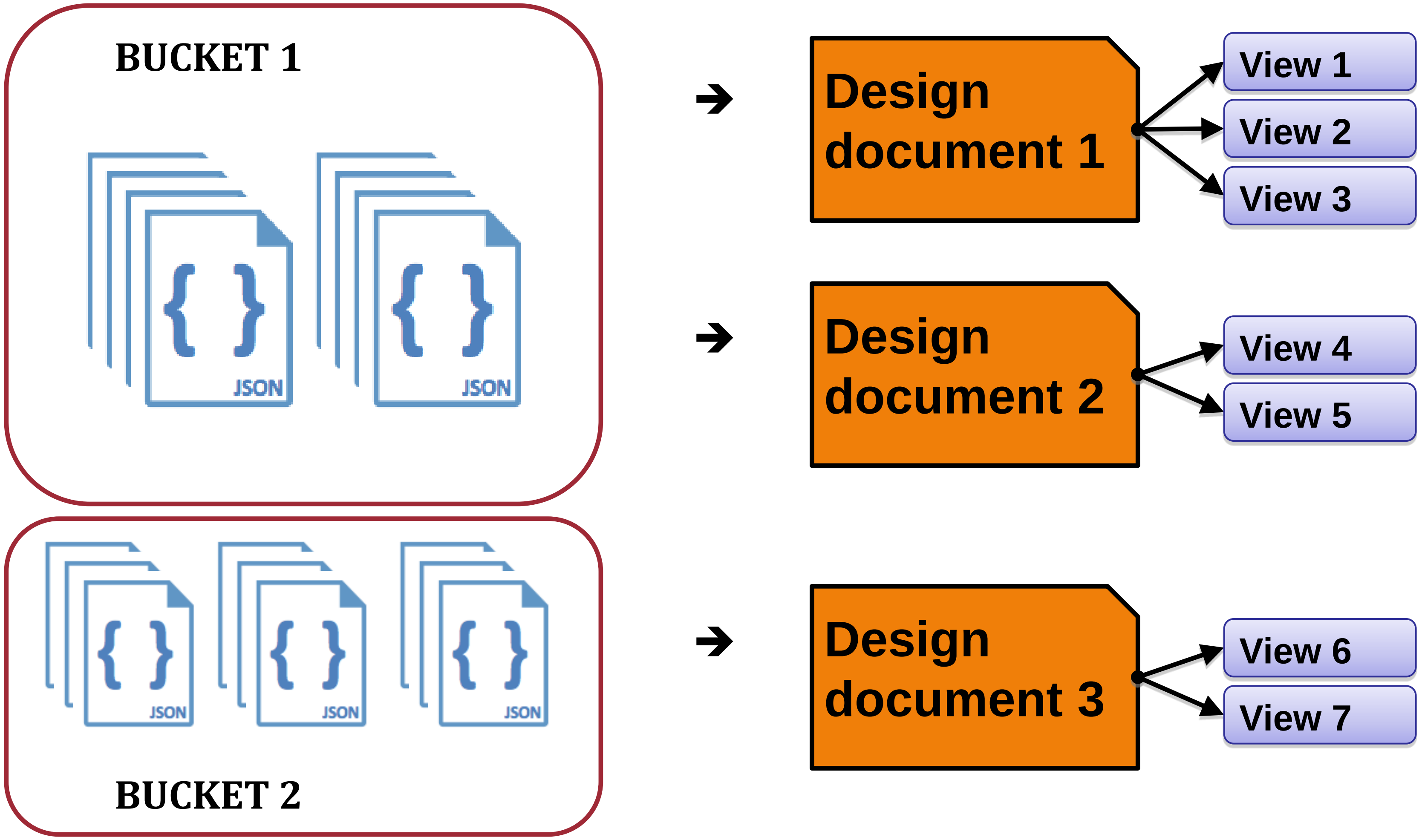
Reduce (built in: _count, _sum, _stats)

View Lifecycle
Define -> Build -> Query

HP

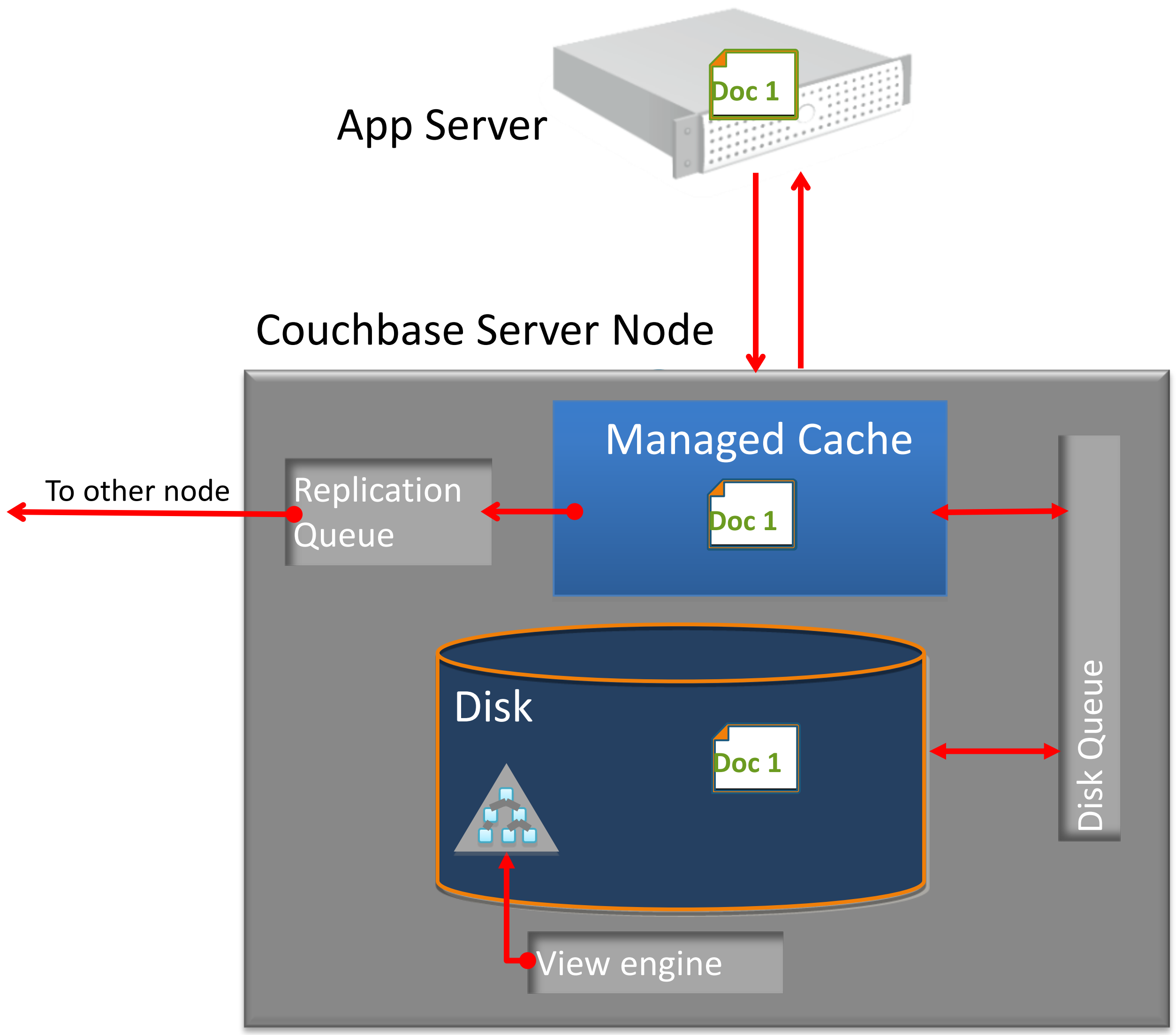
Buckets & Design docs & Views

- Create design documents on a bucket
- Create views within a design document



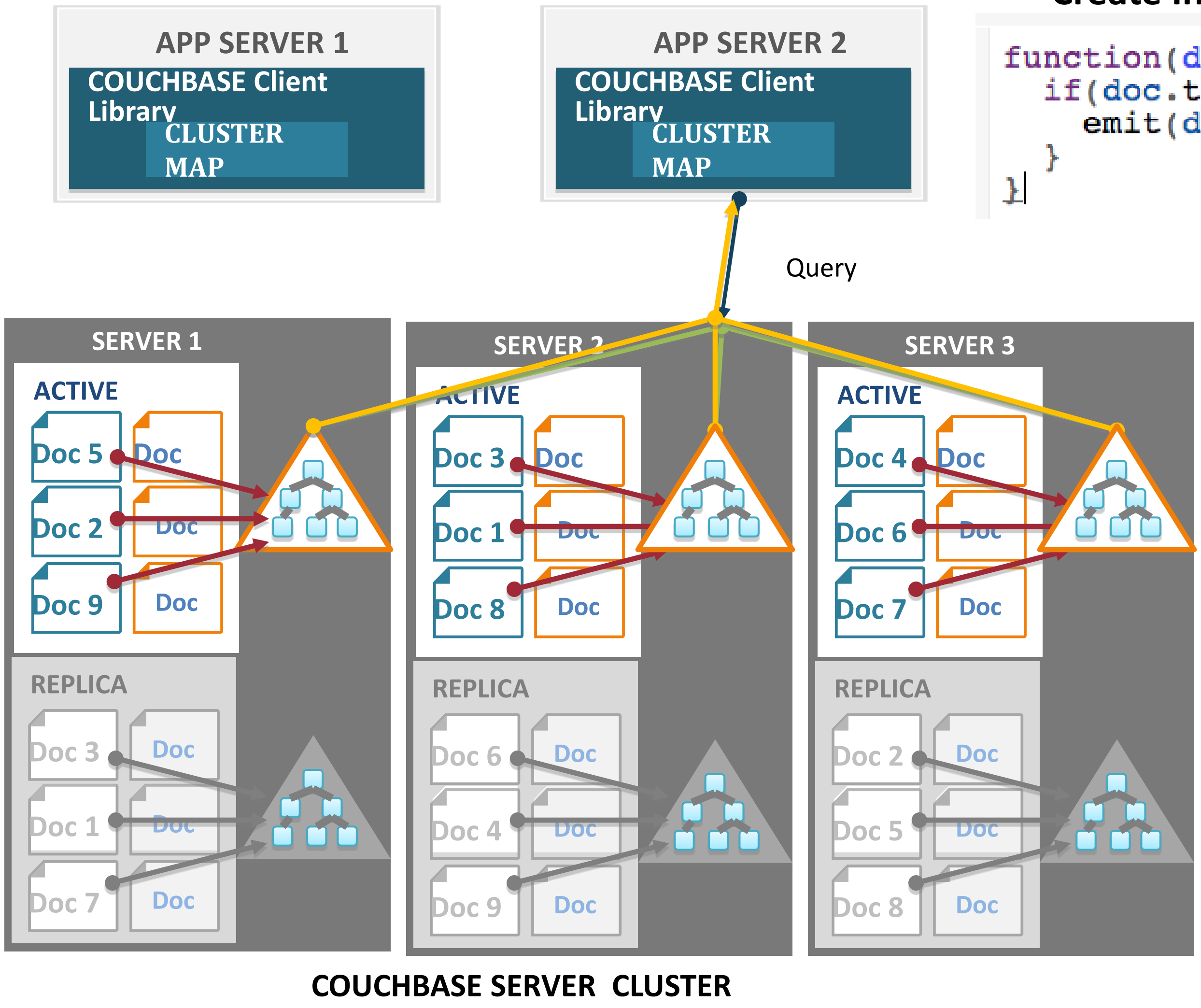
HP

Eventually indexed Views – Data flow



HP

Distributed Indexing and Querying



Create Index / View

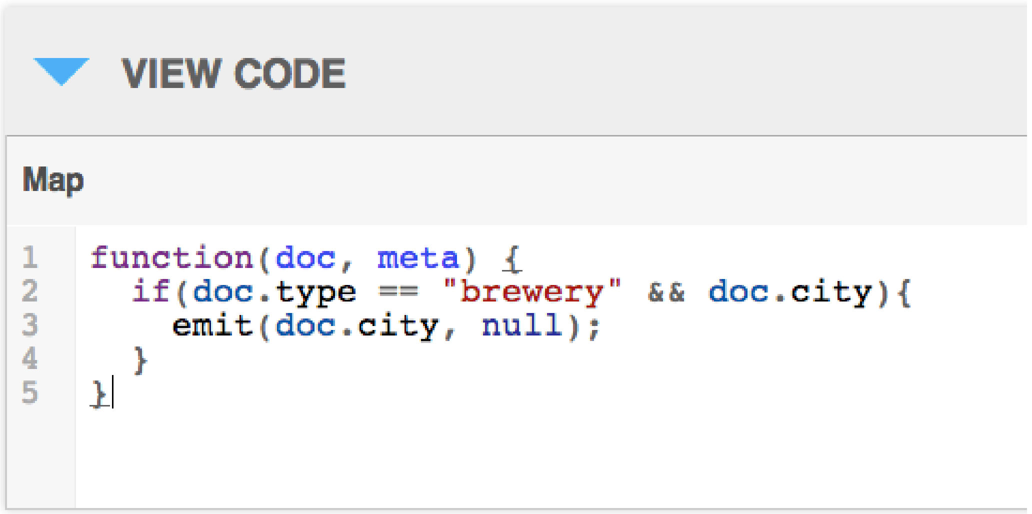
```
function(doc, meta) {
  if(doc.type == "brewery" && doc.city){
    emit(doc.city, null);
  }
}
```

- **Indexing work is distributed amongst nodes**
- **Parallelize the effort**
- **Each node has index for data stored on it**
- **Queries combine the results from required nodes**

User Configured Replica Count = 1

DEFINE → Index / View Definition in JavaScript

CREATE INDEX City ON Brewery.City;

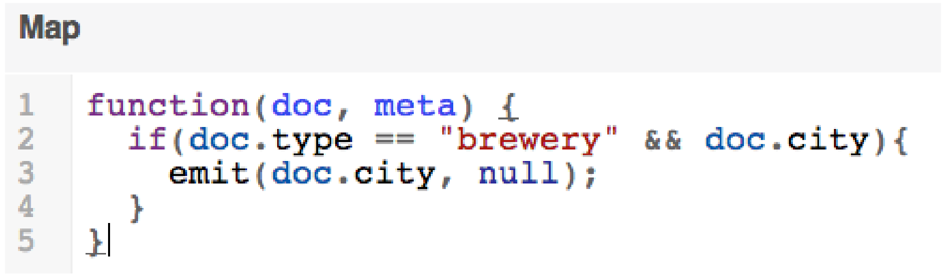


```
function(doc, meta) {
  if(doc.type == "brewery" && doc.city){
    emit(doc.city, null);
  }
}
```

HP

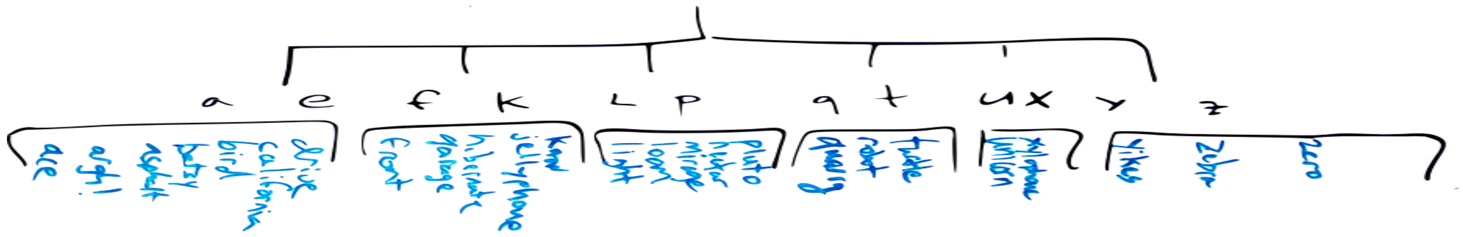
BUILD ➔ Distributed Index Build Phase

- Optimized for lookups, in-order access and aggregations
- View reads are from disk (different performance profile than GET/SET)
- Views built against every document on every node
  - Group them in a **design document**
- Views are automatically kept up to date



Map

```
function(doc, meta) {
  if(doc.type == "brewery" && doc.city){
    emit(doc.city, null);
  }
}
```

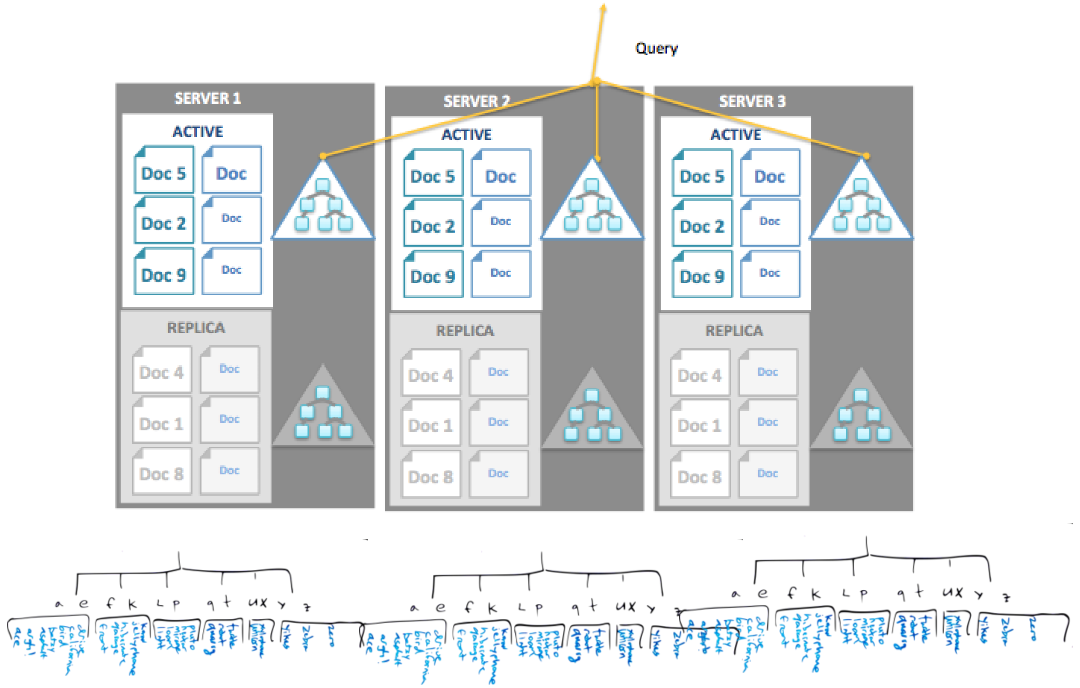


HP QUERY ➔ Dynamic Queries

Optional Aggregation

Query ?startkey="J"&endkey="K"

{"rows":[{"key":"Juneau","value":null}]}



HP

Index building details

- All the views within a design document are incrementally updated when the view is accessed or auto-indexing kicks in
- Automatic view updates
  - In addition to forcing an index build at query time, active & replica indexes are updated every 3 seconds of inactivity if there are at least 5000 new changes (configurable)

HP

Index building details

- The entire view is recreated if the view definition has changed
- Views can be conditionally updated by specifying the "***stale***" argument to the view query
- The index information stored on disk consists of the combination of both the key and value information defined within your view.

HP

Queries run against stale indexes by default

- **stale=update_after** (default if nothing is specified)
  - always get fastest response
  - can take two queries to read your own writes
- **stale=ok**
  - auto update will trigger eventually
  - might not see your own writes for a few minutes
  - least frequent updates -> least resource impact

HP

Queries run against stale indexes by default

- **stale=false**
  - Use with “set with persistence” if data needs to be included in view results
  - BUT be aware of delay it adds, only use when really required

HP

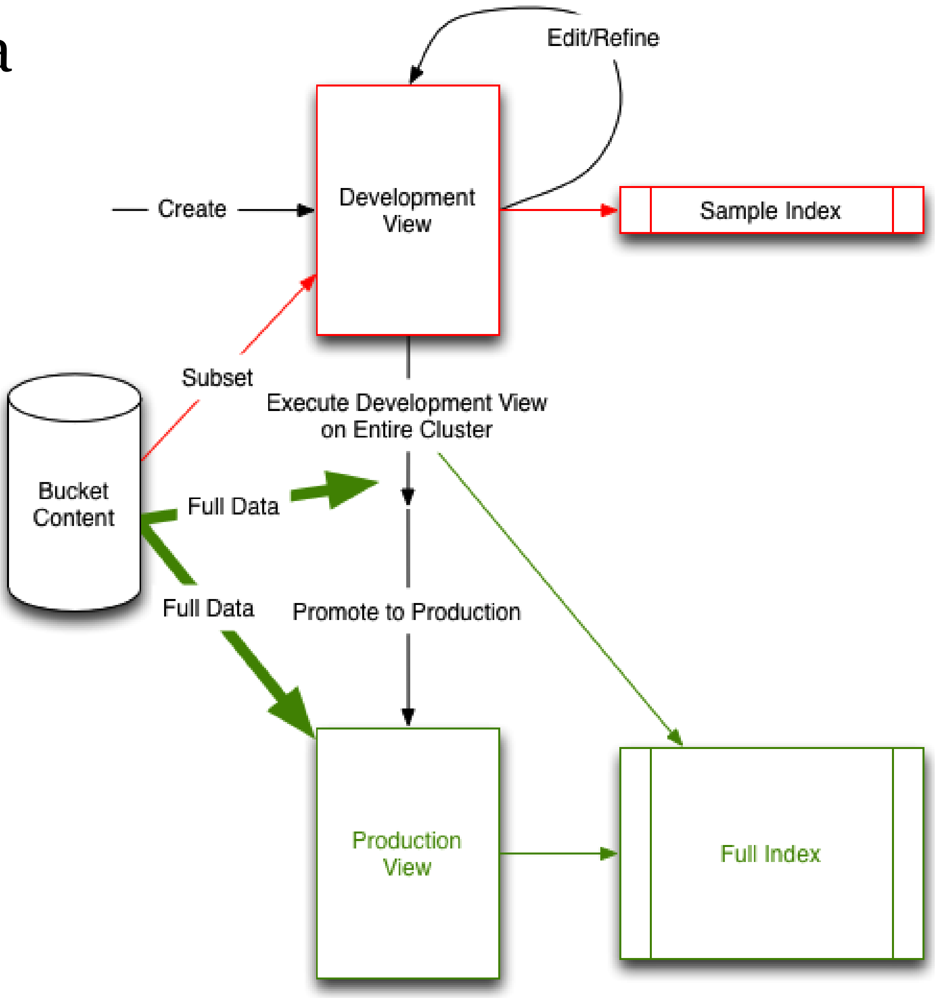
Views and Replica indexes

- In addition to replicas for data (up to 3 copies), optionally create replica for indexes
- Each node manages replica index data structures
- Set at a bucket level
- Replica index populated from replica data
- Replica index is used after a failover

HP

Development vs. Production Views

- Development views index a subset of the data.
- Publishing a view builds the index across the entire cluster.
- Queries on production views are scattered to all cluster members and results are gathered and returned to the client.



Built-in reduce

Built-in reduce

Couchbase has three built-in reduce functions

- **_count**: Returns the number of items
- **_sum**: Calculates the sum for numeric values returned in the value field of the results
- **_stats**: Calculates both the count and sum values, as well as minimum value, maximum value, and the sum of squares of the values (*value 2 + value 2 + . . . + value 2)*

HP

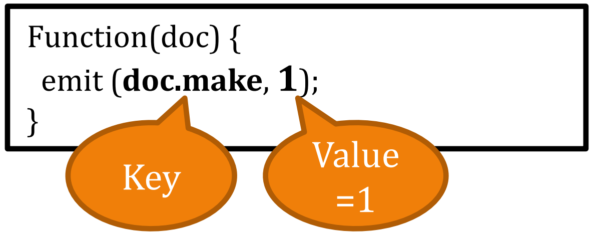
Simple Map/Reduce Example

- A List of Cars

Id: 1	Id: 2	Id: 3	Id: 4	Id: 5
make: Audi	make: Audi	make: VW	make: VW	make: VW
model: A3	model: A4	model: Golf	model: Golf	model: Polo
year: 2000	year: 2009	year: 2009	year: 2008	year: 2010
price: 5.400	price: 16.000	price: 15.000	price: 9.000	price: 12.000

- Step 1: Make a map, ordered by make

```
Function(doc) {
  emit (doc.make, 1);
}
```



- Result:

Key	, Value
Audi	, 1
Audi	, 1
VW,	1
VW,	1
VW,	1

Simple Map/Reduce Example

- Result:

Key	**, Value**
Audi	, 1
Audi	, 1
VW	, 1
VW	, 1
VW	, 1

- Step 2: Write a “sum“-reduce

```
function(keys,values) {
  return sum(values);
}
```

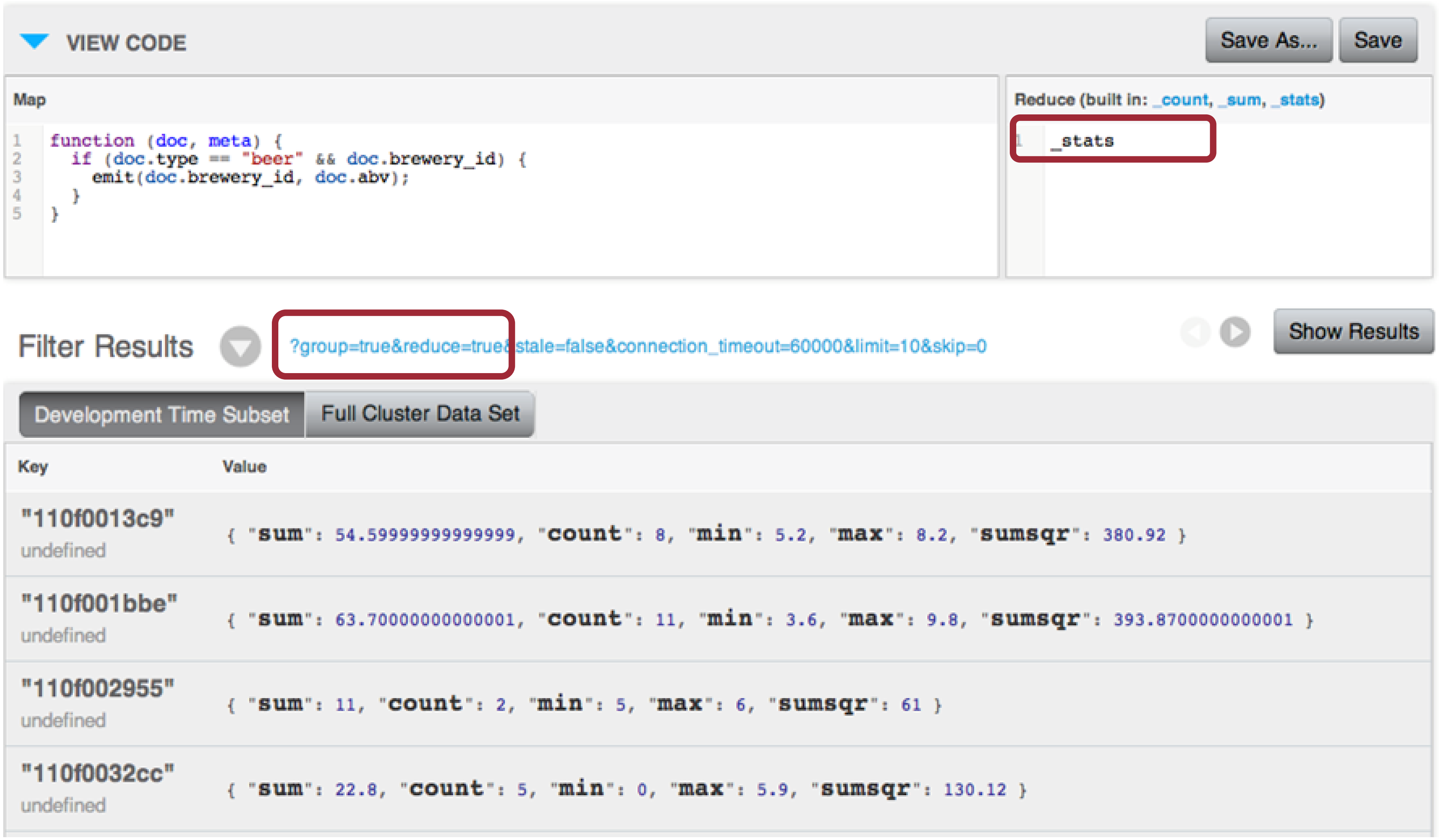
- Result:

Key	**, Value**
null	, 5

HP

Use a built-in reduce function with a group query

- Lets find **average abv** for each brewery!



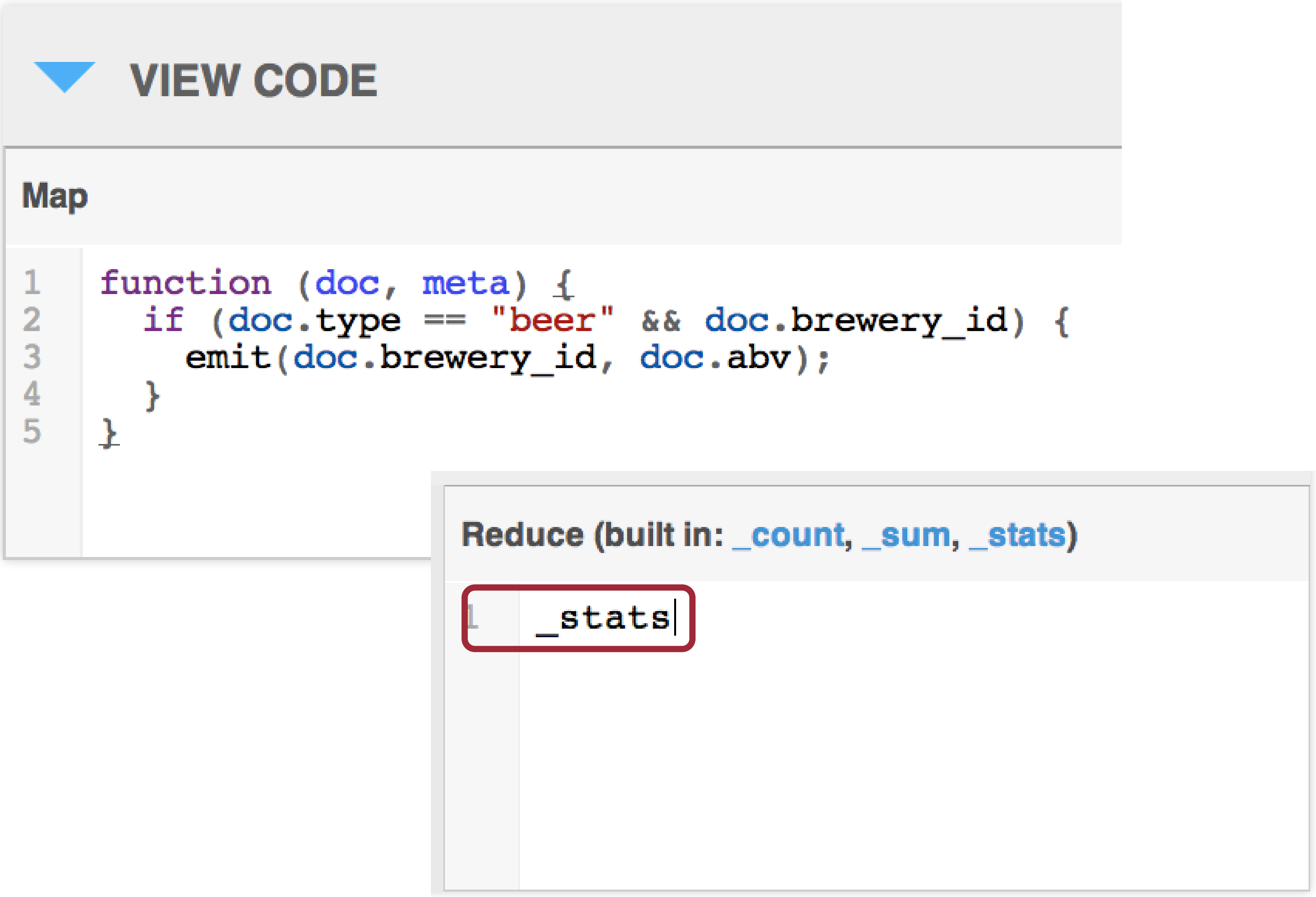
```
function (doc, meta) {
  if (doc.type == "beer" && doc.brewery_id) {
    emit(doc.brewery_id, doc.abv);
  }
}
```

```
_stats
```

?group=true&reduce=true&stale=false&connection_timeout=60000&limit=10&skip=0

Key	Value
"110f0013c9" undefined	{ "sum": 54.59999999999999, "count": 8, "min": 5.2, "max": 8.2, "sumsqr": 380.92 }
"110f001bbe" undefined	{ "sum": 63.70000000000001, "count": 11, "min": 3.6, "max": 9.8, "sumsqr": 393.8700000000001 }
"110f002955" undefined	{ "sum": 11, "count": 2, "min": 5, "max": 6, "sumsqr": 61 }
"110f0032cc" undefined	{ "sum": 22.8, "count": 5, "min": 0, "max": 5.9, "sumsqr": 130.12 }

We are reducing doc.abv with _stats



VIEW CODE

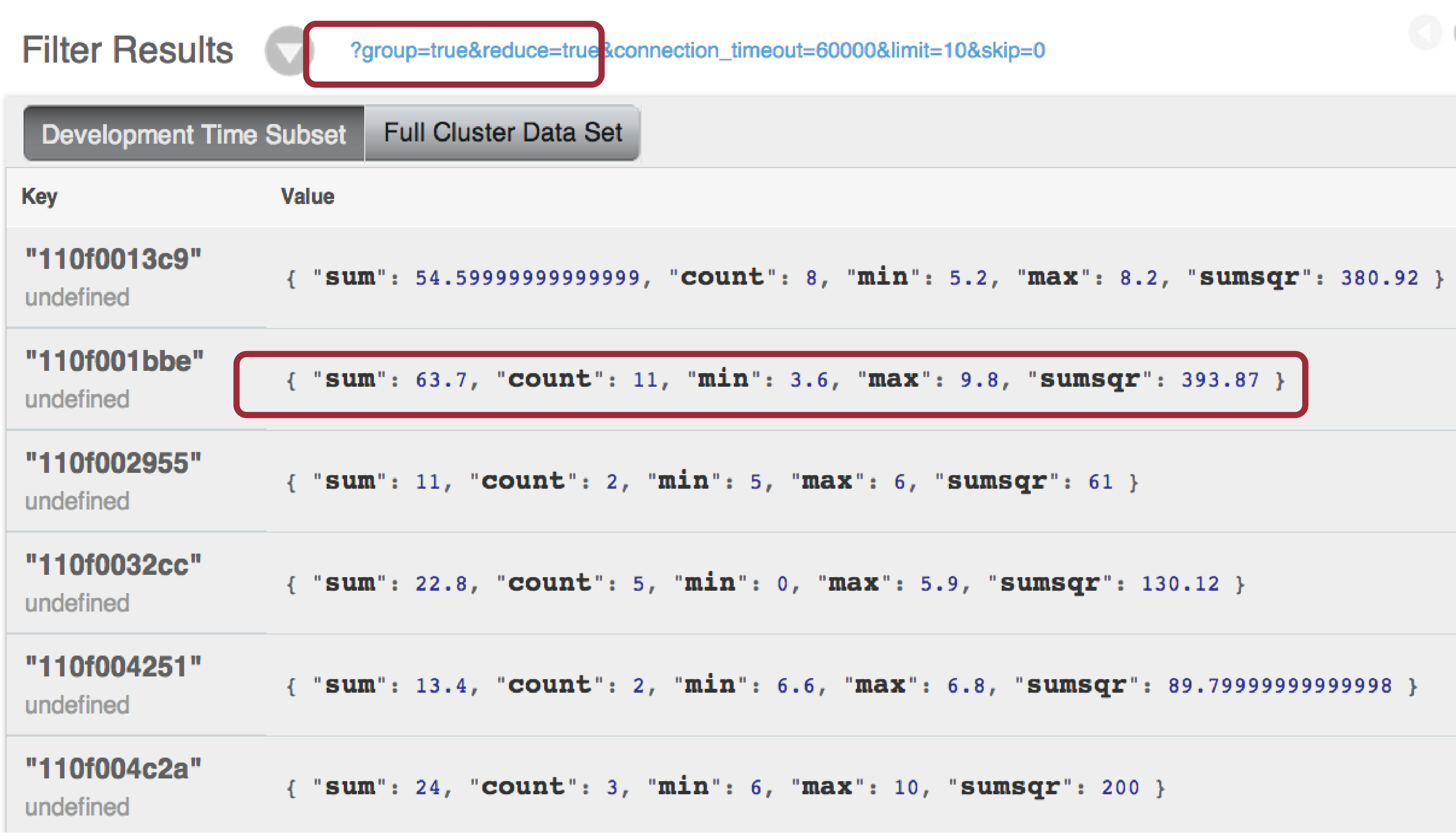
Map

```
function (doc, meta) {
  if (doc.type == "beer" && doc.brewery_id) {
    emit(doc.brewery_id, doc.abv);
  }
}
```

Reduce (built in: _count, _sum, _stats)

```
_stats
```

Group reduce (reduce by unique key)



?group=true&reduce=true&connection_timeout=60000&limit=10&skip=0

Key	Value
"110f0013c9" undefined	{ "sum": 54.59999999999999, "count": 8, "min": 5.2, "max": 8.2, "sumsqr": 380.92 }
"110f001bbe" undefined	{ "sum": 63.7, "count": 11, "min": 3.6, "max": 9.8, "sumsqr": 393.87 }
"110f002955" undefined	{ "sum": 11, "count": 2, "min": 5, "max": 6, "sumsqr": 61 }
"110f0032cc" undefined	{ "sum": 22.8, "count": 5, "min": 0, "max": 5.9, "sumsqr": 130.12 }
"110f004251" undefined	{ "sum": 13.4, "count": 2, "min": 6.6, "max": 6.8, "sumsqr": 89.79999999999998 }
"110f004c2a" undefined	{ "sum": 24, "count": 3, "min": 6, "max": 10, "sumsqr": 200 }

Lab: View Creation