CouchDB

CMPU4003 ADVANCED DATABASES

What is Apache CouchDB?

An open source document based key-value storage NoSQL database

Uses JSON to store data

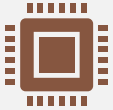
CouchDB was written in Erlang programming language (https://www.erlang.org/)

It was started by Damien Katz in 2005.

Became an Apache project in 2008.

What is Apache CouchDB?

JavaScript as its query language



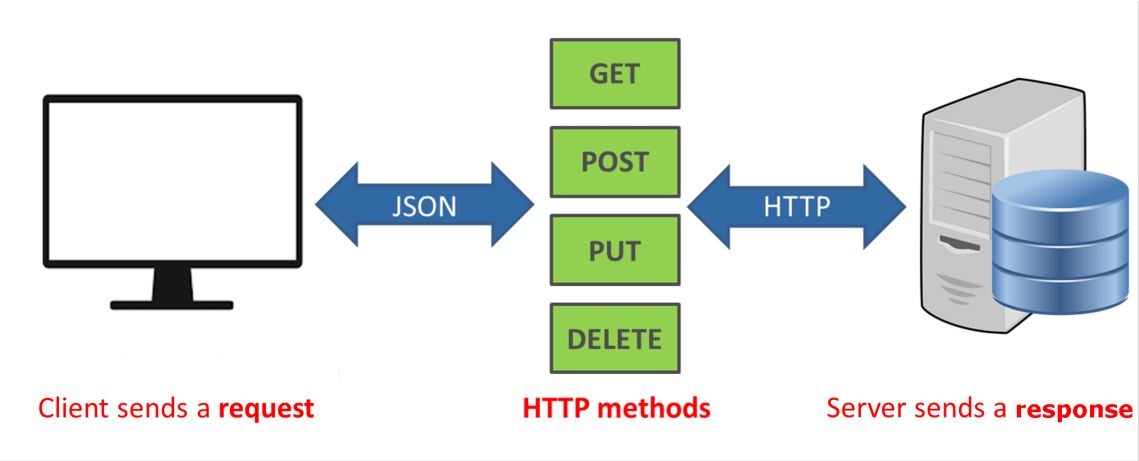
Provides a RESTful API

REST (REpresentational State Transfer) paradigm

Provides standards which streamline communication between web components

Separate concerns of the client and the server

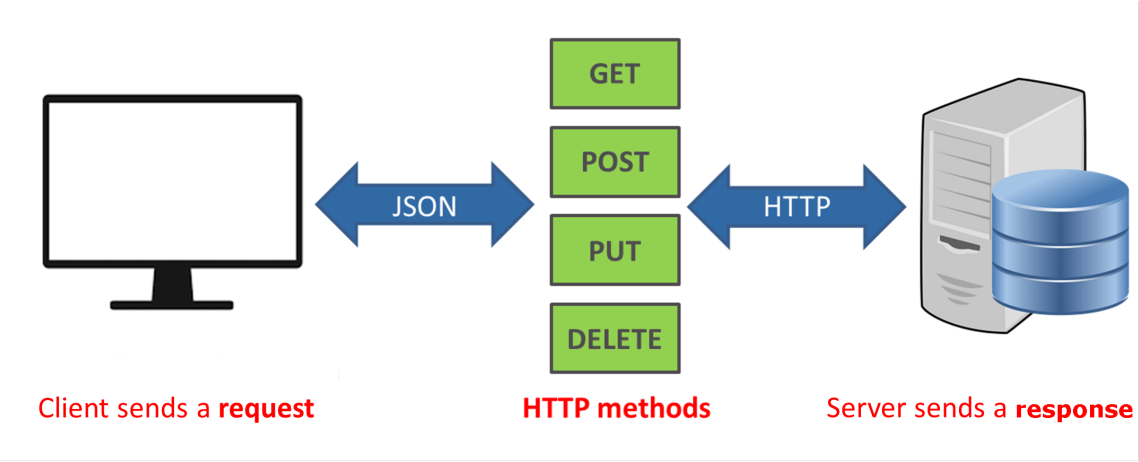
What is Apache CouchDB?



A RESTful web application exposes information about itself in the form of information about its resources

It also enables the client to take actions on those resources, such as create new resources (i.e. create a new user) or change existing resources (i.e. edit a post).

What is Apache CouchDB?



Stateless

- The server does not need to know anything about what state the client is in and vice versa
- A request from the client to the server must contain all information the server needs to understand and complete the request
- Client must keep the state if needed

How is data stored in CouchDB

Primary unit of data is a **document**

- Can consist of any number of **fields** and **attachments**
- Also include **metadata** maintained by the database system

Each database is a **collection of documents**

An application may access multiple databases on different servers

Interacting with CouchDB

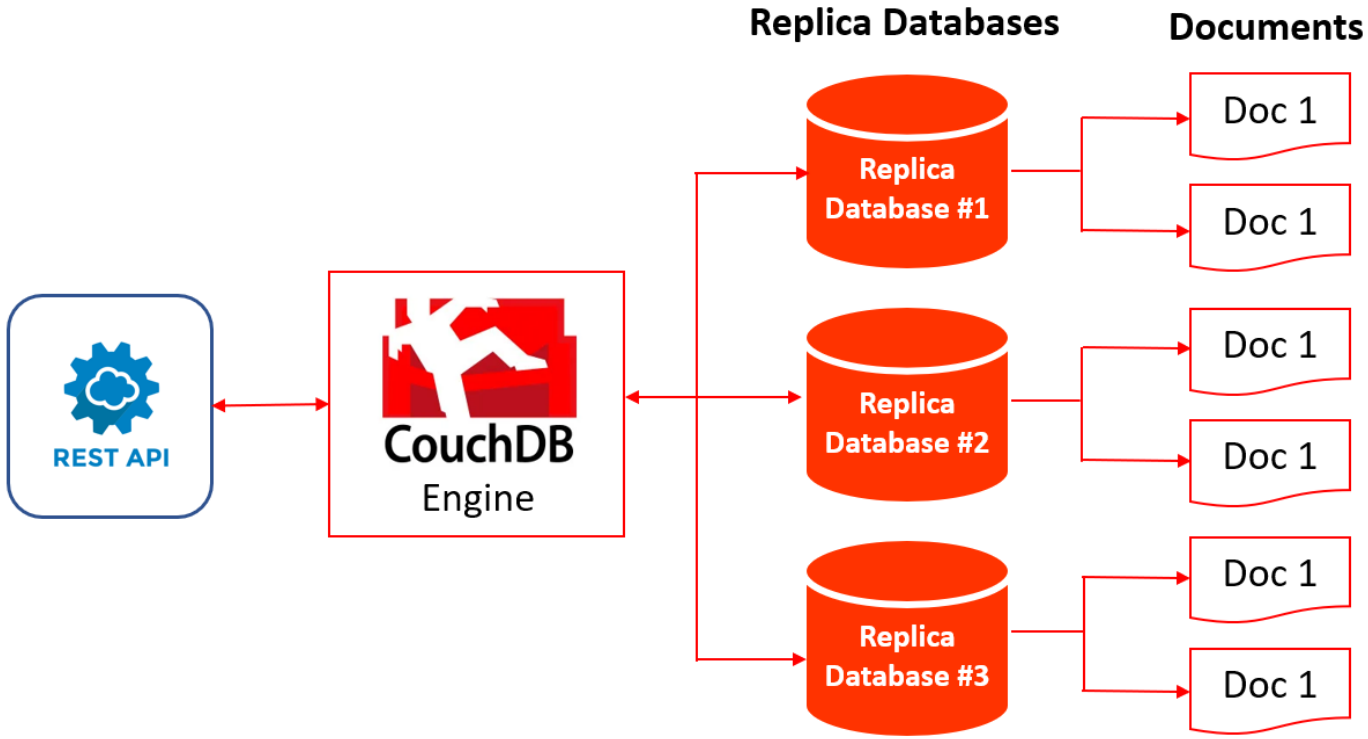
Use HTTP request headers

Through these requests :

- Can retrieve data/databases
- Store data/databases
- View data/databases
- Configure data/databases

Interacting with CouchDB

- GET – Used to get a specific item.
  - Different items require different URL patterns.
  - GET request can be used to retrieve static items e.g. database documents and configuration, statistical information
    - Will be in JSON format (mostly)
- POST – Used to upload data
  - Can be used to set values, upload documents, set document values, and can also start certain administration commands.
- PUT - Create new objects, databases, documents, views and design documents.
- DELETE – Delete documents, views, and design documents.
- COPY – Using COPY method, you can copy documents and objects.



CouchDB Architecture

Interacting with CouchDB

cURL (Client URL) utility

A command line utility used to transfer data

Allows a user access HTTP protocol straight away from the command line

Interacting with CouchDB

Using Curl (Client URL) you exchange data between a client and a server

- Working on your own machine – launch a command window (or Git Bash)

```
curl -X GET http://admin:couchdb@127.0.0.1:5984
```

```
curl -X GET
http://admin:couchdb@127.0.0.1:5984/_all_dbs
```

CouchDB Architecture

The API can be subdivided into the following sections:

- Server
- Databases
- Documents
- Replication

Server

curl -X GET
http://adminusername:adminpassword@serveraddress:port#

curl -X GET http://admin:couchdb@127.0.0.1:5984

```
C:\Users\deirdre.lawless>curl -X GET http://admin:couchdb@127.0.0.1:5984
{"couchdb":"Welcome","version":"3.3.3","git_sha":"40afbcfc7","uuid":"84377ead0747fdd26fdc86a81e84d2fd","features":["acce
ss-ready","partitioned","pluggable-storage-engines","reshard","scheduler"],"vendor":{"name":"The Apache Software Foundat
ion"}}
```

Databases

```
curl -X GET http://admin:CouchDB@127.0.0.1:5984/_all_dbs
```

```
C:\Users\deirdre.lawless>curl -X GET http://admin:couchdb@127.0.0.1:5984/_all_dbs
["_replicator","examresults","examresultsreplica","qanda","results_db","resultsdb","votes","votes_db"]
```

Create a Database

```
curl -X PUT
http://admin:couchdb@127.0.0.1:5984/dbdemo
```

```
C:\Users\deirdre.lawless>curl -X PUT http://admin:couchdb@127.0.0.1:5984/dbdemo
{"ok":true}
```

Delete a Database

```
curl -X DELETE
http://admin:couchdb@127.0.0.1:5984/dbdemo1
```

All Documents

```
curl -X GET http://admin:couchdb@127.0.0.1:5984/examresults/_all_docs
```

```
C:\Users\deirdre.lawless>curl -X GET http://admin:couchdb@127.0.0.1:5984/examresults/_all_docs
{"total_rows":4000,"offset":0,"rows":[
{"id":"_design/resultquery","key":"_design/resultquery","value":{"rev":"14-ccc26cbeebbd74a8a2dc1f45b50d6e76"}},
{"id":"_design/resultquery1","key":"_design/resultquery1","value":{"rev":"1-ad7cdda1f0fcd3d95c2d3deacafcd28d"}},
{"id":"ddaac67cbfc0cd252b93a072500004ac","key":"ddaac67cbfc0cd252b93a072500004ac","value":{"rev":"5-5b7375dbfc4f0b730
d7cd9ff540db"}},
{"id":"ddaac67cbfc0cd252b93a07250000691","key":"ddaac67cbfc0cd252b93a07250000691","value":{"rev":"2-71ca372d0acfcc36b
845be0ada7eb"}},
{"id":"ddaac67cbfc0cd252b93a072500009c1","key":"ddaac67cbfc0cd252b93a072500009c1","value":{"rev":"1-aa02567e2e3a84799
54d547b5f5d1"}},
{"id":"ddaac67cbfc0cd252b93a07250001582","key":"ddaac67cbfc0cd252b93a07250001582","value":{"rev":"1-a8378919110c6c5f6
18eb0d577656"}},
{"id":"ddaac67cbfc0cd252b93a07250001e32","key":"ddaac67cbfc0cd252b93a07250001e32","value":{"rev":"2-a249fe170ba578e79
45e430b6b8f4"}},
{"id":"ddaac67cbfc0cd252b93a0725000296e","key":"ddaac67cbfc0cd252b93a0725000296e","value":{"rev":"1-d34aaee02f9aed893
9855122e67ce"}},
```

Get a Document

```
curl -X GET
http://admin:couchdb@127.0.0.1:5984/examresults/ddaac67cbfc0cd252b93a072500004ac
```

You need to provide the id.

Results in the document:


```
{"_id":"ddaac67cbfc0cd252b93a072500004ac","_rev":"5-5b7375dbfc4f0b730c17d7cd9ff540db","studentid":69,"firstname":"Peter
smith","lastname":"Mcclain","examid":37,"examdate":"2019-12-01","score":60.15,"grade":"C"}
```

Documents

Each document in CouchDB has an ID.

This ID is unique per database.

You are free to choose any string to be the ID, but for best results use a UUID (or GUID),

- i.e., a Universally (or Globally) Unique Identifier
  - Random numbers (you should allow the DB to allocate) that have such a low collision probability that everybody can make thousands of UUIDs a minute for millions of years without ever creating a duplicate.

Ensure two independent people cannot create two different documents with the same ID.

- Collision

Insert a Document

```
curl -X PUT
http://admin:couchdb@127.0.0.1:5984/examresults/S12345
-databases \
   -H "Content-Type: application/json" \
   -d '{
     "student_id": "S12345",
     "course": "Databases",
     "score": 82
   }’
```

Assigning id: S12345-databases

Insert a Document

```
curl -X POST
http://admin:couchdb@127.0.0.1:5984/examresults \
  -H "Content-Type: application/json" \
  -d '{
    "student_id": "S12345",
    "course": "Databases",
    "score": 82
  }'
```

Letting Couch Assign the ID

```
{"ok":true,"id":"bddc8705b5647a926bebc19472001c8c","rev":"1-1749c868c86cf033f703177033c8ca04"}
```

Documents

The central data structure.

JSON format.

Can have attachments.

Revisions

If you want to change a document in CouchDB

- The full document is loaded out of CouchDB
- Changes are made to the JSON structure (or object, when you are doing actual programming)
- The document is saved as a **new revision** he entire of that document back into CouchDB
- A revision identified is part of the metadata
- Each revision is identified by a new **_rev** value.

Revisions

If you want to update or delete a document, then CouchDB expects you to include the _rev field of the revision you wish to change.

Revisions

Changing Documents

When CouchDB accepts the change, it will generate a new revision number.

- This ensures that, if somebody else made a change before you got to request the document update, CouchDB will not accept your update because you are likely to overwrite data you didn't know existed.
- Simple language: whoever saves a change to a document first, wins.

Bulk Loading Data

Suppose we want to create a database that has information about student achievements on different courses

Rather than create documents one by one we can bulk load the data

First create the database

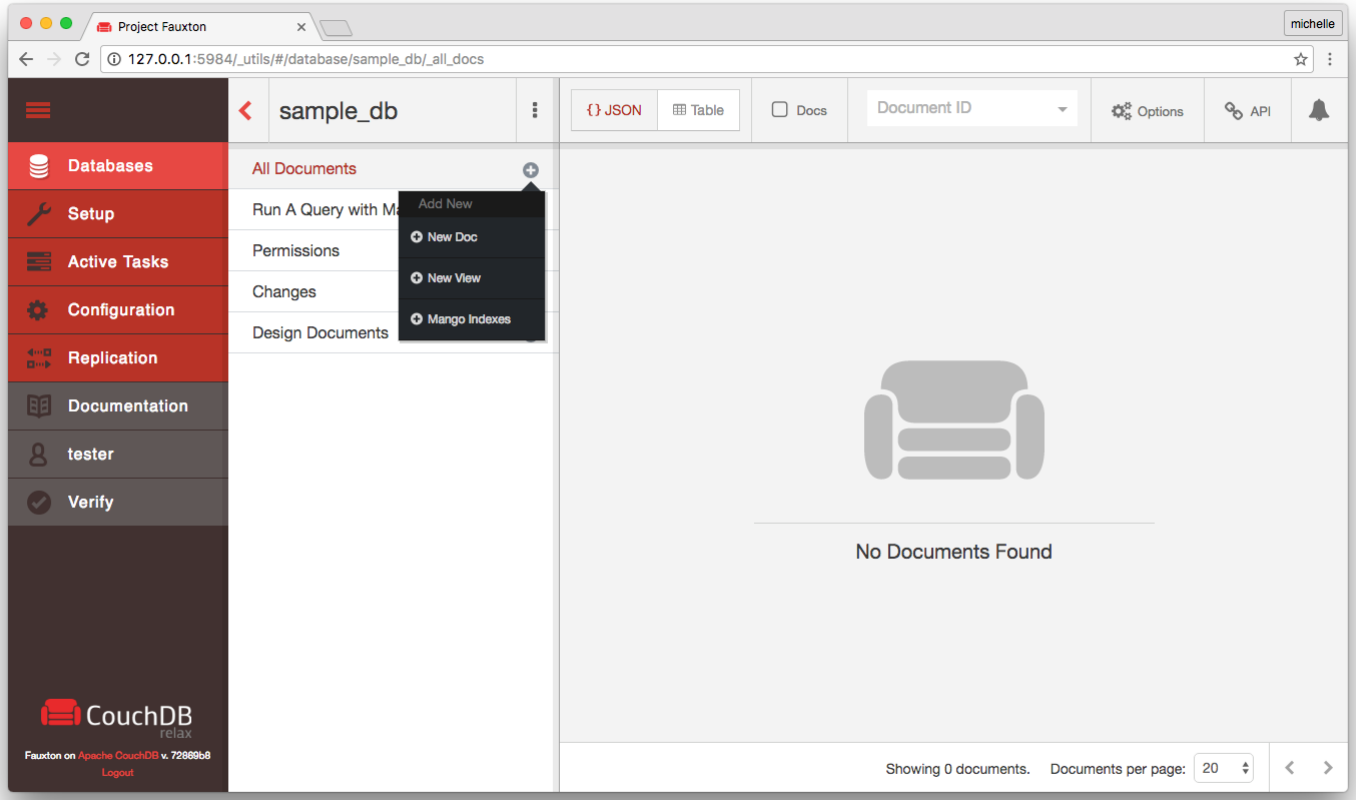
```
curl -X PUT
http://admin:couchdb@127.0.0.1:5984/students
```

Then load the data

```
curl -X POST
http://admin:couchdb@127.0.0.1:5984/students/_bulk_docs -H
"Content-type: application/json" -d @setupstudentdata.json
```

Verify the data

```
curl http://admin:couchdb@127.0.0.1:5984/students/_all_docs
```



Fauxton

http://127.0.0.1:5984/_utils/

Querying CouchDB

Design Document

A special type of document within a CouchDB database.

You can use design document to build indexes, validate document updates, format query results, and filter replications.

Contain application code (javascript)

Like normal documents except prefixed by **_design (this is the id)**

CouchDB looks for views and other application functions in the design documents

Design Documents

CouchDB delegates computation of design documents functions to **query servers**

- A Query server is an external process that communicates with CouchDB by JSON protocol
- Communicates through stdio interface
- Processes all design functions calls, such as JavaScript views.

CouchDB Architecture

By default, CouchDB has a built-in Javascript query server

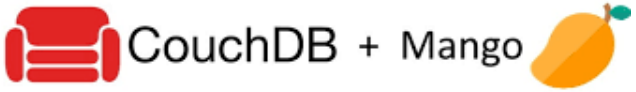
We can define Javascript functions to pull back documents/parts of documents we want to query

- We can also configure other engines if we wish to

Mango Query Server

CouchDB also has a built-in Mango query server for us to query documents.

Mango is a MongoDB inspired query language interface



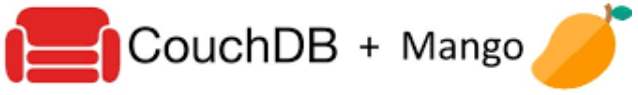
Designed to be simple to implement on the client side

Providing users a more natural conversion to Apache CouchDB than using the standard RESTful HTTP interface

Mango Query Server

Mango provides a single HTTP API endpoint that accepts JSON bodies via HTTP POST.

These bodies provide a set of instructions that will be handled with the results being returned to the client in the same order as they were specified.



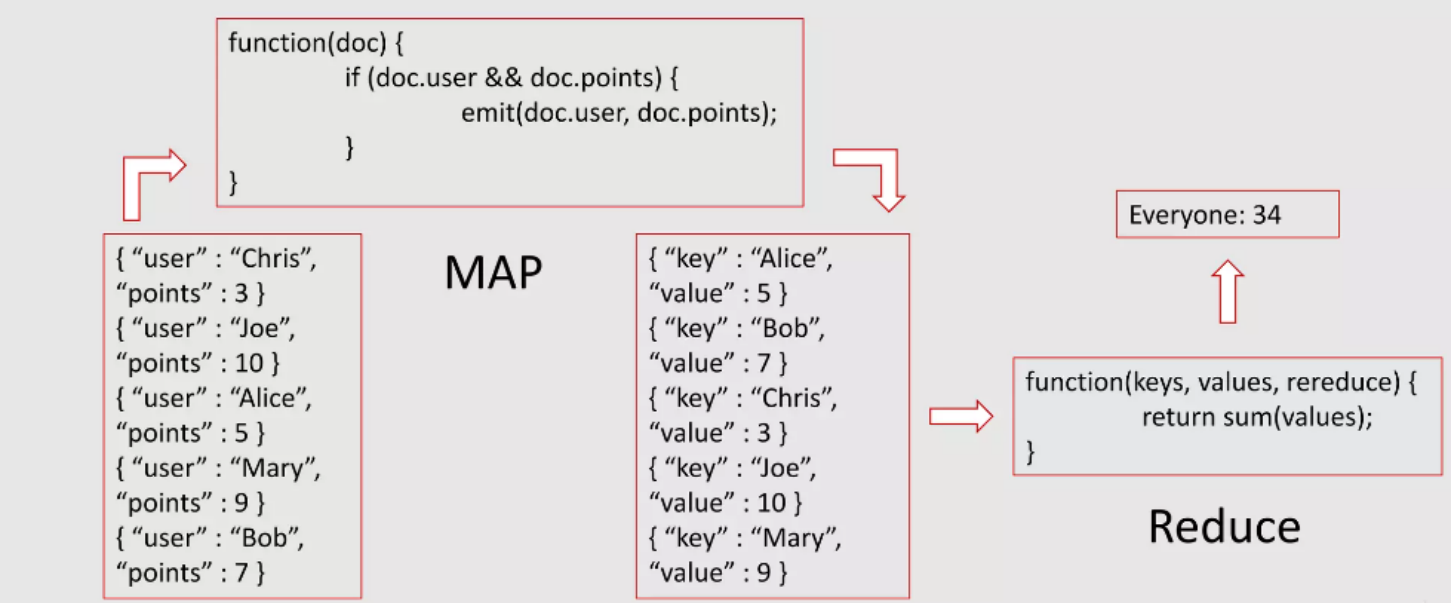
View

Views are the primary tool used for querying and reporting on CouchDB documents

Used for:

- Filtering documents to find those relevant to a particular process
- Extracting data from documents and presenting it in a specific order.
- Building efficient indexes to find documents by any value or structure that resides in them.
- Use these indexes to represent relationships among documents.
- Make calculations on the data in your documents.

CouchDB's MapReduce queries are stored in the views field of design documents.



MapReduce

Invented by Google

No Join between documents

Map function: builds a list of key/value pairs

Reduce function: Reduces this down to a single result

View

Suppose we have three documents.

We want to filter based on date and title.

We create a Javascript function

```
function(doc) {
    if(doc.date && doc.title) {
        emit(doc.date, doc.title);
    }
}
```

This is a MAP function

We can create this view function as a string stored inside the views field of a design document.

View

Suppose we have three documents.

We want to filter based on date and title.

We create a Javascript function

```
function(doc) {
    if(doc.date && doc.title) {
        emit(doc.date, doc.title);
    }
}
```

emit() function
Always takes two arguments: the first is **key**, and the second is **value**.

Creates an entry in our view result.

Can be called multiple times in the map function to create multiple entries in the view results from a single document.

View

Suppose we want to query our Students data by course and by degree

```
{
 "_id": "_design/students",
 "views": {
  "by_course": {
   "map": "function(doc) { if (doc.course_name) {
emit(doc.course_name, doc); } }"
  },
  "by_degree": {
   "map": "function(doc) { if (doc.degree_name) {
emit(doc.degree_name, doc); } }"
  }
 }
}
```

View - Filter

To create this view you can use this command:

```
curl -X PUT http://admin:couchdb@127.0.0.1:5984/students/_design/students \
  -H "Content-Type: application/json" \
  -d '{
    "_id": "_design/students",
    "views": {
      "by_course": {
        "map": "function(doc) { if (doc.course_name) { emit(doc.course_name, doc); } }"
      },
      "by_degree": {
        "map": "function(doc) { if (doc.degree_name) { emit(doc.degree_name, doc); } }"
      }
    }
  }'
```

<Curl Command> <server>/db/_design/nameofdesigndoc –d {view : map function}

View - Filter

We then *query the view,* CouchDB takes the source code and runs it for you on every document in the database your view was defined in. You query your view to retrieve the view result using the following command:

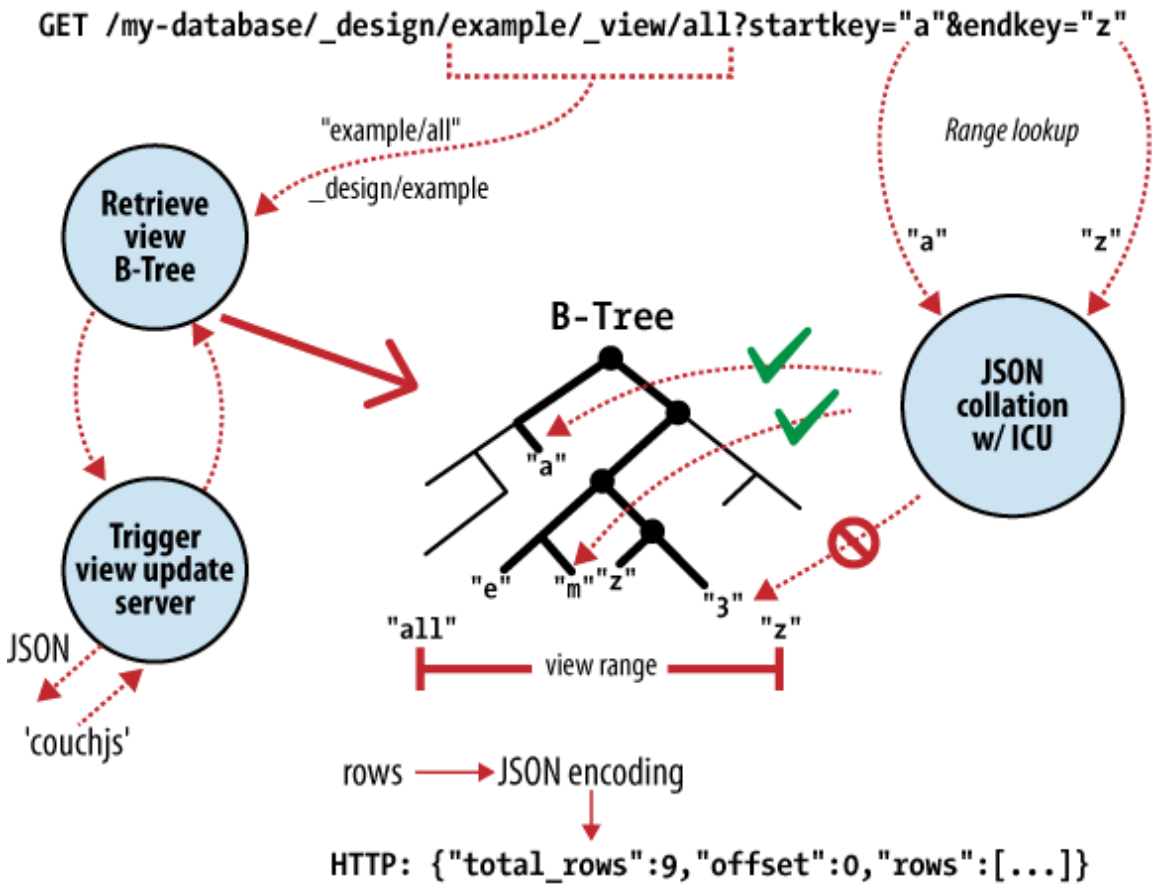
curl -X GET "http://admin:couchdb@127.0.0.1:5984/students/_design/students/_view/by_course"


```
{"total_rows":11,"offset":0,"rows":[
{"id":"1","key":"Advanced Databases","value":{"_id":"1","_rev":"1-9d497a2d349f5c6c573fbe4254ad596b","student_sk":1,"degree_sk":1,"course_sk":1,"date_sk":1,"pass":"Pass","course_name":"Advanced Databases","degree_name":"BSc in Computer Science","examdate":"2022-01-16"}},
{"id":"7","key":"Advanced Databases","value":{"_id":"7","_rev":"1-373fd7778585fa0a8576d92de35af4b8","student_sk":4,"degree_sk":3,"course_sk":1,"date_sk":3,"pass":"Pass","course_name":"Advanced Databases","degree_name":"BSc in Computer Science(Infrastructure)","examdate":"2022-01-13"}},
{"id":"3","key":"Big Data Analytics","value":{"_id":"3","_rev":"1-2a85b802070a0497b5292a55e3338065","student_sk":1,"degree_sk":1,"course_sk":4,"date_sk":4,"pass":"Pass","course_name":"Big Data Analytics","degree_name":"BSc in Computer Science","examdate":"2022-06-16"}},
{"id":"9","key":"Big Data Analytics","value":{"_id":"9","_rev":"1-254268a661eb550e13f4f1b6a650402f","student_sk":4,"degree_sk":3,"course_sk":4,"date_sk":4,"pass":"Pass","course_name":"Big Data Analytics","degree_name":"BSc in Computer Science(Infrastructure)","examdate":"2022-06-16"}},
{"id":"8","key":"Forensics","value":{"_id":"8","_rev":"1-7f73f97ab0f28d426de87c11eec99eb9","student_sk":4,"degree_sk":3,"course_sk":3,"date_sk":3,"pass":"Pass","course_name":"Forensics","degree_name":"BSc in Computer Science(Infrastructure)","examdate":"2022-01-13"}},
{"id":"10","key":"IT Security","value":{"_id":"10","_rev":"1-893e8d636d77ac180453f5c72913e0dd","student_sk":5,"degree_sk":3,"course_sk":5,"date_sk":7,"pass":"Pass","course_name":"IT Security","degree_name":"BSc in Computer Science(Infrastructure)","examdate":"2022-01-15"}},
{"id":"4","key":"IT Security","value":{"_id":"4","_rev":"1-6bb1410486f0eae00b51034d3a334a35","student_sk":1,"degree_sk":1,"course_sk":5,"date_sk":5,"pass":"Pass","course_name":"IT Security","degree_name":"BSc in Computer Science","examdate":"2022-06-15"}},
{"id":"6","key":"IT Security","value":{"_id":"6","_rev":"1-815c5243281eda3a30cf33cd0864ad44","student_sk":3,"degree_sk":2,"course_sk":5,"date_sk":5,"pass":"Fail","course_name":"IT Security","degree_name":"BSc in Computer Science(International)","examdate":"2022-06-15"}},
{"id":"2","key":"Machine Learning","value":{"_id":"2","_rev":"1-e62062299c8f46774558393656f6c42b","student_sk":1,"degree_sk":1,"course_sk":2,"date_sk":2,"pass":"Pass","course_name":"Machine Learning","degree_name":"BSc in Computer Science","examdate":"2022-01-12"}},
{"id":"5","key":"Machine Learning","value":{"_id":"5","_rev":"1-f7c1b0c58b3f45d4490b5e5ae14fca01","student_sk":3,"degree_sk":2,"course_sk":2,"date_sk":2,"pass":"Fail","course_name":"Machine Learning","degree_name":"BSc in Computer Science(International)","examdate":"2022-01-12"}},
{"id":"11","key":"Programming Paradigms","value":{"_id":"11","_rev":"1-426048ba8bc2a99184eb155824e78d21","student_sk":4,"degree_sk":3,"course_sk":6,"date_sk":6,"pass":"Fail","course_name":"Programming Paradigms","degree_name":"BSc in Computer Science(Infrastructure)","examdate":"2022-06-13"}}
```

Map Functions

All map functions have a single parameter doc.

- This is a single document in the database.
  - In our example the map function checks whether our document has a course_name attribute
- It then calls the built-in **emit()** function with that attribute as an argument
- **emit()** takes 2 arguments
  - The first is key, the second is value
  - This creates an entry in our View Result
  - It can be called multiple times in a map function to create multiple entries in the view result from a single document



CouchDB uses a B-tree storage engine.

- B-tree -> a sorted data structure that allows for searches, insertions, and deletions in logarithmic time

Source:
https://guide.couchdb.org/editions/1/en/consistency.html

How does it work?

When you query your view, CouchDB takes the source code and runs it for you on every document in the database.

If you have a lot of documents, that takes quite a bit of time

- CouchDB is designed to avoid any extra costs: it only runs through all documents once, when you first query your view.
  - If a document is changed, the map function is only run once, to recompute the keys and values for that single document.

The view result is stored in a B-tree, just like the structure that is responsible for holding your documents.

- View B-trees are stored in their own file.
- The B-tree provides very fast lookups of rows by key, as well as efficient streaming of rows in a key range. In our example, a single view can answer all questions that involve time: “Give me all the blog posts from last week” or “last month” or “this year.”

MapReduce

Reduce function operates on rows emitted from the map function

Suppose for our course data we wanted to count how many students were taking a course

We create a design document `coursematch`

MapReduce

We define a map function to emit course names:

```
if (doc.course_name) {
    emit(doc.course_name, 1);
  }
```

We define a reduce function:

```
sum(values) – we use _sum
```

MapReduce

To create the design document we need to ensure that CouchDB knows it is a view:

```
{  "_id": "_design/course_counts",
  "views": {    "course_count":
{     "map": "function(doc) { if (doc.course_name) { emit(doc.course_name, 1); } }",
"reduce": "_sum"    }  }
}
```

MapReduce

We can execute the view in a Browser

http://admin:couchdb@127.0.0.1:5984/students/_design/course_counts/_view/course_count

Showing the groupings per course

http://admin:couchdb@127.0.0.1:5984/students/_design/course_counts/_view/course_count?group=true

Consistency

The CouchDB file layout and commitment system features all **Atomic Consistent Isolated Durable (ACID) properties.**

- On-disk, CouchDB never overwrites committed data or associated structures, ensuring the database file is always in a consistent state.

Single document updates (add, edit, delete) are all or nothing, either succeeding entirely or failing completely.

- The database never contains partially saved or edited documents.

Consistency

The CouchDB document update model is lockless and optimistic.

- Document edits are made by client applications loading documents, applying changes, and saving them back to the database.
- If another client editing the same document saves their changes first, the client gets an edit conflict error on save.
- To resolve the update conflict
  - The latest document version can be opened, the edits reapplied and the update tried again.