

# CouchDB



Presented By

Team 9:

Donita Almeida

Dhiraj Gurnani

George Zachariah

Ritika Shetty

Saina Patel

# Introduction

- Why CouchDB ?
  - High availability is important
  - Eventually consistent
  - Powerful data processing using the built-in query engine
  - Modular & Scalable design
  - User friendly

# Features

- A NoSQL database
- Uses JSON for documents
- Uses JavaScript for MapReduce indexes
- Uses HTTP for it's API
- Documents can have attachments
- Incremental peer-peer replication

# SQL vs CouchDB

<b>SQL</b>	<b>CouchDB</b>
Relational	Non-Relational
Tables	Documents with types
Rows and Columns	Document Fields
SQL Query Engine	Map / Reduce Engine

# CouchDB Core API (Command Line Utility )

- Server API
- Database API
- Document API
- Replication API

# Server API (Access via CURL)

- CouchDB runs on port 5984
- `curl -X GET http://127.0.0.1:5984/`
- returns the server information

```
eg:{
  "couchdb": "Welcome",
  "uuid": "85fb71bf700c17267fef77535820e371",
  "vendor": {
    "name": "The Apache Software Foundation",
    "version": "1.4.0"
  },
  "version": "1.4.0"
}
```

# Database API

- `curl -X GET http://127.0.0.1:5984/demo`
- Returns the information about database demo

```
{  
  "compact_running" : false,  
  "doc_count" : 0,  
  "db_name" : "demo",  
  "purge_seq" : 0,  
  "committed_update_seq" : 0,  
  "doc_del_count" : 0,  
  "disk_format_version" : 5,  
  "update_seq" : 0,  
  "instance_start_time" : "1306421773496000",  
  "disk_size" : 79  
}
```

# Database API (Contd..)

Create a database

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

Delete a database

- `curl -X DELETE http://127.0.0.1:5984/baseball`



# Document API via CURL

- curl -X **PUT** <http://127.0.0.1:5984/albums>

Create a document

- curl -X **PUT** http://127.0.0.1:5984/albums/1000  
-d '{"title":"Abbey Road","artist":"The Beatles"} '

Retrieve a document

- curl -X **GET** <http://127.0.0.1:5984/albums/1000>

# Document API (Contd..)

- **\_rev** - If you want to update or delete a document, CouchDB expects you to include the **\_rev** field of the revision you wish to change
- `curl -X PUT http://127.0.0.1:5984/albums/1000 -d '{"_rev":"1-42c7396a84eaf1728cdbf08415a09a41","title":"Abbey Road", "artist":"The Beatles","year":"1969"}'`

# Availability and Partition Tolerance

- Every Node in the network is a Master
- CouchDB is highly available and eventually consistent
- For making the System Highly available, couchdb has used two concepts:
  1. Key to your Data
  2. Multi – version Concurrency Control

# Key To Your Data

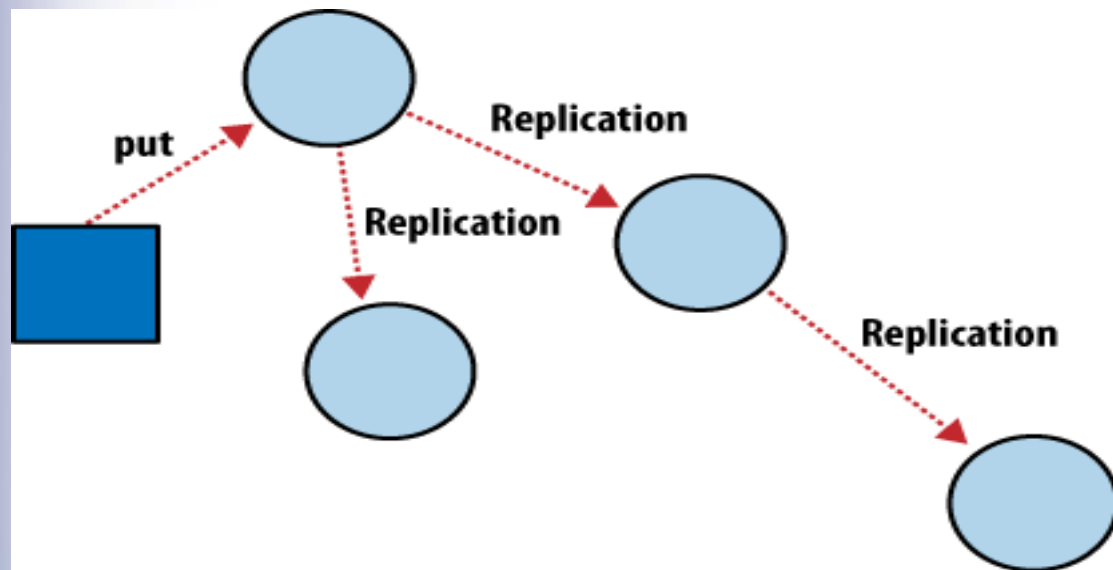
- B-tree storage engine
- This storage engine allows the data to be stored in sorted order based on the keys assigned to each document.
- Allows to perform search, insertion and deletion in logarithmic time.

# Multi Version Concurrency Control

- In relational database, locks are used to ensure that a row is updated by only 1 user at a time. But this approach waste a lot of time.
- To avoid it CouchDB uses MVCC, where different versions of same document are created.
- i.e. update on documents create a new version at some other place and old version is also present.

# Eventual Consistency

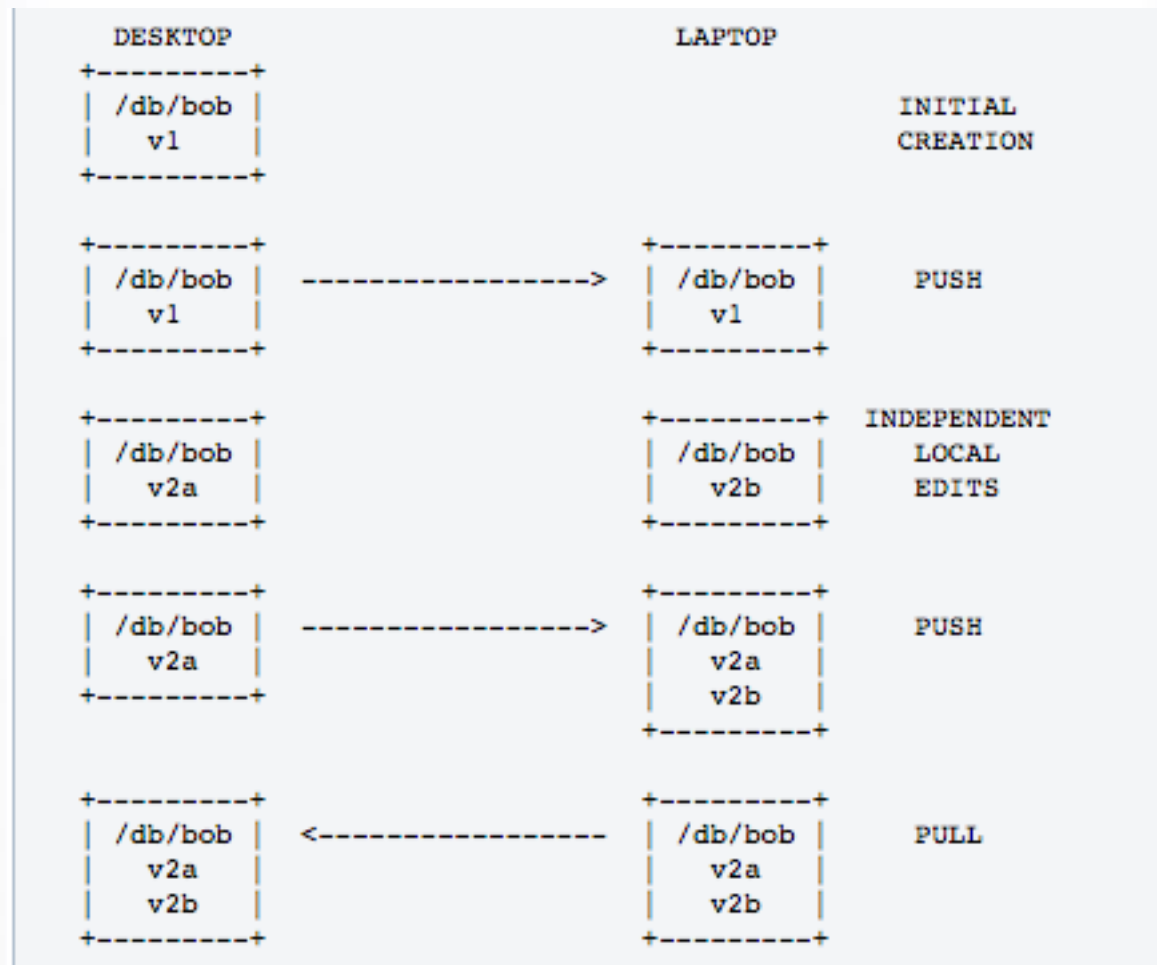
- It uses incremental replication, where document changes is periodically communicated between servers.
- With this approach, the servers need not be in constant connection.



# Replication API via curl

- Command to replicate a Database :
- `curl -X POST http://127.0.0.1:5984/\_replicate -d '{"source":"http://example.org/database", "target":"http://admin:password@127.0.0.1:5984/database"}' -H "Content-Type: application/json"`
- This is unidirectional command. To make the replication bydirectional, we call the same command by swapping source and target values.

# Replication





# Continuous Replication

- `curl -X POST http://127.0.0.1:5984/\_replicate -d '{"source":"db", "target":"db-replica", "continuous":true}' -H "Content-Type: application/json"`
- It will keep listening to `\_changes` API and replicate any missing documents to the target.

# Revision Tree

```
      ,--> r2a -> r3a -> r4a
r1  --> r2b -> r3b
     `--> r2c -> r3c
```

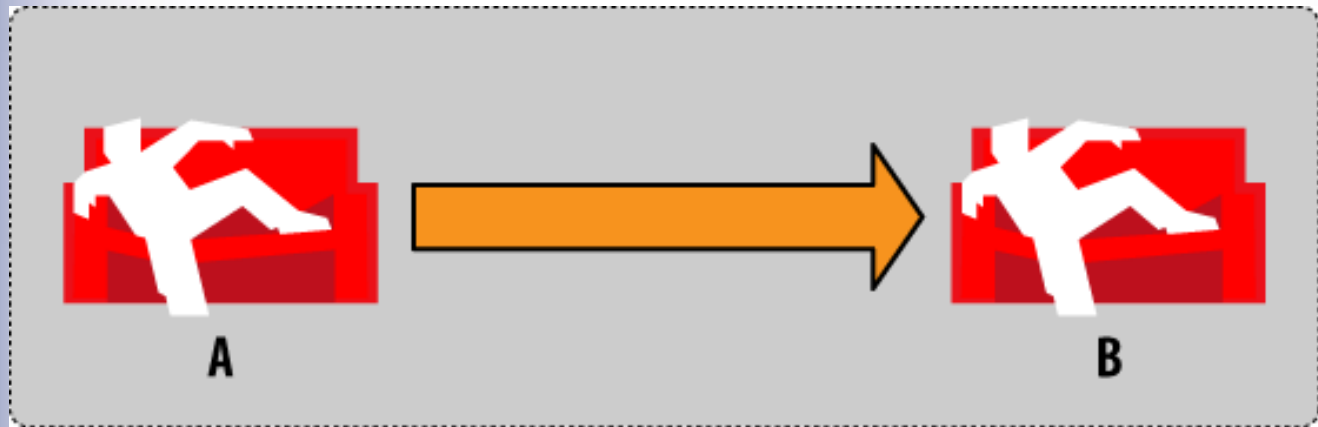
- To access a particular revision of a document
  - GET /somedatabase/some\_doc\_id?rev=946B7D1C  
HTTP/1.
- To get information about which revisions are present
  - GET /somedatabase/some\_doc\_id?revs\_info=true  
HTTP/1.0

# CouchDB Conflict Resolution

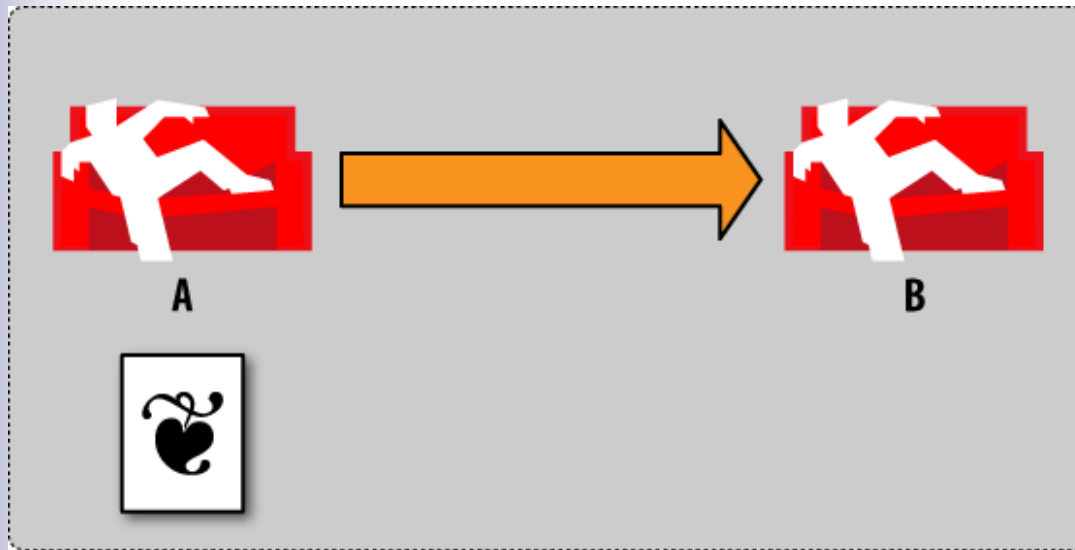
- How does CouchDB's replication system deal with conflicts?
  1. Flags the affected document with a special attribute `"_conflicts": true`
  2. Determines which of the changes will be stored as the latest revision (winning revision).
  3. The losing revision gets stored as the previous revision

# CouchDB Conflict Resolution

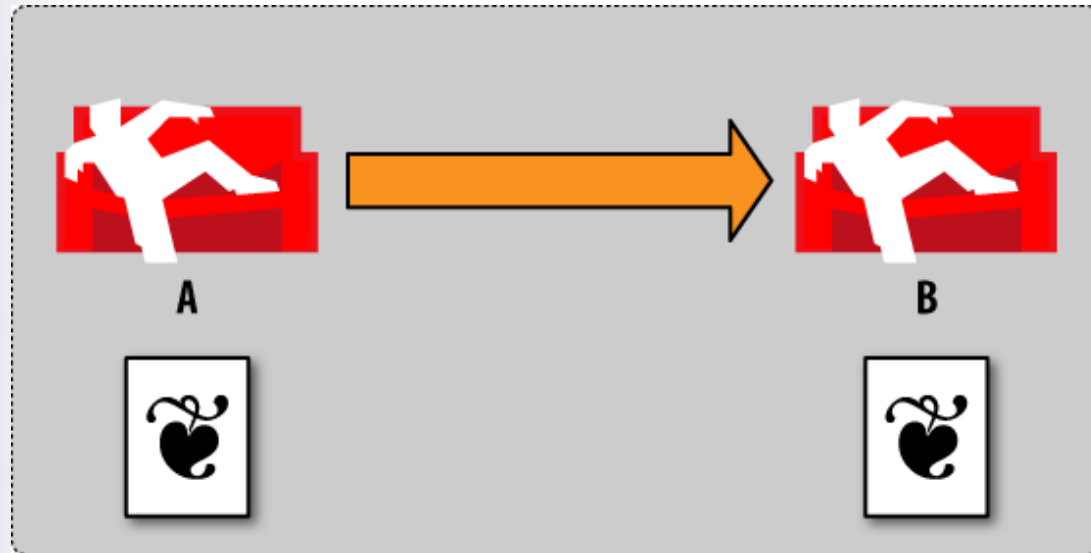
1. We have two CouchDB databases A and B, and we are replicating from A to B.



2. We create a document in database A. Database B won't know about the new document for now.

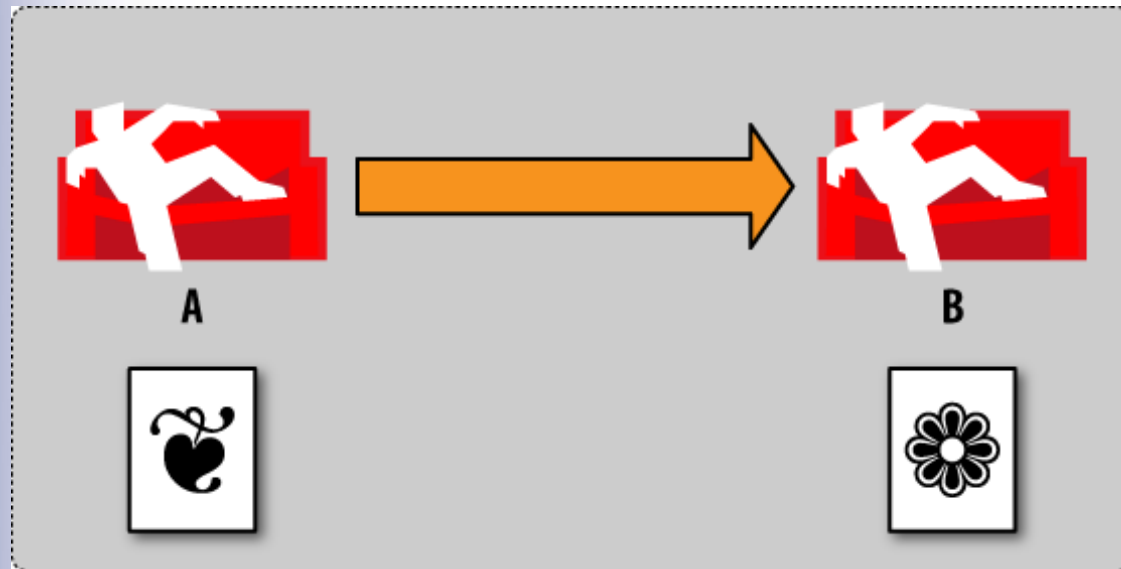


3. We now trigger replication and tell it to use database A as the source and database B as the target



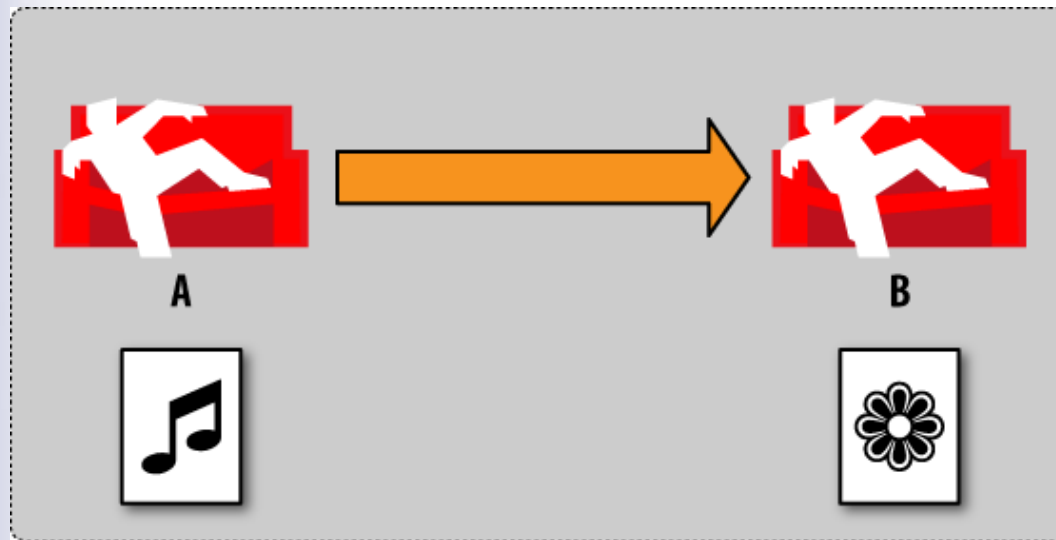
4. Now we go to database B and update the document.

Upon change, CouchDB generates a new revision for us.



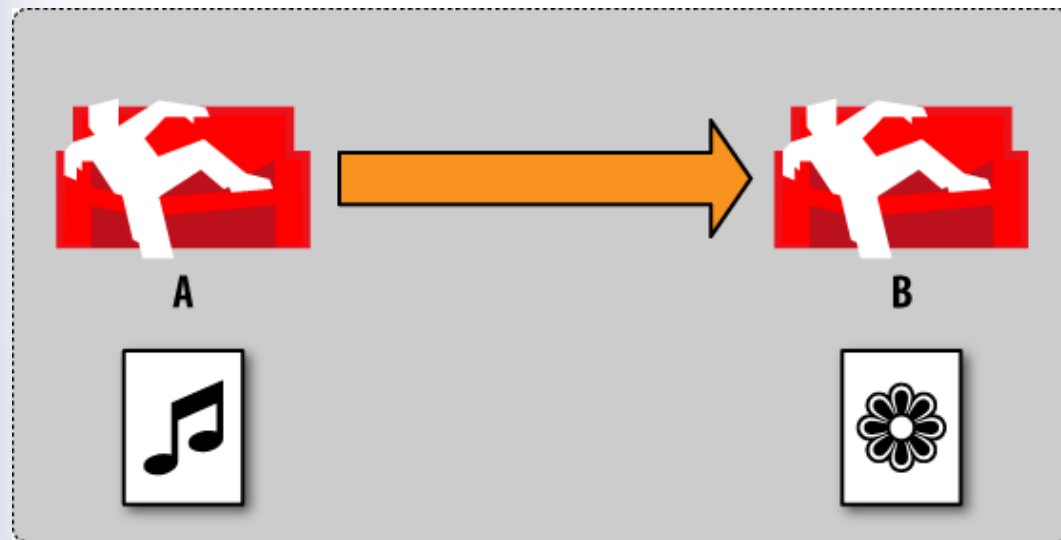
5. Now we make a change to our document in database A by changing some other values.

There are two different revisions of that same document in each database.

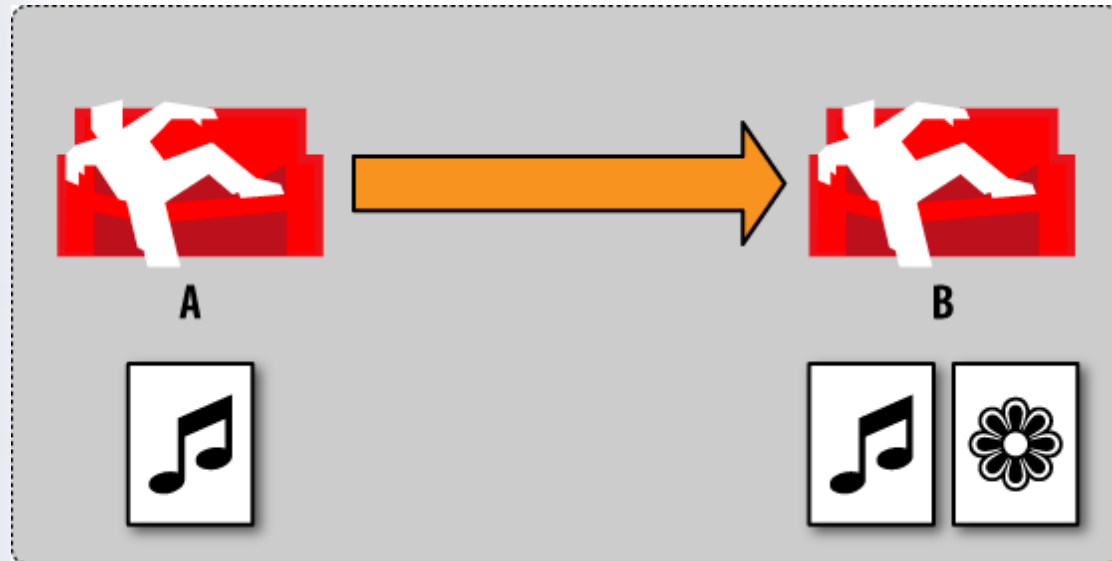




6. Now we trigger replication again from database A to database B as before

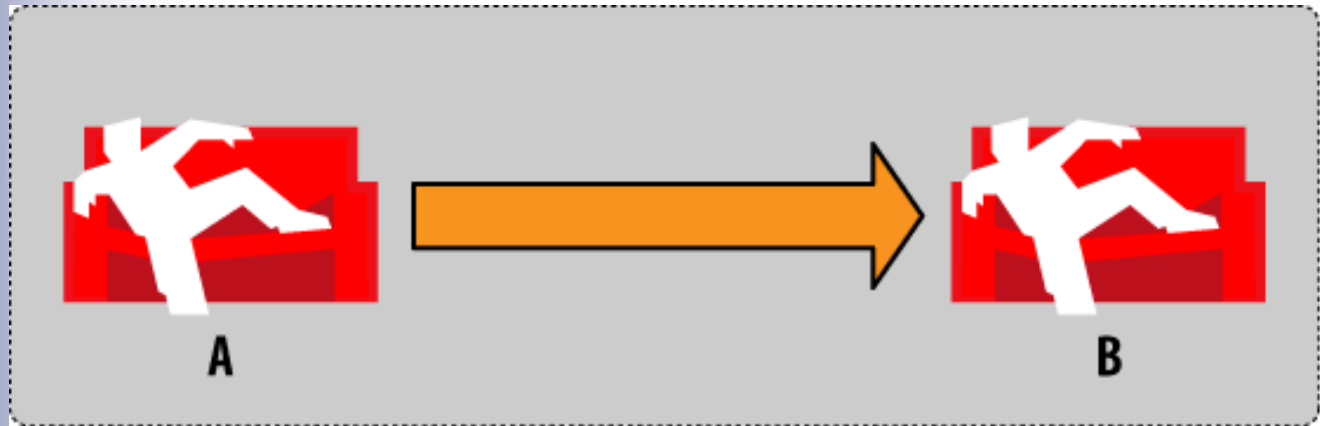


7. When replicating two different revisions are detected for the same document, and it creates a conflict .



8. Finally, we tell CouchDB which version we want as the latest revision by resolving the conflict.

Now both databases have the same data.



# Couch DB Views

## Why views?

No tables and collection.

So views

- **View server execute functions**

1. Map – used to display a view
2. Reduce(optional) – which is used to create a sorted view

# Example Database

```
{  
  "_id": "album1",  
  "artist": "Megadeth",  
  "title": "Endgame",  
  "year": 2010  
}  
  
{  
  "_id": "album2",  
  "artist": "Slayer",  
  "title": "World Painted Blood",  
  "year": 2009  
}  
  
{  
  "_id": "album3",  
  "artist": "Arcturus",  
  "title": "Sideshow Symphonies",  
  "year": 2005  
}
```

```
{  
  "_id": "album4",  
  "artist": "Pantera",  
  "title": "Reinventing the Steel",  
  "year": 2009  
}
```

```
{  
  "_id": "album5",  
  "artist": "Slayer",  
  "title": "South of Heaven",  
  "year": 2009  
}
```

# Map and Reduce functions

```
"_id": "_design/foobar",
"language": "javascript",
"views": {
  "by_year": {
    "map": "function(doc) {
      if (doc.year) {
        emit(doc.year, 1);
      }
    }",
    "reduce": "function(keys, values, rereduce) {
      return sum(values);
    }"
  }
}
```

```
$ curl
```

```
http://localhost:5984/albums/\_design/foobar/view/by\_year
```

```
{
```

```
  "update_seq": 6,
```

```
  "rows": [
```

```
    {"key": null, "value": 5}
```

```
  ]
```

```
}
```



```
$ curl http://localhost:5984/albums/_design/foobar/_view/by_year?group=true
```

```
{
  "update_seq": 6,
  "rows": [
    {"key": 2005, "value": 1},
    {"key": 2009, "value": 3},
    {"key": 2010, "value": 1}
  ]
}
```

```
$ curl 'http://localhost:5984/albums/_design/foobar/_view/by_year?
group=true&startkey=2009&endkey=2010'
```

```
{
  "update_seq": 6,
  "rows": [
    {"key": 2009, "value": 3},
    {"key": 2010, "value": 1}
  ]
}
```

```
$ curl \
http://localhost:5984/albums/_design/foobar/_view/by_
year?reduce=false
```

```
{
  "update_seq":6,
  "rows": [
    {"id": "album3", "key": 2005, "value": 1},
    {"id": "album2", "key": 2009, "value": 1},
    {"id": "album4", "key": 2009, "value": 1},
    {"id": "album5", "key": 2009, "value": 1},
    {"id": "album1", "key": 2010, "value": 1}
  ]
}
```

# Limitations

- Temporary views on large datasets are very slow.
- Replication of large databases may fail
- Documents are quite large as the data is represented using "JSON" format
- "Only" eventual consistency.
- Couch maintains a different document for every update you make this fills up your hard disk fast

# Compaction

- DB/view files are written in append mode
- Will continue to grow indefinitely
- A DB or View compaction operation can be triggered
- `curl -X POST`  
`http://127.0.0.1:5984/albums/compact`
- `curl -X POST`  
`http://127.0.0.1:5984/albums/_design/view`

# What Compaction does?

- i. Creates a new file
- ii. Traverses the DB or View B-Tree and lookups the most recent data pointed by each node
- iii. Writes that most recent data to the new file
- iv. deletes the original file and renames the compacted file to the original DB/View file name.

# Who uses CouchDB?



# Document Creation

The screenshot displays the Apache CouchDB Futon web interface in a browser. The address bar shows the URL: `127.0.0.1:5984/_utils/document.html/reducedemo/869e3d7c5034f7c1d0a45ce3b3034b3a`. The breadcrumb navigation indicates the path: `Overview > reducedemo > 869e3d7c5034f7c1d0a45ce3b3034b3a`.

At the top, there are action buttons: `Save Document` (checked), `Add Field`, `Upload Attachment...`, and `Delete Document...`. Below these are tabs for `Fields` and `Source`, with `Source` being the active tab.

The main content area, titled `Source`, contains a JSON document in a yellow editor. The document is:

```
{
  "_id": "869e3d7c5034f7c1d0a45ce3b3034b3a",
  "_rev": "3-92c1b32a871fdd7b294b5ec98062681a",
  "title": "The Hobbit",
  "languages": [
    "en",
    "fr",
    "de",
    "de",
    "de",
    "fr",
    "fr",
    "en",
    "fr",
    "en"
  ]
}
```

A tooltip `Double click to edit` is visible over the `"title"` field. At the bottom of the editor, it says `Showing revision 3 of 3` with links for `Previous Version` and `Next Version`.

On the right sidebar, the CouchDB logo with the tagline `relax` is at the top. Below it are sections for `Tools` (Overview, Configuration, Replicator, Status), `Documentation` (Manual), `Diagnostics` (Verify Installation), and `Recent Databases` (listing `_replicator`, `democc`, `reducedemo`, `test_suite_db`, `testreplicate`, and `testrohan`, with `reducedemo` highlighted).

At the bottom of the sidebar, a message reads: `Welcome to Admin Party! Everyone is admin. Fix this`. Below that, it says `Futon on Apache CouchDB 1.6.1`.

The Windows taskbar is visible at the very bottom, showing the search bar and various application icons. The system clock in the bottom right corner indicates `8:43 AM 4/16/2016`.

# Map View

The screenshot displays the Apache CouchDB web interface in a browser. The address bar shows the URL `127.0.0.1:5984/_utils/database.html?reducedemo/_temp_view`. The page title is "Overview > reducedemo".

At the top, there are navigation links: "New Document", "Security...", "Compact & Cleanup...", and "Delete Database...". On the right, there are controls for "Jump to: [Document ID]", "View: [Temporary view...]", and "Stale views".

The main content area is divided into two sections. The left section, titled "View Code", contains a "Map Function" editor with the following JavaScript code:

```
function(doc) {  
  if (doc.languages) {  
    for (var language in doc.languages) {  
      emit(doc.languages[language], 1);  
    }  
  }  
}
```

The right section, titled "Reduce Function (optional)", is currently empty. Below the code editors are buttons for "Run", "Language: javascript", "Revert", "Save As...", and "Save".

A warning message states: "Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application."

Below the warning is a table with columns "Key" and "Value". The "Key" column is set to "Grouping: exact". The table contains 11 rows, each with a key (e.g., "fz", "fz+", "fz+", "fz+", "fz+", "fz+", "fz+", "fz+", "fz+", "fz+", "fz+") and a value of 1. The table is paginated, showing "Showing 1-11 of 11 rows".

At the bottom of the table, it says "View request duration: 00:00:00.021".

On the right side of the interface, there is a sidebar with the CouchDB logo and a list of links: "Tools" (Overview, Configuration, Replicator, Status), "Documentation" (Manual), "Diagnostics" (Verify Installation), and "Recent Databases" (test\_suite\_db, testreplicate, testrohan). At the bottom of the sidebar, there is a "Welcome to Admin Party!" message and a link to "Fix this".

The bottom of the screenshot shows a Windows taskbar with various application icons and a system clock indicating 8:45 AM on 4/16/2016.



# Map/reduce View

The screenshot shows the Apache CouchDB web interface. The browser's address bar displays the URL `127.0.0.1:5984/_utils/database.html?reducedemo/_temp_view`. The page title is "Overview > reducedemo".

At the top, there are navigation links: "New Document", "Security...", "Compact & Cleanup...", "Delete Database...", "Jump to: [Document ID]", "View: Temporary view...", and "Stale views".

The "View Code" section is expanded, showing the Map Function and the optional Reduce Function. The Map Function is written in JavaScript:

```
function(doc) {
  if (doc.languages) {
    for (var language in doc.languages) {
      emit(doc.languages[language], 1);
    }
  }
}
```

The Reduce Function (optional) is:

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

Below the code, there is a "Run" button, a "Language" dropdown set to "javascript", and buttons for "Revert", "Save As...", and "Save".

A warning message states: "Warning: Please note that temporary views are not suitable for use in production, as they are really slow for any database with more than a few dozen documents. You can use a temporary view to experiment with view functions, but switch to a permanent view before using them in an application."

The results table shows the following data:

Key	Grouping: exact	Value	Reduce
"fr"		4	
"en"		3	
"de"		4	

Below the table, it says "Showing 1-3 of unknown rows". There are navigation links for "Previous Page", "Rows per page: 25", and "Next Page".

The "View request duration: 00:00:00.030" is displayed.

On the right side, there is a sidebar with the CouchDB logo and a list of links: "Tools" (Overview, Configuration, Replicator, Status), "Documentation" (Manual), "Diagnostics" (Verify Installation), and "Recent Databases" (listing several databases, with "reducedemo" highlighted).

The bottom of the image shows the Windows taskbar with the search bar and various application icons.

# Replication

The screenshot shows the Apache CouchDB Replicator interface in a web browser. The browser's address bar displays the URL `127.0.0.1:5984/_utils/replicator.html`. The interface has a dark header with 'Overview' and 'Replicator' tabs. The main content area is titled 'Replicate changes from:' and 'to:'. Under 'Replicate changes from:', the 'Local Database' is set to 'reducedemo' (selected with a radio button) and the 'Remote database' is 'http://'. Under 'to:', the 'Local database' is 'testreplicate' (selected with a radio button) and the 'Remote database' is 'http://'. A double-headed arrow icon is between the two sections. A 'Continuous' checkbox is checked, and a 'Replicate' button is to its right. Below this, an 'Event' section shows a log entry: `{"ok":true,"_local_id":"59d141227428c64f134ee8fe9894008d+continuous"}`. Below the log, it says 'No replication'. On the right side, there is a sidebar with the CouchDB logo (a red silhouette of a person jumping) and the text 'CouchDB relax'. Below the logo, there are sections for 'Tools' (Overview, Configuration, Replicator, Status), 'Documentation' (Manual), 'Diagnostics' (Verify Installation), and 'Recent Databases' (listing \_replicator, democc, reducedemo, test\_suite\_db, testreplicate, and testrol). At the bottom of the sidebar, there is a message: 'Welcome to Admin Party! Everyone is admin. [Fix this](#)'. The footer of the sidebar says 'Futon on Apache CouchDB 1.6.1'. The Windows taskbar at the bottom shows the search bar and various application icons. The system clock in the bottom right corner shows '8:47 AM 4/16/2016'.

## Demo for CouchDB on EC2

Start couchDB:

`sudo service couchdb start`

```
ec2-user@ip-172-31-38-41:~$ sudo service couchdb start
Starting database server couchdb
[ec2-user@ip-172-31-38-41 ~]$
```

View all DBs:

`curl -X GET http://127.0.0.1:5984/_all_dbs`

```
ec2-user@ip-172-31-38-41:~$ curl -X GET http://127.0.0.1:5984/_all_dbs
[{"_replicator", "_users"}]
[ec2-user@ip-172-31-38-41 ~]$
```

Create a DB:

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/_all_dbs  
{"_replicator": "users"}  
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test  
{"ok": true}  
[ec2-user@ip-172-31-38-41 ~]$
```

create a document:

```
curl -X PUT http://127.0.0.1:5984/my_database/"001" -d'{ " Name " : " Raju " , " age " : " 23 " , "  
Designation " : " Designer " }'
```

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test/"001" -d'{ " Name " : " Raju " , " age " : " 23 " , " Designation " : " Designer " }'  
{"ok": true, "id": "001", "rev": "1-8b0b70974c6dd43edf4e07f12a9655af"}  
[ec2-user@ip-172-31-38-41 ~]$
```

retrieve a document:

curl -X GET [http://127.0.0.1:5984/my\\_database/001](http://127.0.0.1:5984/my_database/001)

```
ec2-user@ip-172-31-38-41:~$ curl -X PUT http://127.0.0.1:5984/test/001 -d '{ " Name " : " Raju " , " age " : " 23 " , " Designation " : " Designer " }'
{"ok":true,"id":"001","rev":"1-8b0b70974c6dd43edf4e07f12a9655af"}
ec2-user@ip-172-31-38-41 ~$ curl -X GET http://127.0.0.1:5984/test/001
{"_id":"001","rev":"1-8b0b70974c6dd43edf4e07f12a9655af"," Name ":" Raju "," age ":" 23 "," Designation ":" Designer "}
```

curl -X GET [http://127.0.0.1:5984/my\\_database/\\_all\\_docs](http://127.0.0.1:5984/my_database/_all_docs)

```
ec2-user@ip-172-31-38-41:~$ curl -X GET http://127.0.0.1:5984/test/_all_docs
{"total_rows":2,"offset":0,"rows":[{"_id":"001","key":"001","value":{"rev":"1-8b0b70974c6dd43edf4e07f12a9655af"}}, {"_id":"002","key":"002","value":{"rev":"1-63c0fcc1756f1a54808aa2d5d511dd1a"}}]}
```

update a document:

```
curl -X PUT http://127.0.0.1:5984/test/001/ -d' { " age " : " 24 " , "_rev" : "1-8b0b70974c6dd43edf4e07f12a9655af" } '
```

```
ec2-user@ip-172-31-38-41:~$ curl -X PUT http://127.0.0.1:5984/test/001/ -d' { " age " : " 25 " , "_rev" : "1-8b0b70974c6dd43edf4e07f12a9655af" } '
{"ok":true,"id":"001","rev":"2-7e7722e997338c4734360f67c4fe6452"}
ec2-user@ip-172-31-38-41 ~]$
```

GET revision info:

```
curl -X GET http://127.0.0.1:5984/my\_database/001?revs\_info=true
```

```
ec2-user@ip-172-31-38-41:~$ curl -X GET http://127.0.0.1:5984/test/001?revs_info=true
{"_id":"001","_rev":"2-7e7722e997338c4734360f67c4fe6452","age":" 25 " , "_revs_info":[{"rev":"2-7e7722e997338c4734360f67c4fe6452","status":"available"}, {"rev":"1-8b0b70974c6dd43edf4e07f12a9655af","status":"available"}]}
ec2-user@ip-172-31-38-41 ~]$
```

delete a document:

curl -X DELETE <http://127.0.0.1:5984/test/001?rev=1-3fcc78daac7a90803f0a5e383f4f1e1e>

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test/001?rev=2-7e7722e997338c4734360f67c4fe6452  
{"ok":true,"id":"001","rev":"3-5f51c51c8594f18517bab328ad4da0e4"}  
[ec2-user@ip-172-31-38-41 ~]$
```

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test/001?rev=2-7e7722e997338c4734360f67c4fe6452  
{"ok":true,"id":"001","rev":"3-5f51c51c8594f18517bab328ad4da0e4"}  
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test/001?revs_info=true  
{"error":"not_found","reason":"deleted"}  
[ec2-user@ip-172-31-38-41 ~]$
```

Replicate a DB:

create database test2:

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test2  
{"ok":true}  
[ec2-user@ip-172-31-38-41 ~]$
```

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X PUT http://127.0.0.1:5984/test2  
{"ok":true}  
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/test2/_all_docs  
{"total_rows":0,"offset":0,"rows":[]}  
}  
[ec2-user@ip-172-31-38-41 ~]$
```



```
curl -H "Content-Type: application/json" -X POST http://127.0.0.1:5984/_replicate -d
{"source":"test","target":"test2"}
```

```
ec2-user@ip-172-31-38-41:~$ curl -X PUT http://127.0.0.1:5984/test2
{"ok":true}
ec2-user@ip-172-31-38-41:~$ curl -X GET http://127.0.0.1:5984/test2/_all_docs
{"total_rows":0,"offset":0,"rows":[]
}
ec2-user@ip-172-31-38-41:~$ curl -H "Content-Type: application/json" -X POST http://127.0.0.1:5984/_replicate -d '{"source":"test","target":"test2"}'
{"ok":true,"session_id":"5907b30061fab49de3efb8e04252d7c2","source_last_seq":4,"replication_id_version":3,"history":[{"session_id":"5907b30061fab49de3efb8e04252d7c2","s
tart_time":"Sat, 16 Apr 2016 03:30:03 GMT","end_time":"Sat, 16 Apr 2016 03:30:03 GMT","start_last_seq":0,"end_last_seq":4,"recorded_seq":4,"missing_checked":2,"missing_
found":2,"docs_read":2,"docs_written":2,"doc_write_failures":0}]
ec2-user@ip-172-31-38-41:~$
```

```
ec2-user@ip-172-31-38-41:~$ curl -X GET http://127.0.0.1:5984/test2/_all_docs
{"total_rows":1,"offset":0,"rows":[{"id":"002","key":"002","value":{"rev":"1-63c0fcc1756f1a54808aa2d5d51dd1a"}}
]}
ec2-user@ip-172-31-38-41:~$
```

Delete a database:

```
ec2-user@ip-172-31-38-41:~  
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test  
{"ok":true}  
[ec2-user@ip-172-31-38-41 ~]$ curl -X DELETE http://127.0.0.1:5984/test2  
{"ok":true}  
[ec2-user@ip-172-31-38-41 ~]$ curl -X GET http://127.0.0.1:5984/_all_dbs  
["replicator","users"]  
[ec2-user@ip-172-31-38-41 ~]$
```