Implementation of CouchDBViews

Subita Kumari and Pankaj Gupta

Abstract Flexible data model and horizontal scalability are the need of contemporary era to handle huge heterogeneous data. This has lead to the popularity of NoSQL Databases. CouchDB is an admired and easy to use choice among NoSQL Document-Oriented databases. CouchDB is developed in Erlang language. CouchDB's RESTful (Representational State Transfer) APIs (Application Programming Interface) make it special because they allow database access through http (Hyper Text Transfer Protocol) requests. This access in the form of HTTP requests is achieved with the help of command line utility Curl. The Futon, web-based utility of CouchDB, is also used to manage documents, databases, and replication in CouchDB. CouchDB uses a special type of system for querying data than traditional RDBMS (Relational Database Management Systems) i.e. views. This paper explains various unique features of CouchDB which distinguish it from RDBMS. It also includes implementation of temporary and permanent views using MapReduce.

Keywords CouchDB · MapReduce · Views · Futon · Curl

1 Introduction

The database technologies have been evolving at a very fast pace. Database systems are moving from SQL to NoSQL systems to handle datasets of diverse categories. Most NoSQL systems have been developed with the general goal of offering simple operations on flexible data structures. Indeed, they are focused to provide massive throughput and high scalability [1]. NoSQL databases support large data volumes,

S. Kumari (✉)
Computer Science and Engineering, UIET, MDU, Rohtak, India
e-mail: subita.hooda@gmail.com

P. Gupta
Computer Science and Engineering, VCE, Rohtak, India
e-mail: pankajgupta.vce@gmail.com

simple and easy copy, eventual consistency and simple API. NoSQL databases are becoming the core phenomena for big data applications. NoSQL databases are broadly classified into three categories: document stores databases, key-value stores databases and column-oriented databases [2]. Today most used document-oriented databases are MongoDB and CouchDB. This paper explores CouchDB documents, databases and views in detail and compares with traditional RDBMS basic architecture. Document-oriented databases provide the alternative of a *'row'* of traditional RDBMS in the form of a more flexible *'document'*. They permit access to data values of database via its content. Documents are hierarchical structures that can be in BSON, JSON, or other formats like XML and therefore consist of more composite elements such as collections, maps or scalar values [3]. There is no need to define schema beforehand means there is a flexibility of defining the size/type of key/value fields of documents on the runtime. This makes it easier to perform operations like insertion and deletion of fields whenever required. Each document is collection of set of fields and is associated with a unique identifier, for indexing and querying purpose.

2 CouchDB Experimental Setup

Developed and maintained by the Apache Software Foundation, CouchDB is an admired and easy to use choice among NoSQL Document-Oriented databases. CouchDB is developed in Erlang language. CouchDB's RESTful APIs make it special because they allow database access through HTTP requests [4]. The software required for the CouchDB setup is:

- CouchDB Version 1.6.1
- Web-Based Interface Futon
- Command Line Utility Curl.

CouchDB makes use of APIs by using the command-line utility *curl*. A very interesting fact is that users are provided insights of the database and great control on HTTP requests by *curl*. Suppose CouchDB is running and the following request is sent to database via curl

```
curl http://127.0.0.1:5984[5]
```

The above curl command will seek an http GET from CouchDB and will return the detailed JSON object as follows:

```
{"couchdb":"Welcome","uuid":"6c200d933f15fda6543e960e874796a8","version":
"1.6.1","vendor":{"version":"1.6.1","name":"The Apache Software
Foundation"}}
```

CouchDB is giving a welcome message with the running version number. Another way to interact with CouchDB is using Futon. It provides default web interface for administration of CouchDB. A full access to features of CouchDB is provided more conveniently by Futon. It provides us the facility to create and remove databases; create, delete, view and modify documents; replicate databases; and compose and run MapReduce views. To load Futon the following link is used in the browser application.

`http://127.0.0.1:5984/_utils/ [5].`

The remaining sections explain the other distinctive features of CouchDB experimentally under above set up.

3 CouchDB Features

3.1 Data Model

CouchDB is composed of databases. Databases are a further collection of variable-schema documents. As opposed to this, RDBMSs are a collection of fixed schema relations which are composed of rows. CouchDB stores JSON (Java Script Object Notation) documents in a binary format. The documents stored fields in the form of key-value pairs. Every document is identified by a unique field "_id". This "_id" may be automatically generated by CouchDB or manually provided by the user. There is no maximum limit for key-value pairs and size of documents. The file extension of database files is .couch. The data model is shown in Fig. 1 [4].

3.2 RESTful API

One of the very outstanding features of CouchDB is its RESTful API. REST architecture explains how to provide services for interaction among machines. REST facilitates the use of HTTP requests for carrying out basic Create, Read,

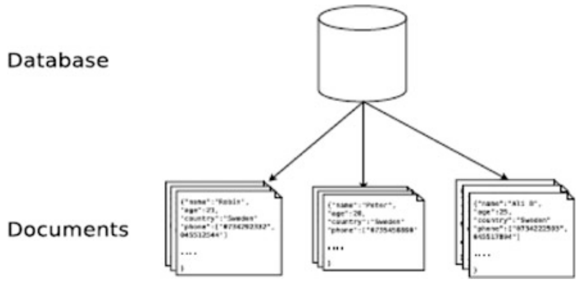


Fig. 1 CouchDB data model

Update and Delete operations (CRUD) on documents [4]. Following HTTP methods are used for performing basic CRUD operations-

- PUT—Create databases/documents/attachments
- POST—Update databases/documents/attachments
- GET—Read databases/documents/attachments
- DELETE—Delete databases/documents/attachments

RESTful architecture gives unique identifiers to databases/documents/attachments etc. in the form of URIs. Suppose a database named "university" is being created on a local CouchDB setup. Using the command-line utility curl following statement will do the needful:

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

The success of the above statement in curl will be replied back by CouchDB in the form of a JSON document, e.g. the creation of database named "university" is notified by:

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

All other basic operations on documents (CRUD) can be performed with similar curl statements.

3.3 Documents

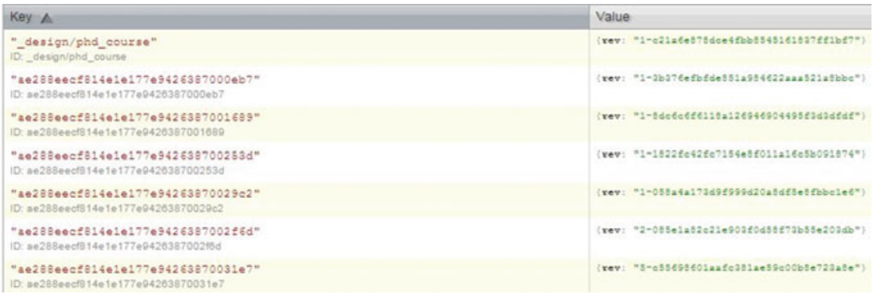
Documents are the fundamental data structure of CouchDB. It is just like a physical document—such as an invoice, an address card, or an invitation card [6]. Documents use the dynamic schema as opposed to relations of RDBMS which use fixed schema. The document contains unique _id and _rev field showing id and revision number of document. Documents are stored by CouchDB using the JSON format in the form of key-value pair. The example below shows one such document of PhD student of the "university" database. "_id" field uniquely identifies the document in the "university" database and "_rev" field shows the version number of the document. Every time a document is updated a new version of the document is created and older versions also remain on the disk. The example below shows the 7th version of the document shown.

```
{
    "_id": "ae288eecf814e1e177e94263870031e7",
    "_rev": "7-de540398f44262de6a80b0d67d72df9d",
    "name": "Reet",
    "age": 25,
    "doj": "15/06/2015",
    "branch": "Electronics",
    "topic": "Wireless Communication",
    "supervisor": "Deepak",
    "course": "Phd",
    "Marks": {
         "Digital": 88,
         "MicroProcessor": 79,
         "Analog": 62
    }
}
```

The example database "university" contains some other documents containing details of UG, PG and PhD students having different schemas as well as a design document "phd_course" implementing permanent views. A design document is identified by word "_design". Figure 2 shows the snapshot of documents of "university" database from Futon.

3.4 Revisions

CouchDB supports well-built versioning system. It means that when some changes are carried out by user in a document, the next version is created by the CouchDB and it does not overwrite the existing document. CouchDB keeps all versions so that user can later on track earlier amendments. This feature makes it highly available database because this feature is inherent to CouchDB as opposed to RDBMS which does not offer this feature so conveniently [4].



Key ▲	Value
"_design/phd_course" ID: _design/phd_course	{rev: "1-c21a6e878dce4fbb8548161837ff1bf7"}
"ae288eecf814e1e177e9426387000eb7" ID: ae288eecf814e1e177e9426387000eb7	{rev: "1-3b376efbfde851a984622aaa521a8bbc"}
"ae288eecf814e1e177e9426387001689" ID: ae288eecf814e1e177e9426387001689	{rev: "1-8dc6c6f6118a12694690449 5f3d3dfdf"}
"ae288eecf814e1e177e942638700253d" ID: ae288eecf814e1e177e942638700253d	{rev: "1-1822fc42fc7154e8f011a16c5b091874"}
"ae288eecf814e1e177e94263870029c2" ID: ae288eecf814e1e177e94263870029c2	{rev: "1-058a4a173d9f999d20a8df8e8fbbc1e6"}
"ae288eecf814e1e177e9426387002f6d" ID: ae288eecf814e1e177e9426387002f6d	{rev: "2-085e1a82c21e903f0d88f73b55e203db"}
"ae288eecf814e1e177e94263870031e7" ID: ae288eecf814e1e177e94263870031e7	{rev: "5-c5569860 1aafc381ae89c00b8e723a8e"}

Fig. 2 Preview of documents of "university" database

3.5 Scaling and Replication

Replicating databases in CouchDB allows easy replication of databases. The databases can be kept in synchronization very easily by the following HTTP request for database replication:

```
POST/replicateHTTP/1.1{"source":"old_database","tar-
get":"http://www.new/database.com/db"}
```

Replication can be performed from the Futon even more easily. Scaling out means splitting the databases across multiple servers of a cluster. CouchDB does not support scaling inherently; instead, CouchDB Lounge9 application serves the purpose of scaling [4].

3.6 Attachments

Like e-mails support attachments, CouchDB also supports binary attachments to documents. Any kind of file can be attached to a document [4]. The example below shows the command to attach an image file to a document:

```
http://www.local-
host:5984/cars/6e4485ed6c37255e54dg12357f18c8af/car_im-
age.jpg
```

3.7 Querying

RDBMS normally use fixed schema means static data and dynamic queries [7]. On the other hand, CouchDB does the opposite. It uses the dynamic schema in the form of JSON documents. Views are used to query data in CouchDB. Views created are of two kinds: temporary and permanent Views. The outcome of MapReduce operations is demonstrated using Views [5]. Map operation carries out parallelism in CouchDB which is its inherent feature as opposed to RDBMS which carries out operations on data in serial order [7]. The user writes statements for Map operations. Each document is traversed by Map function parallel such that the documents with matching criteria only are emitted using emit () function in the form of the key-value pair [4]. Section 3 explains this concept in more detail.

3.8 Indexing

Indexing is required for faster retrieval and better performance of a database. CouchDB inherently provides auto index updating feature. Map functions traverse through all the documents when running for the first time on a data set [7]. The results are then used by CouchDB to make a B-Tree. This is the most suitable and performance oriented data structure for operations like retrieval, insertion, and deletion. On each modification, deletion of a document; CouchDB updates index B-tree automatically [4].

4 CouchDB Views

Views are an old concept which is being used in traditional RDBMSs for long times. There a view behaves like a window based on the original SQL table. It retrieves rows and columns on the basis of criteria and shows the results out of a real table. In CouchDB view are used to query the database and demonstrate the results of MapReduce functions. These functions offer enormous flexibility, being capable of fine tuning to alterations in the structure of document and indexes being automatically and parallel computed. Each document is traversed and passed as an argument to the Map functions. Documents with matching criteria are emitted as key/value pairs. Because of this flexibility, Views in CouchDB are created parallel and incrementally. Some functions carried out by Views are:

- Filter the documents in the database
- Retrieve data from the documents and show it in a particular order
- Build indexes to find documents by any structure that resides in documents
- Carry out all sorts of calculations on the data in the documents.

CouchDB uses two kinds of views for querying the database i.e. temporary view or adhoc view and permanent view or static view.

4.1 Temporary Views

Temporary views are also called adhoc views because their MapReduce functions are changed from Futon in the code section. In “view” drop down box select “Temporary View” to see this view. Emit () function in map function code is used to emit key-value pair. Emit function always takes two arguments i.e. key and value. A default temporary view emits null in key value and document structure in value field. Value field of default view show details of all items of document including _id and _rev field. Figure 3 shows the values of default temporary view of our example database, i.e. “university” database. The value field shows the attributes of all documents. It is clear that various documents have different schema, i.e. different key-value attribute pair. For example, Fig. 3 shows one document having field (name, doj, course, branch, age, marks) and another one having fields (name, doj, course, age, branch, topic, supervisor). These two documents have different schema and have been easily incorporated in same database without need of any kind of joins. This is how CouchDB is different from RDBMS as schema of documents may change on need.

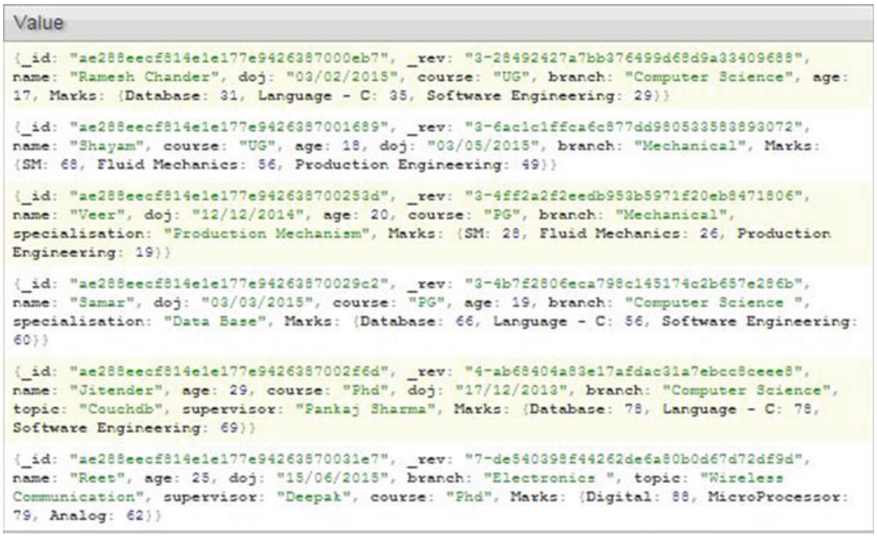
We can change the code of map function to get results according to user query. Suppose from our example database, we want the names of students having age less than 22, and then code of map function would look like the one shown below.

```
function(doc) {
     if(doc.age&&doc.age <  22 && doc.name)
           emit(doc.name , doc.age);  }
```

After we run this code, the result would look like as shown in Fig. 4.

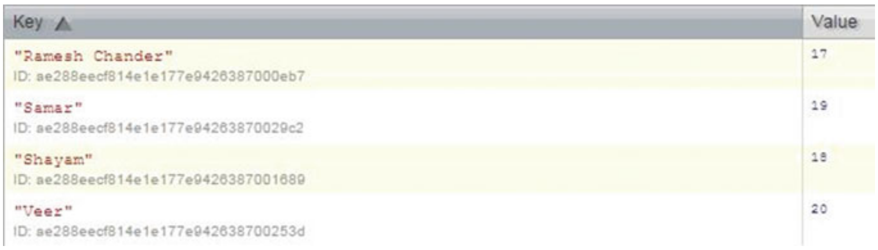
Permanent Views

Temporary views are not appropriate for use in production, as they are actually slow for any database with more than a few dozen documents. They can be used to experiment with view functions, but we must switch to a permanent view if it has to be used in an application. Permanent views are also called static views. Permanent views are created with the help of special type of documents called *design* document. Design documents specify these views in the map function. A single design document can have multiple views. We have created a json file named "design2.


Value

```
{_id: "ae288eecf814e1e177e9426387000eb7", _rev: "3-28492427a7bb376499d68d9a33409688",
name: "Ramesh Chander", doj: "03/02/2015", course: "UG", branch: "Computer Science", age:
17, Marks: {Database: 31, Language - C: 35, Software Engineering: 29}}

{_id: "ae288eecf814e1e177e9426387001689", _rev: "3-6ac1c1ffca6c877dd98053358389307 2",
name: "Shayam", course: "UG", age: 18, doj: "03/05/2015", branch: "Mechanical", Marks:
{SM: 68, Fluid Mechanics: 56, Production Engineering: 49}}

{_id: "ae288eecf814e1e177e942638700253d", _rev: "3-4ff2a2f2eedb953b5971f20eb8471806",
name: "Veer", doj: "12/12/2014", age: 20, course: "PG", branch: "Mechanical",
specialisation: "Production Mechanism", Marks: {SM: 28, Fluid Mechanics: 26, Production
Engineering: 19}}

{_id: "ae288eecf814e1e177e94263870029c2", _rev: "3-4b7f2806eca798c145174c2b657e286b",
name: "Samar", doj: "03/03/2015", course: "PG", age: 19, branch: "Computer Science ",
specialisation: "Data Base", Marks: {Database: 66, Language - C: 56, Software Engineering:
60}}

{_id: "ae288eecf814e1e177e9426387002f6d", _rev: "4-ab68404a83e17afdac31a7ebcc8ceee8",
name: "Jitender", age: 29, course: "Phd", doj: "17/12/2013", branch: "Computer Science",
topic: "Couchdb", supervisor: "Pankaj Sharma", Marks: {Database: 78, Language - C: 78,
Software Engineering: 69}}

{_id: "ae288eecf814e1e177e94263870031e7", _rev: "7-de540398f44262de6a80b0d67d72df9d",
name: "Reet", age: 25, doj: "15/06/2015", branch: "Electronics ", topic: "Wireless
Communication", supervisor: "Deepak", course: "Phd", Marks: {Digital: 88, MicroProcessor:
79, Analog: 62}}
```

Fig. 3 Default temporary view of "university" database



Key ▲	Value
"Ramesh Chander" ID: ae288eecf814e1e177e9426387000eb7	17
"Samar" ID: ae288eecf814e1e177e94263870029c2	19
"Shayam" ID: ae288eecf814e1e177e9426387001689	18
"Veer" ID: ae288eecf814e1e177e942638700253d	20

Fig. 4 Temporary view of name-age pair of “university” database

json” which contains id of the design document “phd_course” and the code of map functions of two view namely “supervisor_detail” and “marks_detail”. The file is shown below in Fig. 5.

The below curl command shows how we created our design document “phd_course” with the help of our file “design2.json”. `curl-X PUT http://127.0.0.1:5984/university/_design/phd_course-d@desktop/design2.json`

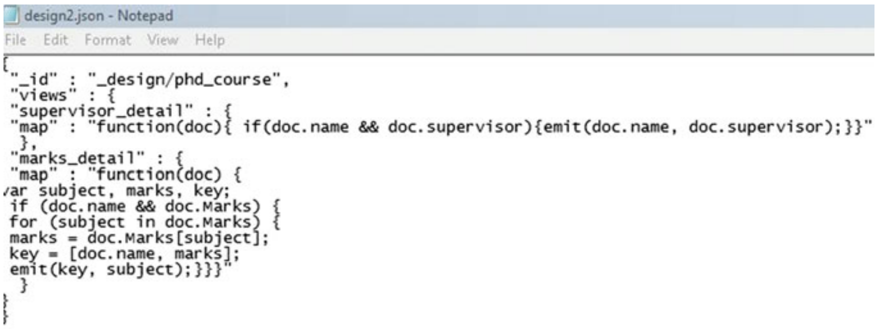
Curl replies with the following JSON document means our design document “phd_course” has been successfully created.

`{"ok":true, "id":"_design/phd_course", "rev":"1-ddde17668d269f5cfeb2aa3259e26b1"}`

Futon shows our design document containing two views as shown in Fig. 6.

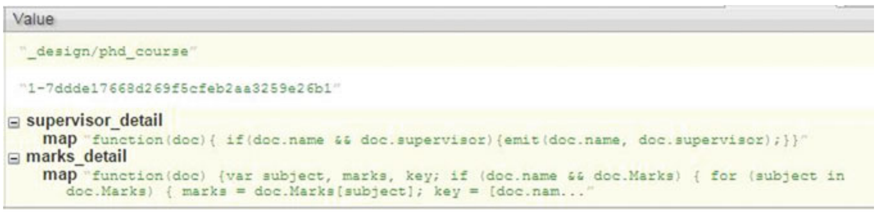
Now to see the result of our view, we select view name “marks_detail” from views drop down box. This view shows the marks of every student in each subject individually and rows sorted by the name of the students as shown in Fig. 7.

CouchDB has given excellent powers to the user in the form of MapReduce functions to design views according to user queries. CouchDB, definitely has some unique features like this incremental MapReduce and indexing which makes it a better fit than traditional RDBMSs for dealing with large datasets.



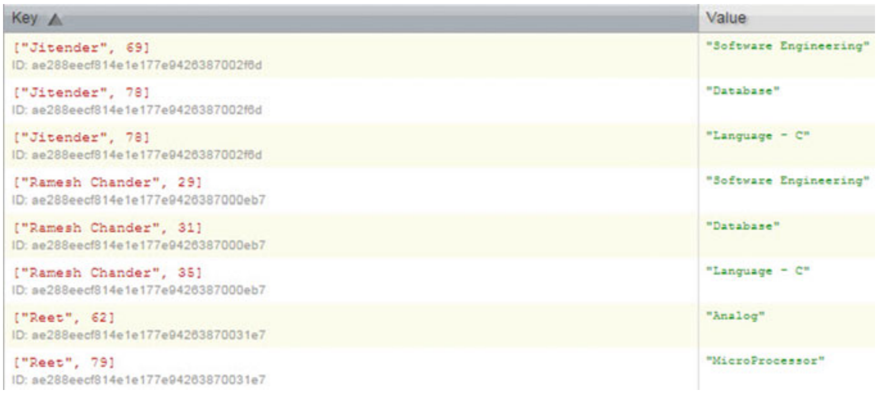
```
{
 "_id" : "_design/phd_course",
 "views" : {
 "supervisor_detail" : {
 "map" : "function(doc){ if(doc.name && doc.supervisor){emit(doc.name, doc.supervisor);}}"
  },
 "marks_detail" : {
 "map" : "function(doc) {
var subject, marks, key;
 if (doc.name && doc.Marks) {
 for (subject in doc.Marks) {
 marks = doc.Marks[subject];
 key = [doc.name, marks];
 emit(key, subject);}}}"
  }
}
}
```

Fig. 5 Code of design document “phd_course” in “design2.json”



```
Value
"_design/phd_course"
"1-7ddde17668d269f5cfeb2aa3259e26b1"
⊟ supervisor_detail
    map "function(doc){ if(doc.name && doc.supervisor){emit(doc.name, doc.supervisor);}}"
⊟ marks_detail
    map "function(doc) {var subject, marks, key; if (doc.name && doc.Marks) { for (subject in
       doc.Marks) { marks = doc.Marks[subject]; key = [doc.nam..."
```

Fig. 6 Design document “phd_course”



Key ▲	Value
["Jitender", 69] ID: ae288eecf814e1e177e9426387002f6d	"Software Engineering"
["Jitender", 78] ID: ae288eecf814e1e177e9426387002f6d	"Database"
["Jitender", 78] ID: ae288eecf814e1e177e9426387002f6d	"Language - C"
["Ramesh Chander", 29] ID: ae288eecf814e1e177e9426387000eb7	"Software Engineering"
["Ramesh Chander", 31] ID: ae288eecf814e1e177e9426387000eb7	"Database"
["Ramesh Chander", 35] ID: ae288eecf814e1e177e9426387000eb7	"Language - C"
["Reet", 62] ID: ae288eecf814e1e177e94263870031e7	"Analog"
["Reet", 79] ID: ae288eecf814e1e177e94263870031e7	"MicroProcessor"

Fig. 7 Result of permanentview “marks_detail”

5 Conclusion and Future Scope

CouchDB features such as dynamic schema, versioning system, and MapReduce view queries have been explained and compared in-line with traditional RDBMS in this paper. Command line utility Curl and web based utility Futon of CouchDB have been used to implement MapReduce operations of permanent views. Focus on present work is majorly on describing the architecture of CouchDB, key features and implementation of views using futon/curl and inline architectural comparison with RDBMS. The future scope includes using the parallelism of MapReduce on multiple servers and on large and diverse datasets with comparison statistics, graphs, and reports with RDBMS.

References

1. Chen, M., Mao, S., Liu, Y.: Big data: a survey. Springer, Mobile NetwApplFeb (2014)
2. Chaiken, R., Jenkins, B., Larson, P.: Scope: Easy and efficient parallel processing of massive data sets. VLDB, 24–30 August, Auckland, New Zealand (2008)

3. Kumari, S., Gupta, P.: Document store NoSQL Databases. Int. J. Artif. Intell. Knowl. Discov. **5**(3) (2015)
4. Henricsson, R., Gustafsson, G.: A comparison of performance in MongoDB and CouchDB using a Python interface. Technical report, Blekinge Institute of Technology (2011)
5. Anderson, J.C., Lehnardt, J., Slater, N.: CouchDB: the definitive guide. O'Reilly Media Inc (2010)
6. Concept of Views in SQL. http://www.tutorialspoint.com/sql/sql-using-views
7. Khanam, Z., Agarwal, S.: Map-reduce implementations: survey and performance comparison. Int. J. Comput. Sci. Inf. Technol. (IJCSIT) **7**(4) (2015)
8. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2004)
9. Gates, A., Natkovich, O., Chopra, S.: Building a high-level dataflow system on top of map-reduce: the pig experience. VLDB, 24–28 August, Lyon, France (2009)
10. Wei, Z., Pierre, G., Chi, C.: Scalable Join Queries in Cloud Data Stores. In: 12th IEEE/ACM IS On CCG Computing (2012)
11. Sharma, V., Dave, M.: SQL and NoSQL databases. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **2**(8), 2027 (2012)
12. Kanwar, R., Trivedi, P. Singh, K.: NoSQL, a Solution for distributed database management system. Int. J. Comput. Appl. (0975–8887) **67**(2), 6–9 (2013)
13. Dharmasiri, H., Goonetillake, M.: A federated approach on heterogeneous NoSQL data stores. In: International Conference on Advances in ICT for Emerging Region, 234–239 (2013)
14. Changlin, H.: Survey on NoSQL database technology. JASEI **2**(2) (2015)
15. Apache: Apache CouchDB documentation and CouchDB latest Version. https://couchdb.apache.org/
16. Why NoSQL, Couchbase white paper. http://www.couchbase.com