**What is RethinkDB**

* RethinkDB database for building real-time web applications
* Open source database for building realtime web applications
* NoSQL database that stores schemaless JSON documents
* Distributed database that is easy to scale
* we can create a cluster with multiple nodes that we can scale up really easily

**Key difference from others DB:**

* Key differentiators of RethinkDB is we can take a query that we’ve written and can subcribe to it to receive change notifications (in a traditional database we have request, response style which mean we have to send a query and then we get a response back)
* No more polling – the database pushes changes to your app
* Reduce the amount of plumbing needed to stream live updates to front-end

**RethinkDB structure:**

* Database -> Table -> Document
* RethinkDB cluster can have multiple databases and each database can have multiple table and within those table we have documents (document is equivalent of what we call row in a sequel database and it is consider as the most atomic element that we’re inserting into our database)

**Sample document:**

{

“name”: “Will Riker”,

“Position”: “Commander”,

“height”: 193,

“birthdate”: “Mon Aug 19 2335”

“ships”: [

{“name”: “USS pegasus”},

{“name”: “USS Potemkin”},

{“name”: “USS Enterprise”},

],

…

}

**Advantages and Disadvantages of this DBSM**

**Advantages:**

* + Highly expressive query language
  + Relational features like table joins
  + Powerful admin UI with point-and-click cluster management(makes it easy to get visibility into our database cluster to see what’s going on, perform Ad-hoc queries - > get results and manage charting and replication without having to rely on our command-line tools or complex configuration systems)

**Disadvantages:**

* + RethinkDB has a few hard limitations, as well as some soft limitations that are dependent on your server configurations.( https://rethinkdb.com/limitations/)

## Cluster/shard limits

## Table/document limits

## Restrictions on keys

## Data types

**Introduction to ReQL:**

* + ReQL embeds natively into our programming language
  + Compose ReQL queries by chaining commands
  + ReQL queries are executed on the server

**ReQL Types:**

* + Structures: Array, Objects
  + Values: String, Numbers, Boonleans, Null
  + Custom: Times, Binary, Geometry

**Sample ReQL Queries:**

r.table(“users”)

.filter(r.row(“age”).gt(30))

r.table(“users”)

.pluck(“last\_name”)

.distinct().count()

r.table(“fellowship”)

.filter({species: “hobbit”})

.update({species: “halfling”})

**How to use this DBSM**

In Window

Start the server from the Windows command prompt:

C:\Users\Slava\RethinkDB\>rethinkdb.exe

Import the driver

First, start Node.js:

$ node

Then. Import the RethinkDB driver:

r = require(‘rethinkdb’);

You can now access RethinkDB commands through the r module.

Open a connection

When you first start RethinkDB, the server opens a port for the client drivers (28015 by default). Let’s open a connection:

**var** connection = **null**;

r.connect( {host: 'localhost', port: 28015}, **function**(err, conn) {

**if** (err) **throw** err;

connection = conn;

})

The variable connection is now initialized and we can run queries.

Create a new table

By default, RethinkDB creates a database test. Let’s create a table authors within this database:

r.db('test').tableCreate('authors').run(connection, **function**(err, result) {

**if** (err) **throw** err;

console.log(JSON.stringify(result, **null**, 2));

})

The result will be:

{

"config\_changes": [

<table configuration data>

],

"tables\_created": 1

}

(The config\_changes field contains metadata about the newly created table; for more details, read about the [tableCreate](https://rethinkdb.com/api/javascript/table_create/) command.) There are a couple of things you should note about this query:

* + - First, we select the database test with the db command.
    - Then, we add the tableCreate command to create the actual table.
    - Lastly, we call run(connection, callback) in order to send the query to the server.

All ReQL queries follow this general structure. Now that we’ve created a table, let’s insert some data!

Insert data

Let’s insert three new documents into the authors table:

r.table('authors').insert([

{ name: "William Adama", tv\_show: "Battlestar Galactica",

posts: [

{title: "Decommissioning speech", content: "The Cylon War is long over..."},

{title: "We are at war", content: "Moments ago, this ship received word..."},

{title: "The new Earth", content: "The discoveries of the past few days..."}

]

},

{ name: "Laura Roslin", tv\_show: "Battlestar Galactica",

posts: [

{title: "The oath of office", content: "I, Laura Roslin, ..."},

{title: "They look like us", content: "The Cylons have the ability..."}

]

},

{ name: "Jean-Luc Picard", tv\_show: "Star Trek TNG",

posts: [

{title: "Civil rights", content: "There are some words I've known since..."}

]

}

]).run(connection, function(err, result) {

if (err) throw err;

console.log(JSON.stringify(result, null, 2));

})

We should get back an object that looks like this:

{

"unchanged": 0,

"skipped": 0,

"replaced": 0,

"inserted": 3,

"generated\_keys": [

"7644aaf2-9928-4231-aa68-4e65e31bf219",

"064058b6-cea9-4117-b92d-c911027a725a",

"543ad9c8-1744-4001-bb5e-450b2565d02c"

],

"errors": 0,

"deleted": 0

}

**RethinkDB Alternatives & Comparisons:**

MongoDB

MongoDB stores data in JSON-like documents that can vary in structure, offering a dynamic, flexible schema. MongoDB was also designed for high availability and scalability, with built-in replication and auto-sharding.

CouchDB

Apache CouchDB is a database that uses JSON for documents, JavaScript for MapReduce indexes, and regular HTTP for its API. CouchDB is a database that completely embraces the web. Store your data with JSON documents. Access your documents and query your indexes with your web browser, via HTTP. Index, combine, and transform your documents with JavaScript.

CockroachDB

It allows you to deploy a database on-prem, in the cloud or even across clouds, all as a single store. It is a simple and straightforward bridge to your future, cloud-based data architecture.

Couchbase

Developed as an alternative to traditionally inflexible SQL databases, the Couchbase NoSQL database is built on an open source foundation and architected to help developers solve real-world problems and meet high scalability demands.

Firebase

Firebase is a cloud service designed to power real-time, collaborative applications. Simply add the Firebase library to your application to gain access to a shared data structure; any changes you make to that data are automatically synchronized with the Firebase cloud and with other clients within milliseconds.