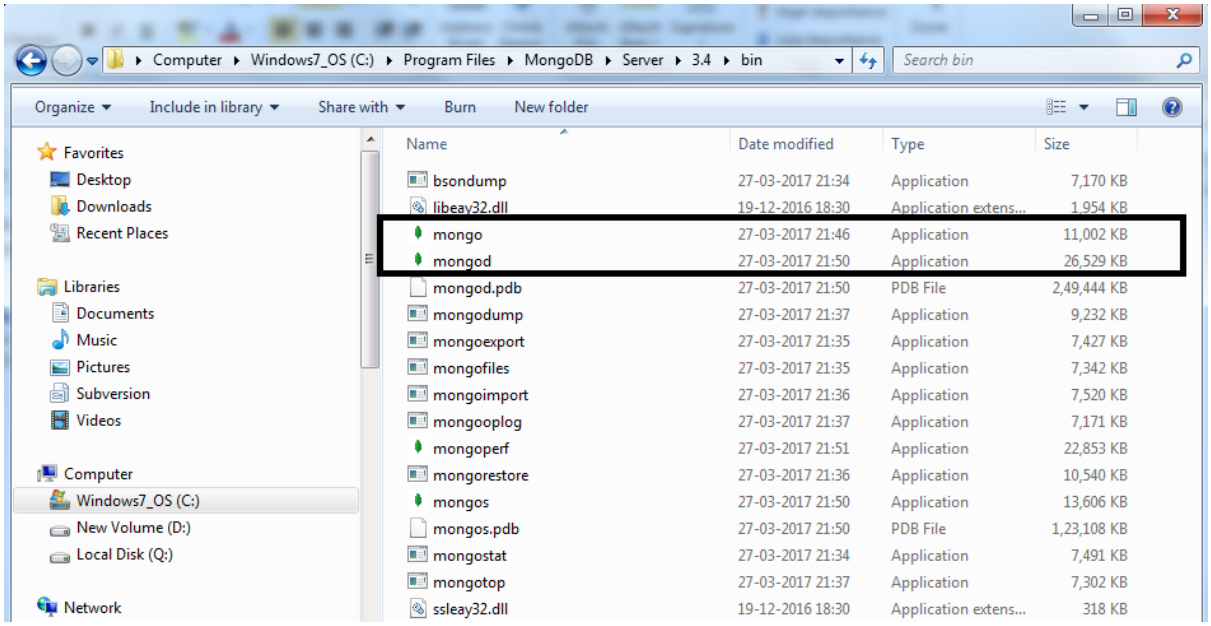
Chapter 1: Introduction to NoSQL Databases

Chapter 2: MongoDB




```
C:\Users\devramk>cd C:\Program Files\MongoDB\Server\3.4\bin

C:\Program Files\MongoDB\Server\3.4\bin>mongod
2017-07-24T17:09:54.750+0530 I CONTROL  [initandlisten] MongoDB starting : pid=12128
ath=C:\data\db\ 64-bit host=PPG-DEVRAM
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windo
R2
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] db version v3.4.3
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] git version: f07437fb5a6cca0
6eb1d6d5e1
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] OpenSSL version: OpenSSL 1.0
p 2016
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] modules: none
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] build environment:
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten]     distmod: 2008plus-ssl
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten]     distarch: x86_64
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten]     target_arch: x86_64
2017-07-24T17:09:54.751+0530 I CONTROL  [initandlisten] options: {}
2017-07-24T17:09:54.770+0530 I -        [initandlisten] Detected data files in C:\da
by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTi
2017-07-24T17:09:54.770+0530 I STORAGE  [initandlisten] wiredtiger_open config: crea
495M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,stat
og=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_id
,checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten]
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten] ** WARNING: Access control i
or the database.
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten] **          Read and write a
nd configuration is unrestricted.
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten]
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten] Hotfix KB2731284 or later up
talled, will zero-out data files.
2017-07-24T17:09:56.462+0530 I CONTROL  [initandlisten]
2017-07-24T17:09:58.860+0530 I FTDC     [initandlisten] Initializing full-time diagn
ure with directory 'C:/data/db/diagnostic.data'
2017-07-24T17:09:58.894+0530 I NETWORK  [thread1] waiting for connections on port 27
```

```
C:\Users\devramk>cd C:\Program Files\MongoDB\Server\3.4\bin

C:\Program Files\MongoDB\Server\3.4\bin>mongo
MongoDB shell version v3.4.3
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.3
Server has startup warnings:
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten]
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten] ** WARNING: Access control
or the database.
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten] **          Read and write
nd configuration is unrestricted.
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten]
2017-07-24T17:09:56.461+0530 I CONTROL  [initandlisten] Hotfix KB2731284 or later
talled, will zero-out data files.
2017-07-24T17:09:56.462+0530 I CONTROL  [initandlisten]
>
```

```
{ "firstName": null }
```

```
{ "isEditable": true }
```

```
{ "city": "london" }
```

```
{
    "birthDate" : new Date(1501086866059)
}
```

```
{
    "_id" : ObjectId("5978c5ca24de39c8f206196b"),
    "birthDate" : ISODate("2017-07-26T16:34:26.059Z")
}
```

```
{
    "cities" : ["London", "Delhi", "New York"]
}
```

```
{
    "_id" : ObjectId("59769440c5160d8a42a02358"),
    "firstName" : "John",
    "lastName" : "Cent",
    "age" : 45,
    "email" : "john.cent@abc.com",
    "address" : [
        {
            "city" : "NY",
            "state" : "NY",
            "country" : "USA"
        }
    ]
}
```


```
> use sample_db;
switched to db sample_db
```

Data is stored in database in form of databases where database is containe

```
> show databases;
local      0.000GB
mongo_lam  0.063GB
test       0.000GB
```

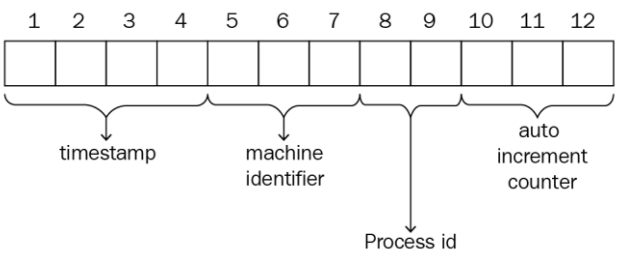
```
> use sample_db;
switched to db sample_db
```

This command create database called as “sample

```
> db.createCollection("users_profile");
{ "ok" : 1 }
```

“user_profiles” should store data related t
users friend list as it should not part of user's pr

```
{
    "_id" : ObjectId("596ce40ceafa9cc9cf097a41"),
    "title" : "MongoDB Overview",
    "description" : "MongoDB is no sql database",
    "by" : "tutorials point",
    "url" : "http://www.tutorialspoint.com",
    "tags" : [
        "MongoDB",
        "database",
        "NoSQL"
    ],
    "likes" : 100.0,
    "comments" : [
        {
            "user" : "user1",
            "message" : "My first comment",
            "dateCreated" : ISODate("2011-02-19T20:45:00.000Z"),
            "like" : 0.0
        },
        {
            "user" : "user2",
            "message" : "My second comments",
            "dateCreated" : ISODate("2011-02-25T02:15:00.000Z"),
            "like" : 5.0
        }
    ]
```




```
db.users_profile.insertOne({
    userId : 1,
    firstName : "John",
    lastName : "Rcihard",
    age : 26,
    email : "john2992@mail.com"
});
```

```
/* 1 */
{
    "acknowledged" : true,
    "insertedId" : ObjectId("596ce72cbc5266c2c9c44e8d")
}
```

```
db.users_profile.insertMany([
    {
        userId : 1, firstName : "John", lastName : "Rcihard",
        age : 26, email : "john2992@mail.com"
    },
    {
        userId : 2, firstName : "Kedar", lastName : "Sans",
        age : 29, email : "kedar.sans@mail.com"
    },
    {
        userId : 3, firstName : "Chan", lastName : "Kuli",
        age : 27, email : "chann.k@mail.com"
    }
]);
```

```
/* 1 */
{
    "acknowledged" : true,
    "insertedIds" : [
        ObjectId("596ce893bc5266c2c9c44e8e"),
        ObjectId("596ce893bc5266c2c9c44e8f"),
        ObjectId("596ce893bc5266c2c9c44e90")
    ]
}
```

```
db.users_profile.find({})
```

```
/* 1 */
{
    "_id" : ObjectId("596ce72cbc5266c2c9c44e8d"),
    "userId" : 1.0,
    "firstName" : "John",
    "lastName" : "Rcihard",
    "age" : 26.0,
    "email" : "john2992@mail.com"
}

/* 2 */
{
    "_id" : ObjectId("596ce893bc5266c2c9c44e8e"),
    "userId" : 1.0,
    "firstName" : "John",
    "lastName" : "Rcihard",
    "age" : 26.0,
    "email" : "john2992@mail.com"
}
```

```
db.users_profile.find({
        firstName : "John"
    });
```

```
/* 1 */
{
    "_id" : ObjectId("596ce72cbc5266c2c9c44e8d"),
    "userId" : 1.0,
    "firstName" : "John",
    "lastName" : "Rcihard",
    "age" : 26.0,
    "email" : "john2992@mail.com"
}

/* 2 */
{
    "_id" : ObjectId("596ce893bc5266c2c9c44e8e"),
    "userId" : 1.0,
    "firstName" : "John",
    "lastName" : "Rcihard",
    "age" : 26.0,
    "email" : "john2992@mail.com"
}
```

```
db.users_profile.find({
    firstName : {
        $in :["John", "Kedar"]
      }
    });
```

```
/* 2 */
{
    "_id" : ObjectId("596ce893bc5266c2c9c44e8e"),
    "userId" : 1.0,
    "firstName" : "John",
    "lastName" : "Rcihard",
    "age" : 26.0,
    "email" : "john2992@mail.com"
}

/* 3 */
{
    "_id" : ObjectId("596ce893bc5266c2c9c44e8f"),
    "userId" : 2.0,
    "firstName" : "Kedar",
    "lastName" : "Sans",
    "age" : 29.0,
    "email" : "kedar.sans@mail.com"
}
```

```
db.users_profile.find({
         firstName : "John",
         age : { $lt : 29 }
 });
```

```
db.users_profile.find({
        $or : [
            { "firstName" : "John" },
            { "age" : { $lt : 30 } }
        ]
    });
```

```
db.users_profile.find({
        "firstName" : "John",
        $or : [
            { "age" : { $lt : 30 } },
            { "lastName" : /^s/ },
        ]
    });
```

```
db.users_profile.updateOne(
        { userId : 1 },
        { $set : {
            age : 30
        }
});
```

```
/* 1 */
{
    "acknowledged" : true,
    "matchedCount" : 1.0,
    "modifiedCount" : 1.0
}
```

```
db.users_profile.updateMany(
        { age : 20 },
        { $set : {age : 35}
});
```

```
/* 1 */
{
    "acknowledged" : true,
    "matchedCount" : 0.0,
    "modifiedCount" : 0.0
}
```

```
db.users_profile.replaceOne(
        { userId : 1 },
        {
          userId : 1,
          firstName : "Sam",
          lastName : "Billings",
          age : 26,
          email : "sam2992@mail.com"
        });
```

```
db.users_profile.deleteMany({});
```

```
db.users_profile.deleteOne({
        userId : 1
});
```

```
{
    "_id" : ObjectId("59769440c5160d8a42a02358"),
    "firstName" : "John",
    "lastName" : "Cent",
    "age" : 45,
    "email" : "john.cent@abc.com",
    "address" : [
        {
            "city" : "NY",
            "state" : "NY",
            "country" : "USA"
        }
    ]
}
```

```
{
    "_id" : ObjectId("59769578c5160d8a42a0235a"),
    "firstName" : "John",
    "lastName" : "Cent",
    "age" : 45,
    "email" : "john.cent@abc.com",
    "address" : [
        ObjectId("597694f5c5160d8a42a02359")
    ]
}
```

```
{
    "_id" : ObjectId("597694f5c5160d8a42a02359"),
    "city" : "NY",
    "state" : "NY",
    "country" : "USA"
}
```

```
db.user_profiles.createIndex({  "firstName" : 1 });
```

```
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1.0
}
```

```
db.reviews.createIndex({"comments" : "text"})
```

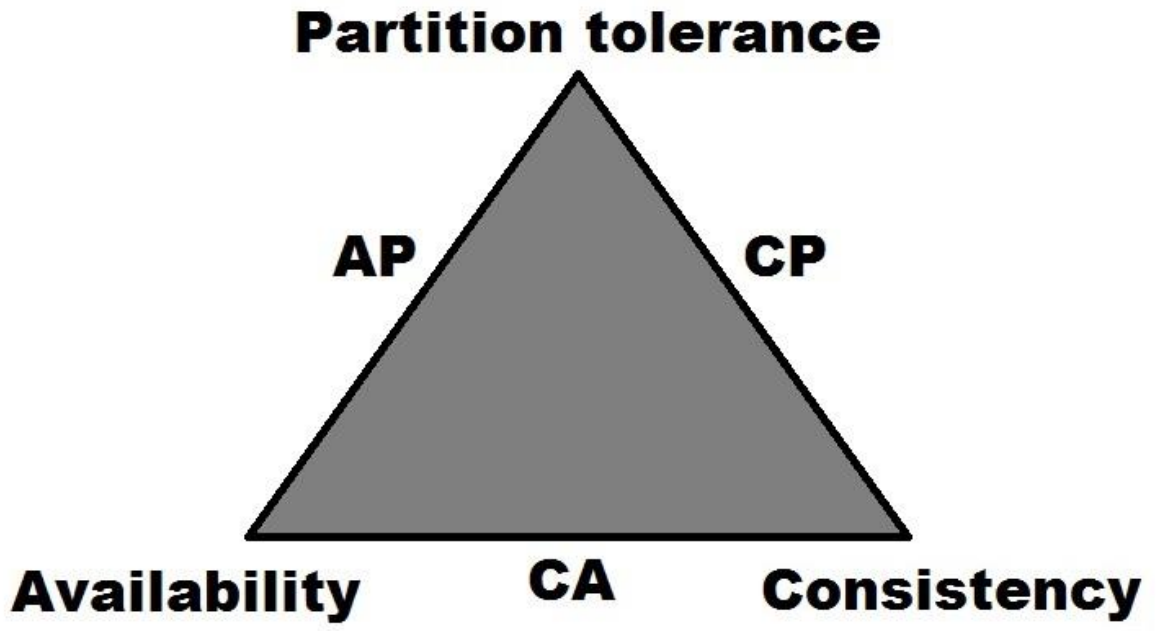
```
db.reviews.createIndex({
        "comments" : "text",
        "subject" : "text"
});
```

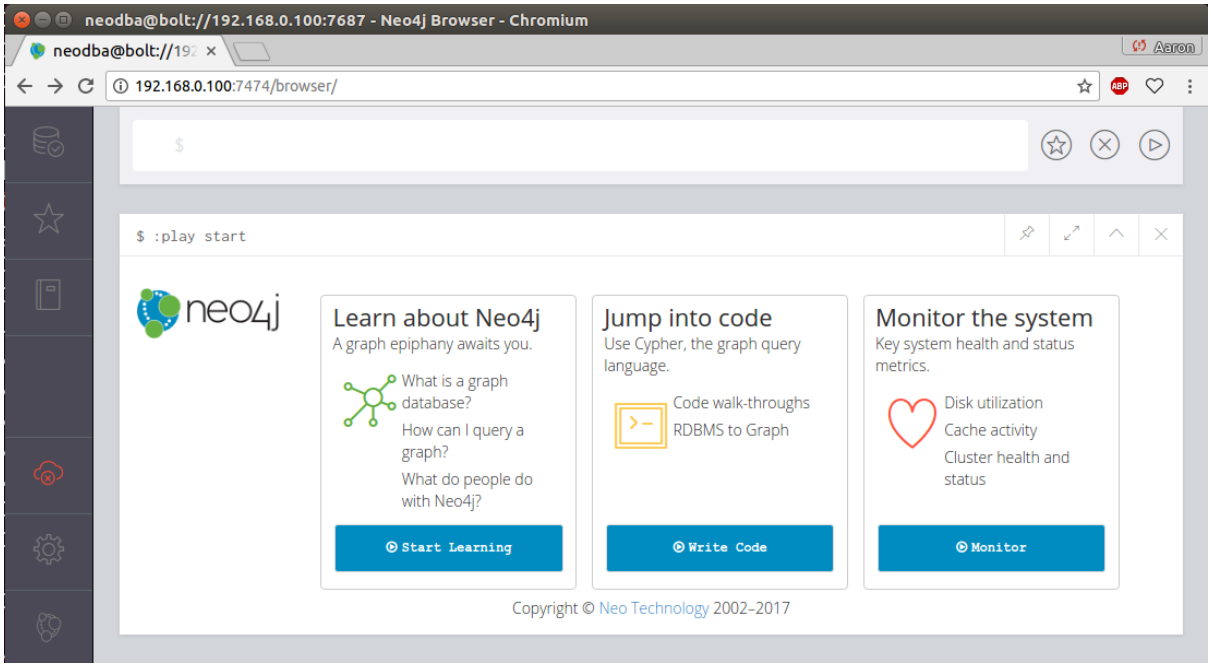
```
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1.0
}
```

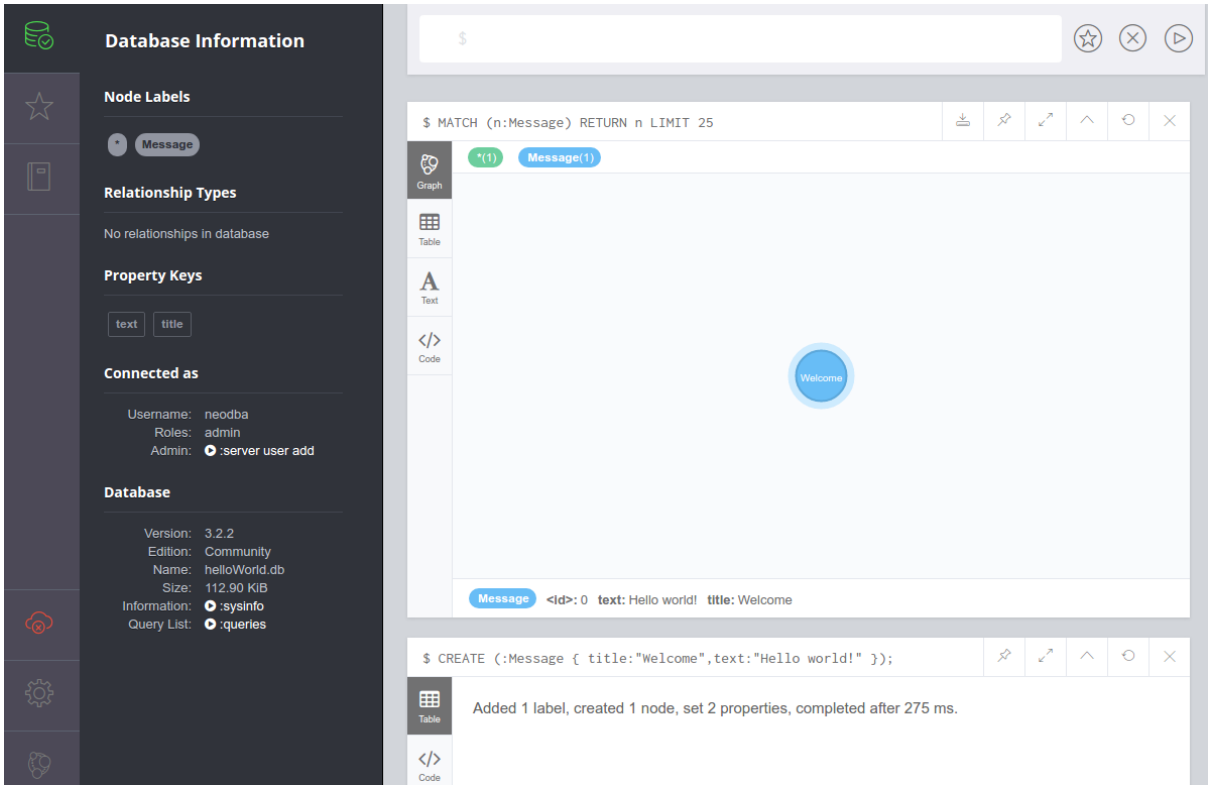
```
db.log.createIndex(
    {
        "createdAt":1
    }, {
       "expiredAfterSeconds" : 300
    })
```

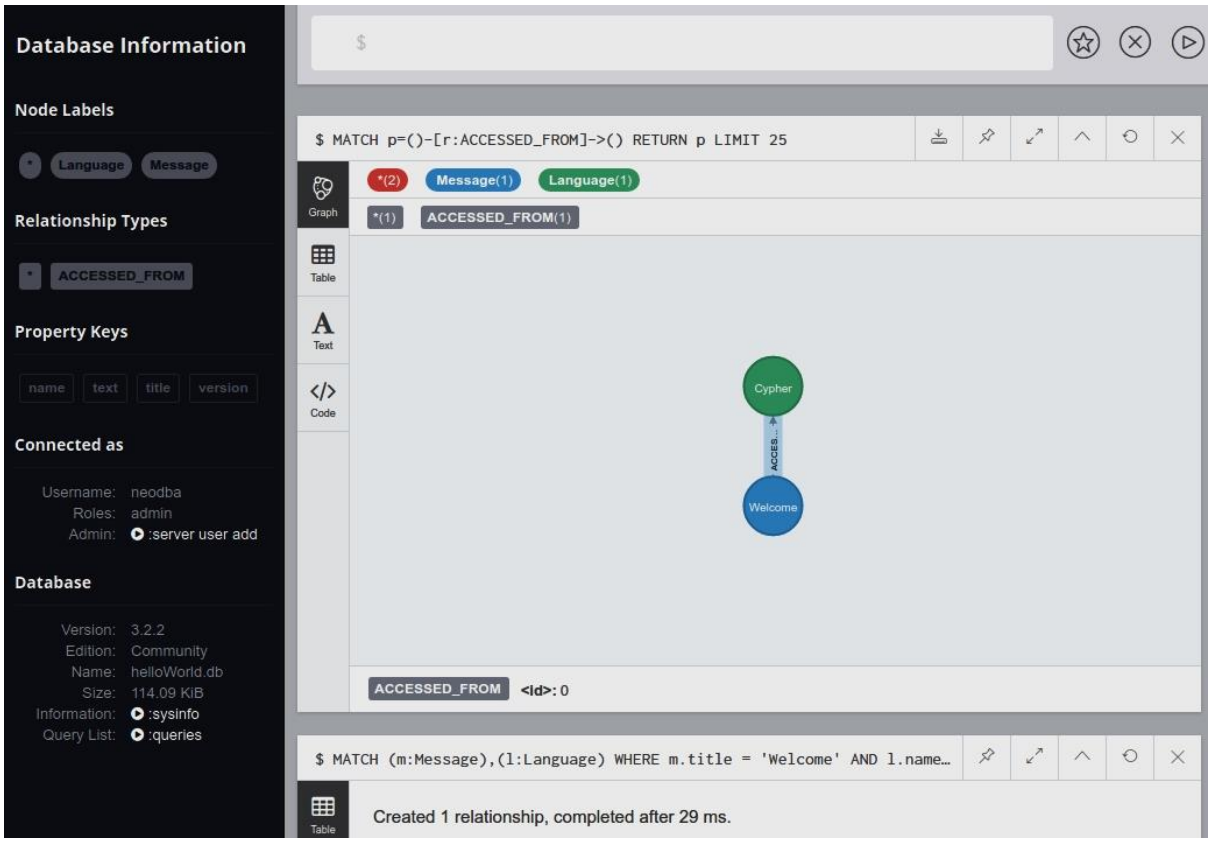
```
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1.0
}
```

Chapter 3: Neo4j

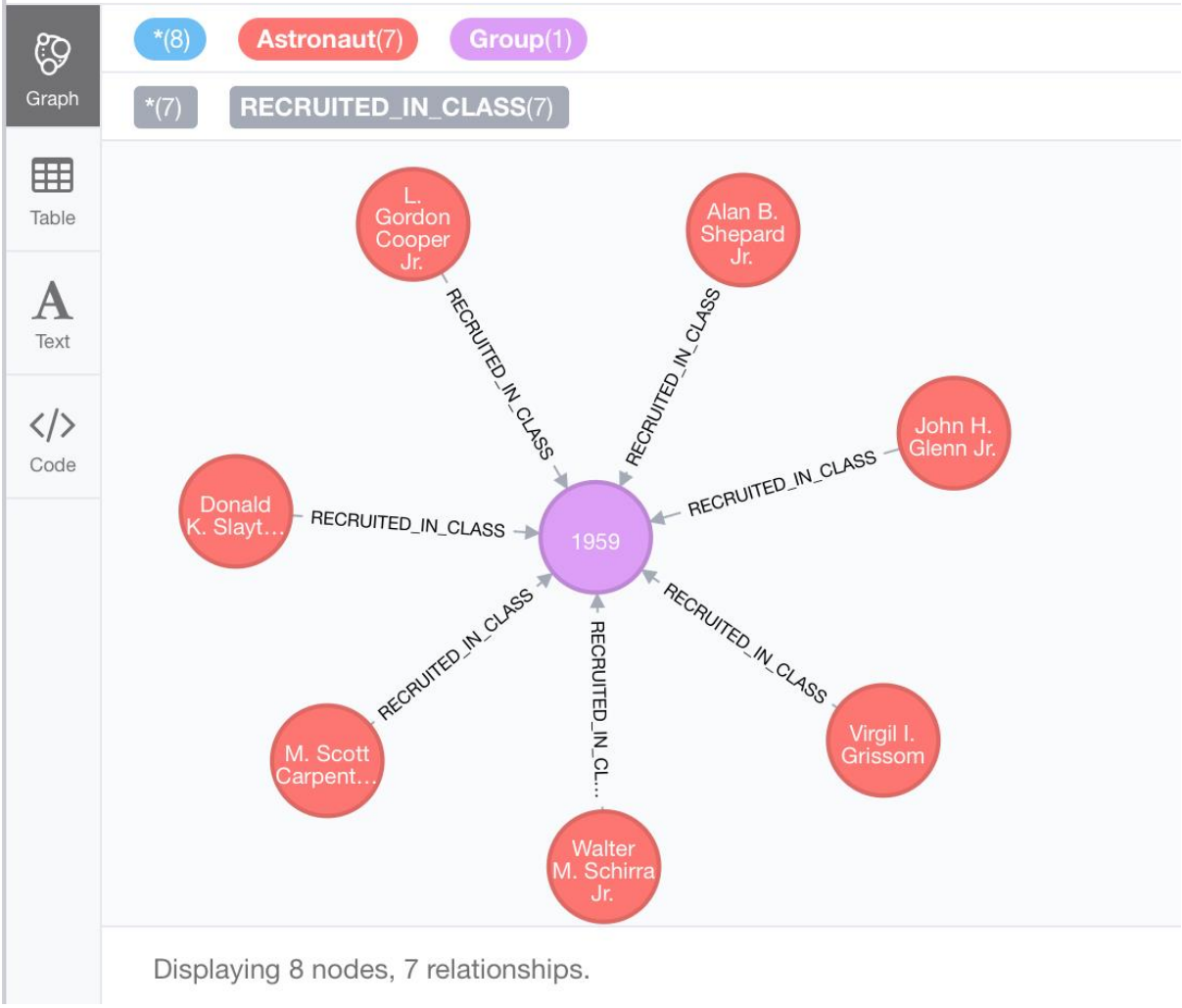




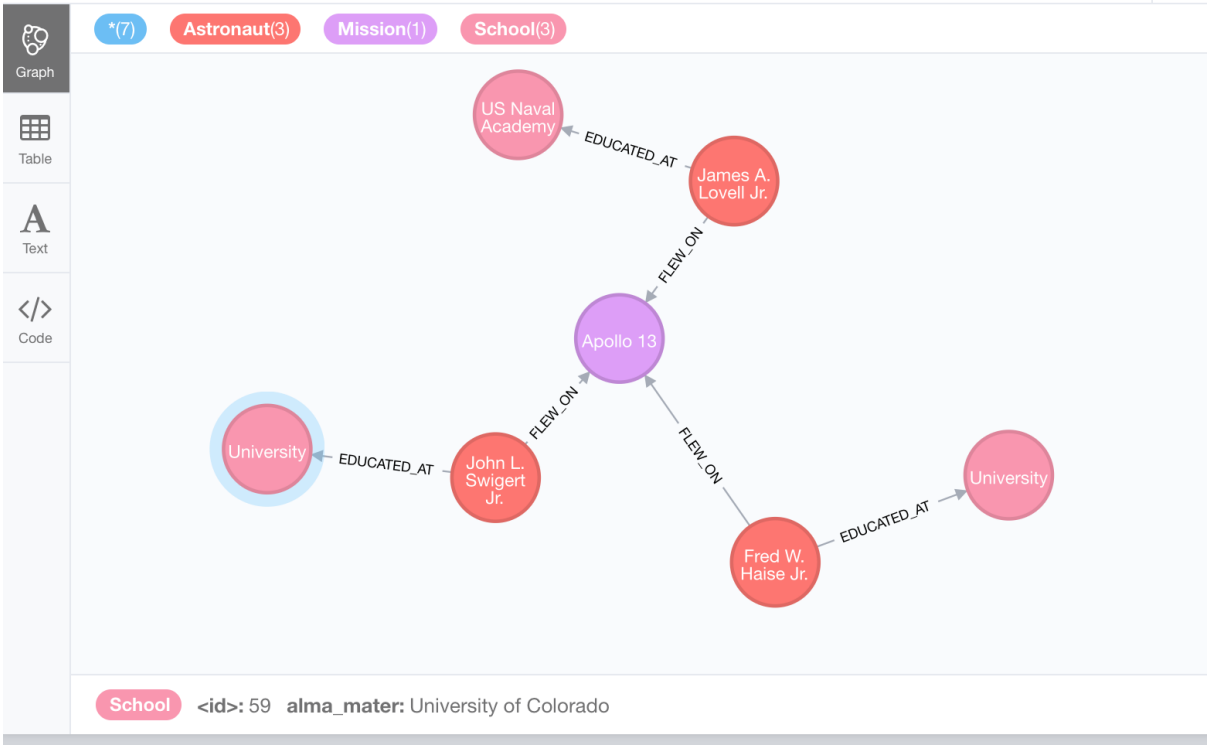

Database Information
Node Labels
Message
Relationship Types
No relationships in database
Property Keys
text
title
Connected as
Username: neodba
Roles: admin
Admin: :server user add
Database
Version: 3.2.2
Edition: Community
Name: helloWorld.db
Size: 112.90 KiB
Information: :sysinfo
Query List: :queries
$ MATCH (n:Message) RETURN n LIMIT 25
*(1)
Message(1)
Graph
Table
Text
Code
Welcome
Message <id>: 0 text: Hello world! title: Welcome
$ CREATE (:Message { title:"Welcome",text:"Hello world!" });
Added 1 label, created 1 node, set 2 properties, completed after 275 ms.


Database Information
Node Labels
Language
Message
Relationship Types
ACCESSED_FROM
Property Keys
name
text
title
version
Connected as
Username: neodba
Roles: admin
Admin: :server user add
Database
Version: 3.2.2
Edition: Community
Name: helloWorld.db
Size: 114.09 KiB
Information: :sysinfo
Query List: :queries
$ MATCH p=()-[r:ACCESSED_FROM]->() RETURN p LIMIT 25
*(2)
Message(1)
Language(1)
*(1)
ACCESSED_FROM(1)
Graph
Table
Text
Code
Cypher
ACCES...
Welcome
ACCESSED_FROM <id>: 0
$ MATCH (m:Message),(l:Language) WHERE m.title = 'Welcome' AND l.name...
Created 1 relationship, completed after 29 ms.

```
$ MATCH p=()-[r:RECRUITED_IN_CLASS]->(g {group:"1"}) RETURN p;
```



```
$ MATCH (s:School)<-[:EDUCATED_AT]- (a:Astronaut)-[:FLEW_ON]->(m:Mission {name:'Apollo 13…
```




```
$ MATCH (m:Mission {name:"ISS-53/54 (Soyuz)"})<-[:FLEW_ON]-(a:Astronaut) R…
```

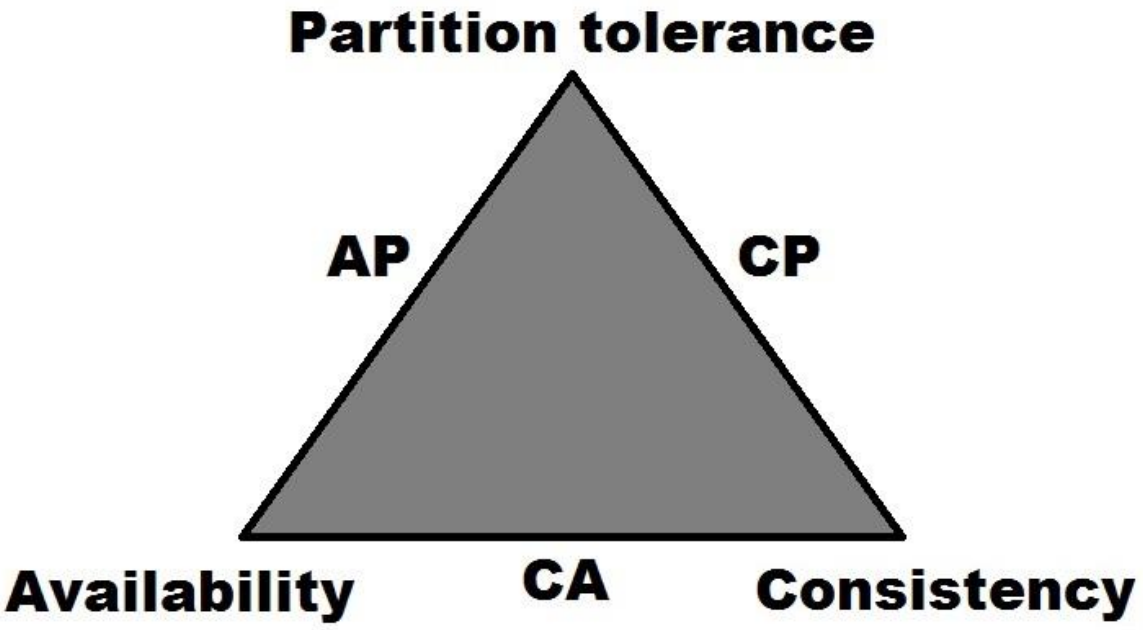

```
$ MATCH (m:Mission {name:"ISS-53/54 (Soyuz)"})RETURN m;
```

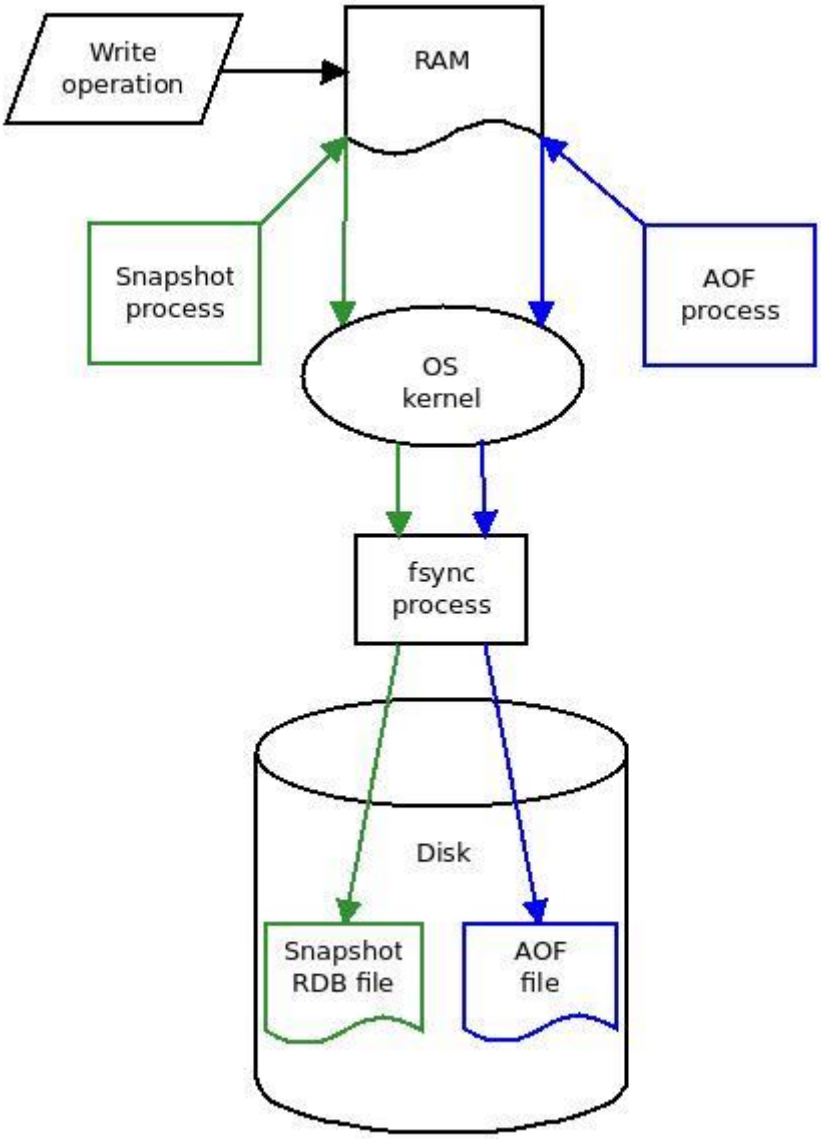
Table

(no changes, no records)

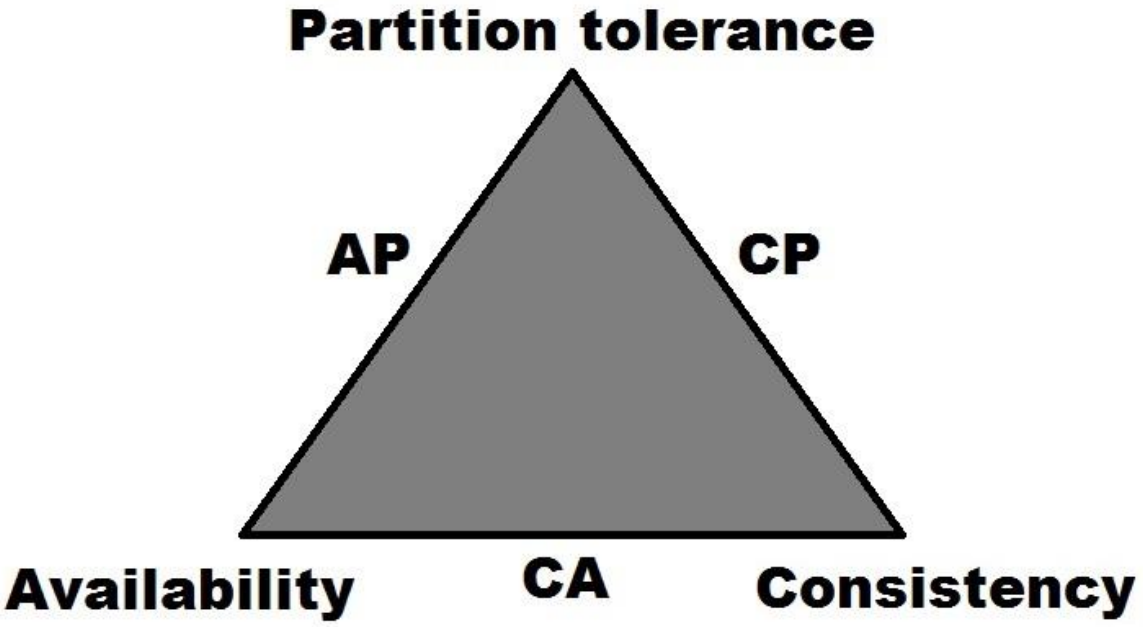
Code

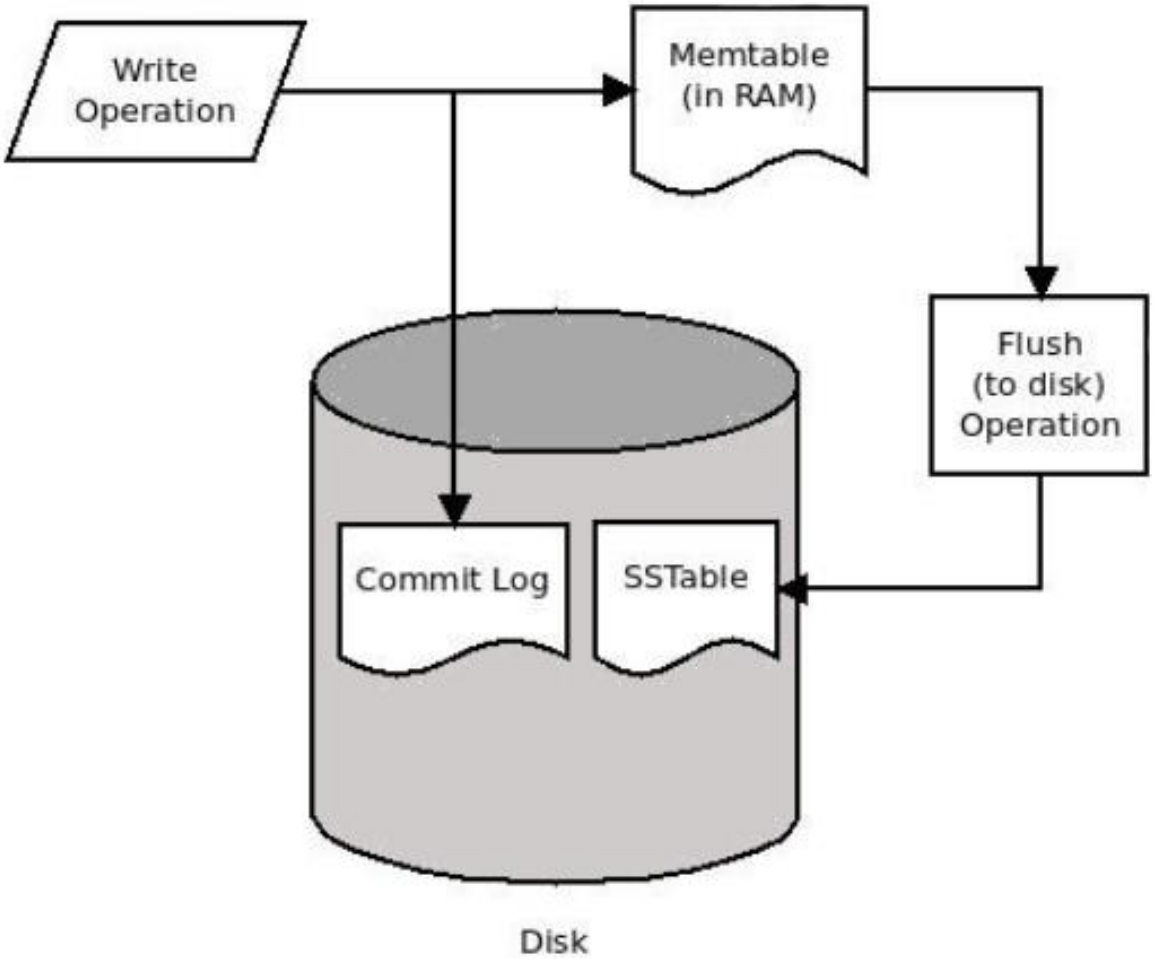
Chapter 4: Redis

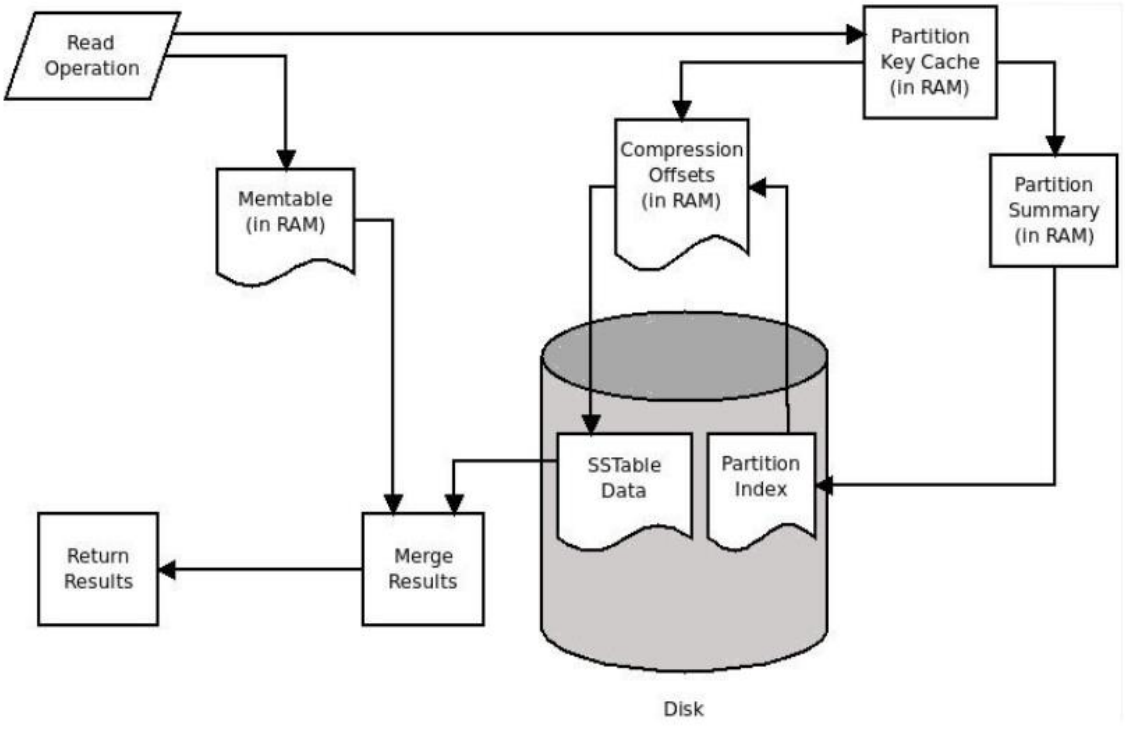


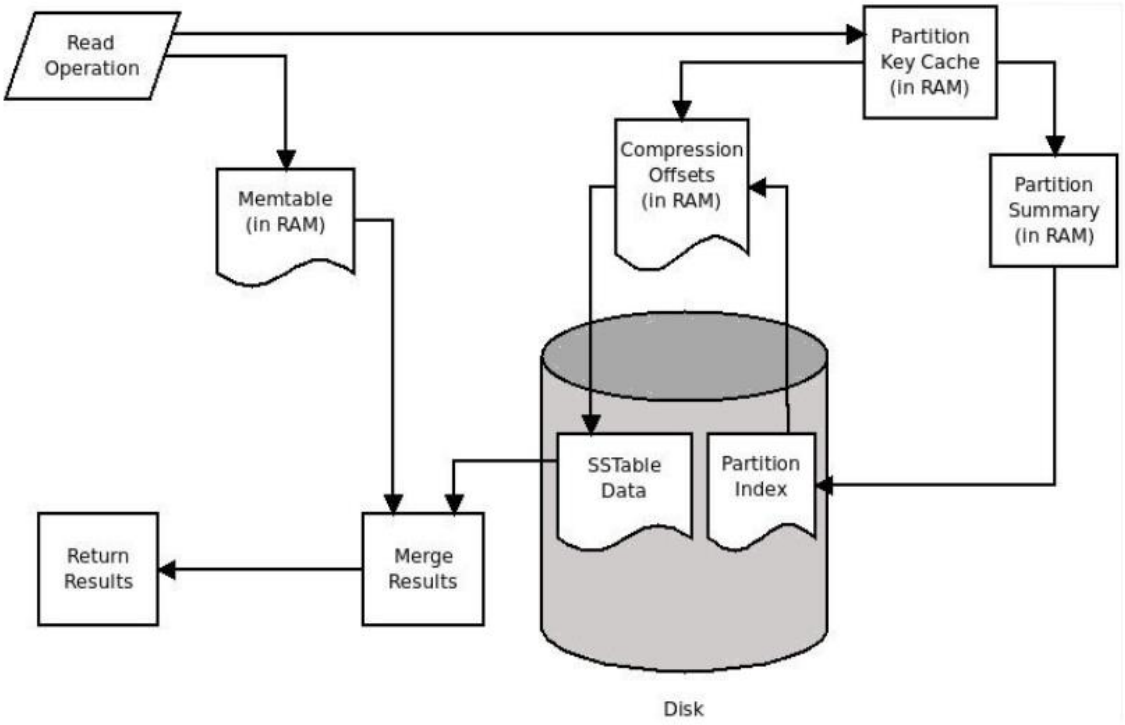

Write
operation
RAM
Snapshot
process
AOF
process
OS
kernel
fsync
process
Disk
Snapshot
RDB file
AOF
file

Chapter 5: Cassandra

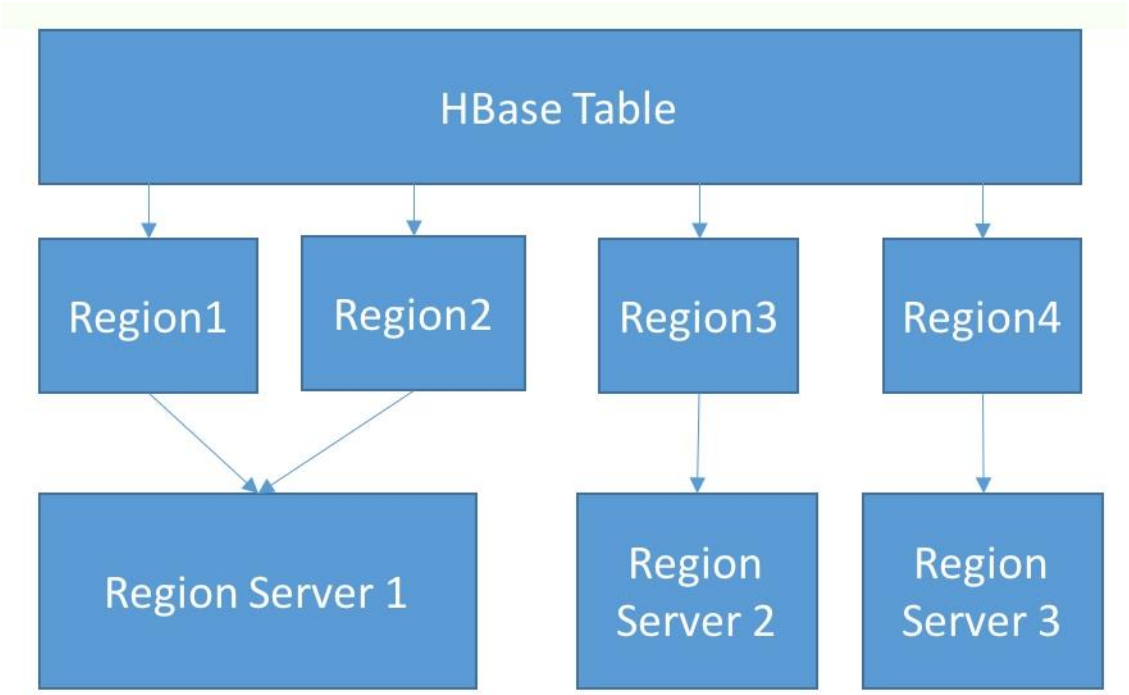


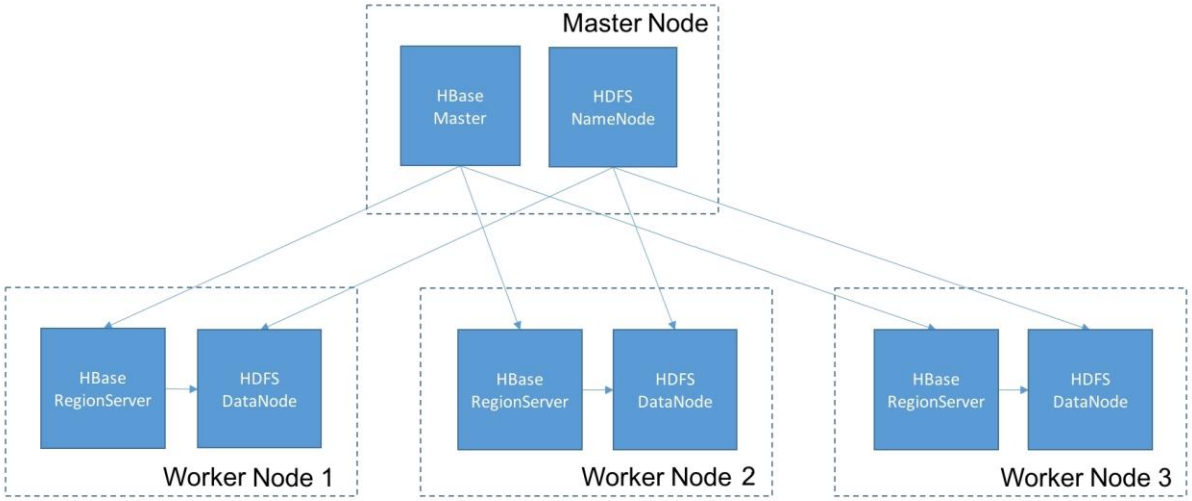

Write Operation
Memtable (in RAM)
Flush (to disk) Operation
Commit Log
SSTable
Disk

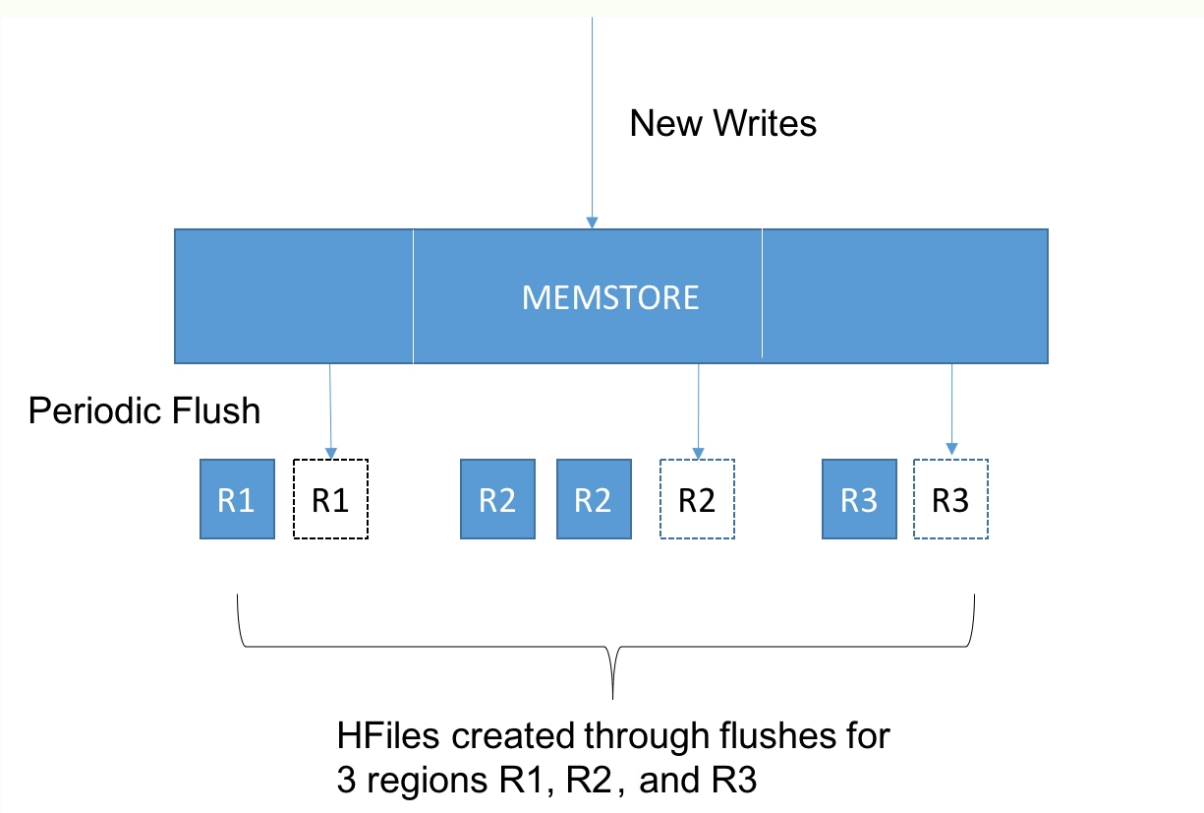

Read Operation
Partition Key Cache (in RAM)
Compression Offsets (in RAM)
Partition Summary (in RAM)
Memtable (in RAM)
SSTable Data
Partition Index
Return Results
Merge Results
Disk


Read Operation
Partition Key Cache (in RAM)
Compression Offsets (in RAM)
Partition Summary (in RAM)
Memtable (in RAM)
SSTable Data
Partition Index
Return Results
Merge Results
Disk

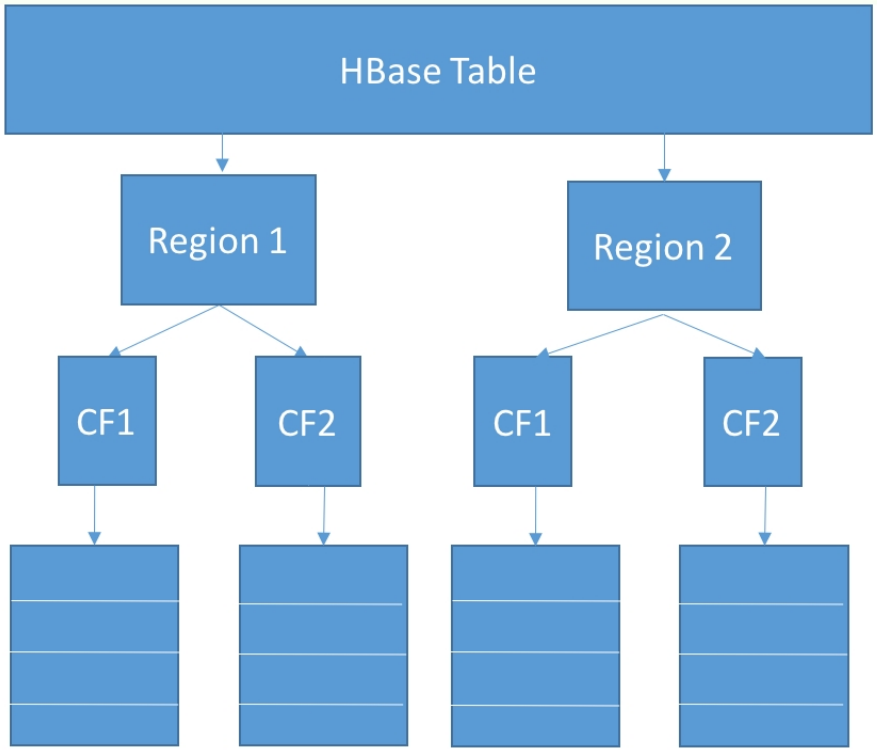
Chapter 6: HBase




Master Node
HBase Master
HDFS NameNode
HBase RegionServer
HDFS DataNode
Worker Node 1
HBase RegionServer
HDFS DataNode
Worker Node 2
HBase RegionServer
HDFS DataNode
Worker Node 3


New Writes
MEMSTORE
Periodic Flush
R1
R1
R2
R2
R2
R3
R3
HFiles created through flushes for
3 regions R1, R2, and R3

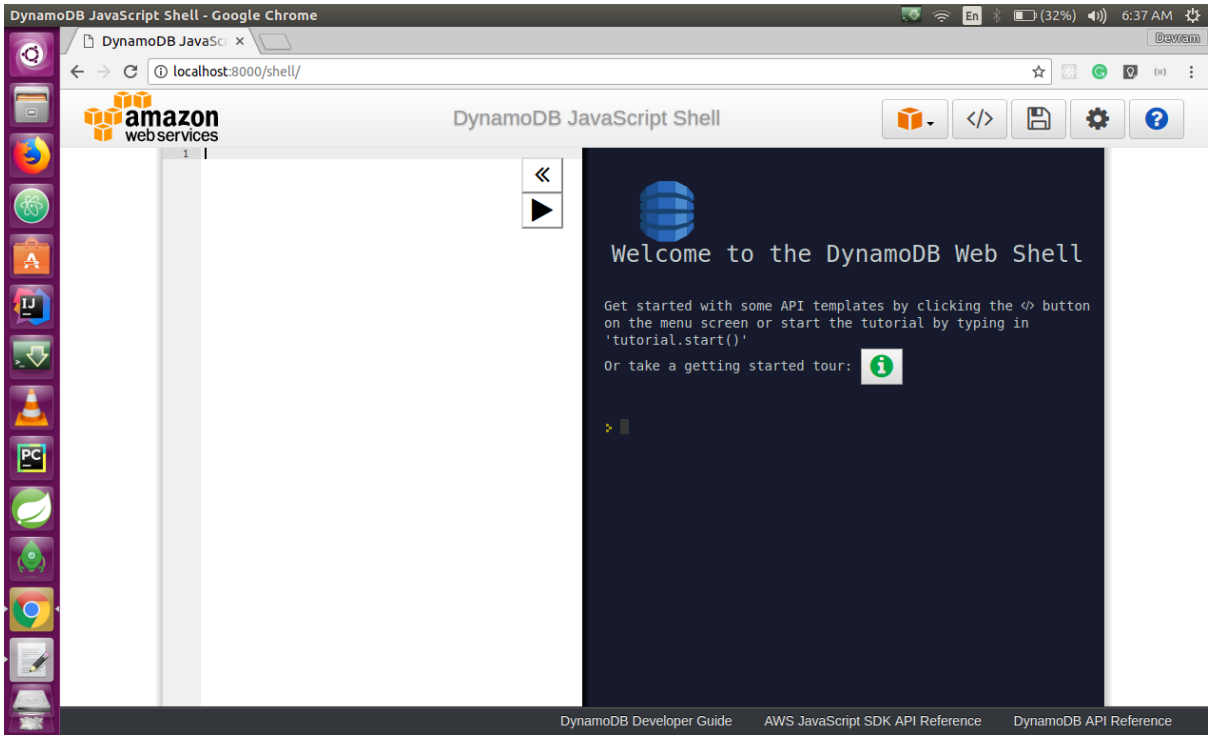
Row Key	Version	Metrics: Humidity	Metrics: Temperature
Key 1	T1	V1	V11
Key 1	T2	V2	V12
Key 2	T1	V3	V13
Key 3	T1	V4	V14
Key 3	T3	V5	V15
Key 3	T4	V6	V16



Chapter 7: DynamoDB

```
devram@devram-Inspiron-3542:~/Desktop/packtpub/d-jar$ java -jar DynamoDBLocal.jar -sharedDb
Initializing DynamoDB Local with the following configuration:
Port:   8000
InMemory:       false
DbPath: null
SharedDb:       true
shouldDelayTransientStatuses:   false
CorsParams:     *
```

```
devram@devram-Inspiron-3542:~$ aws dynamodb list-tables --endpoint-url http://localhost:8000
{
    "TableNames": []
}
```



aws

Sign in

Email address of your AWS account

To sign in as an IAM user, enter your account ID or account alias instead.

Next

New to AWS?

Create a new AWS account

aws English

AWS Accounts Include 12 Months of Free Tier Access

Including use of Amazon EC2, Amazon S3, and Amazon DynamoDB

Visit **aws.amazon.com/free** for full offer terms

Create an AWS account

Email address

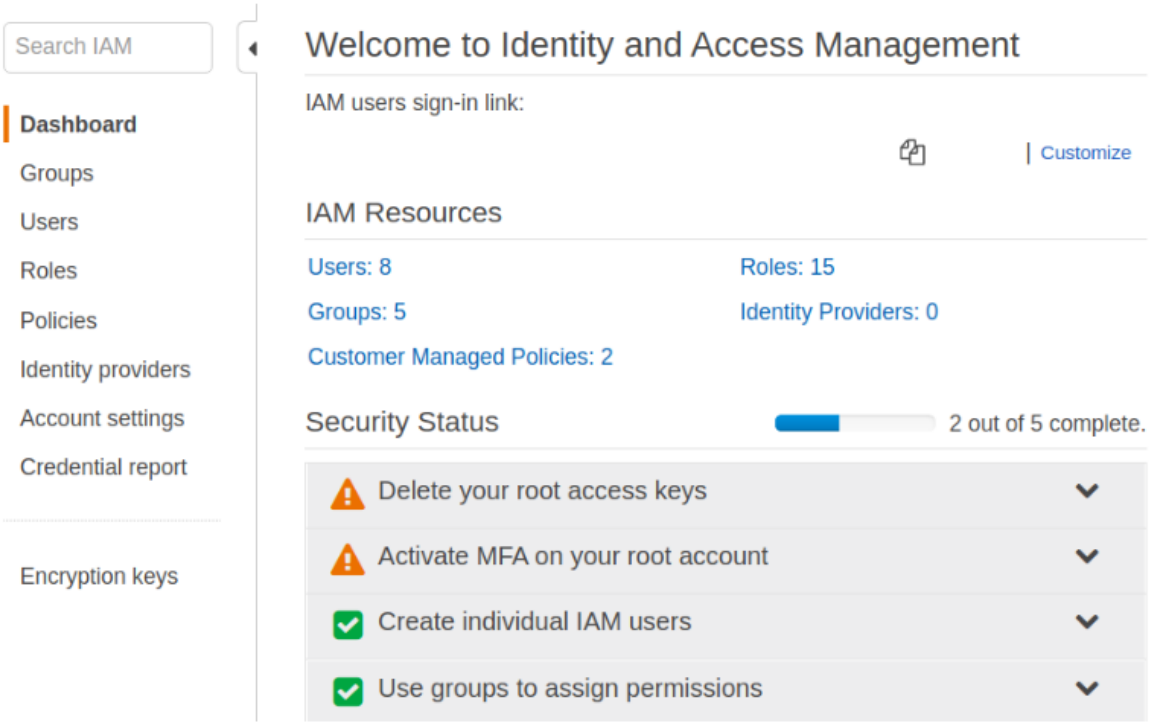
Password

Confirm password

AWS account name

Continue

Sign in to an existing AWS account



Add user

1 2 3 4

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* john

Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. Learn more

Access type* ☐ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

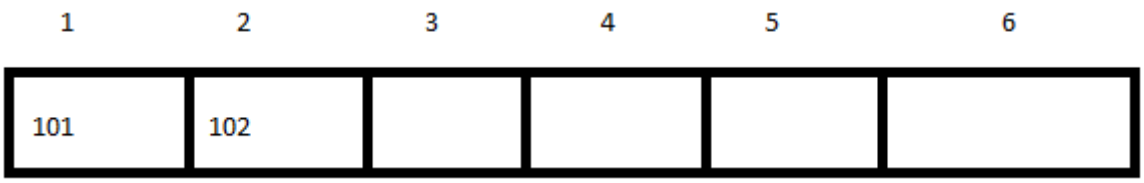
☐ **AWS Management Console access**

Download .csv

		User	Access key ID	Secret access key
▸	✓	john	AKIAJW5T7C2PFBZJYM3A	********* Show

```
{
        PersonID : 101,
        FirstName:"John",
        LastName:"smith",
        Age:26,
        Email:"john.smith@co.in"
},
{
        PersonID : 102,
        FirstName:"Ryan",
        LastName:"Henry",
        Age:27,
        Email:"john.smith@co.in"
},
{
        PersonID : 201,
        FirstName:"Kedar",
        LastName:"Sam",
        Age:23,
        Email:"kedar.sam@co.in"
}
```

```
{
        PersonID : 101,
        FirstName:"John",
        LastName:"smith",
        Age:26,
        Email:"john.smith@co.in"
},
{
        PersonID : 102,
        FirstName:"Ryan",
        LastName:"Henry",
        Age:27,
        Email:"john.smith@co.in"
},
{
        PersonID : 201,
        FirstName:"Kedar",
        LastName:"Sam",
        Age:23,
        Email:"kedar.sam@co.in"
}
```



Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* person

Primary key* Partition key

PersonID String

Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* person

Primary key* Partition key

PersonID String

Add sort key

Age Number

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

Create index

Primary key* Partition key

Email | String

Add sort key

Index name* Email-index

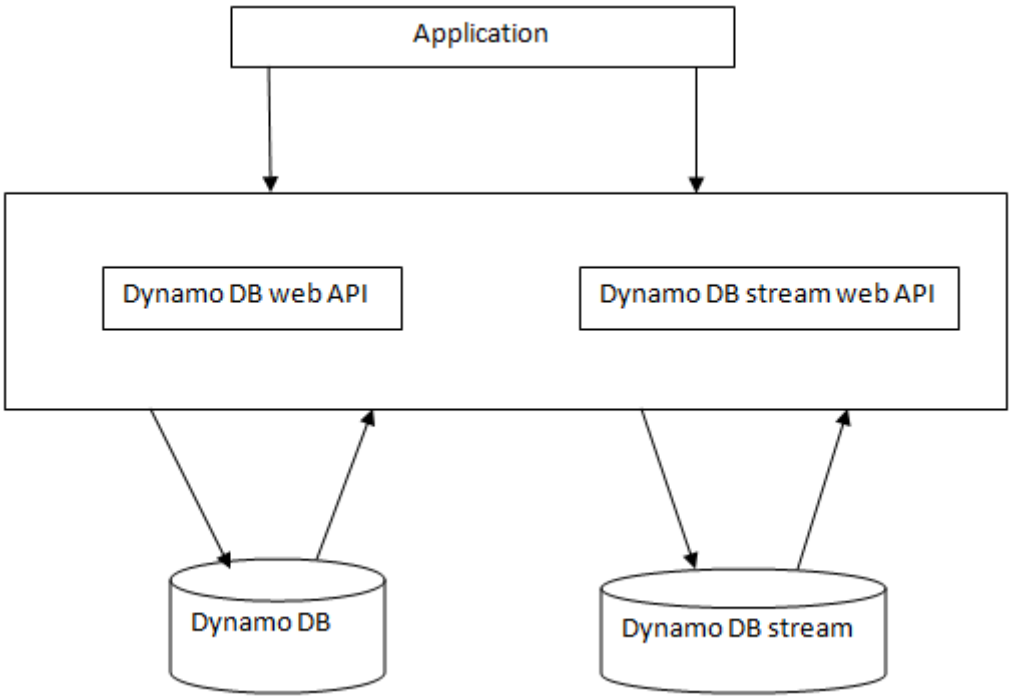
Projected attributes All

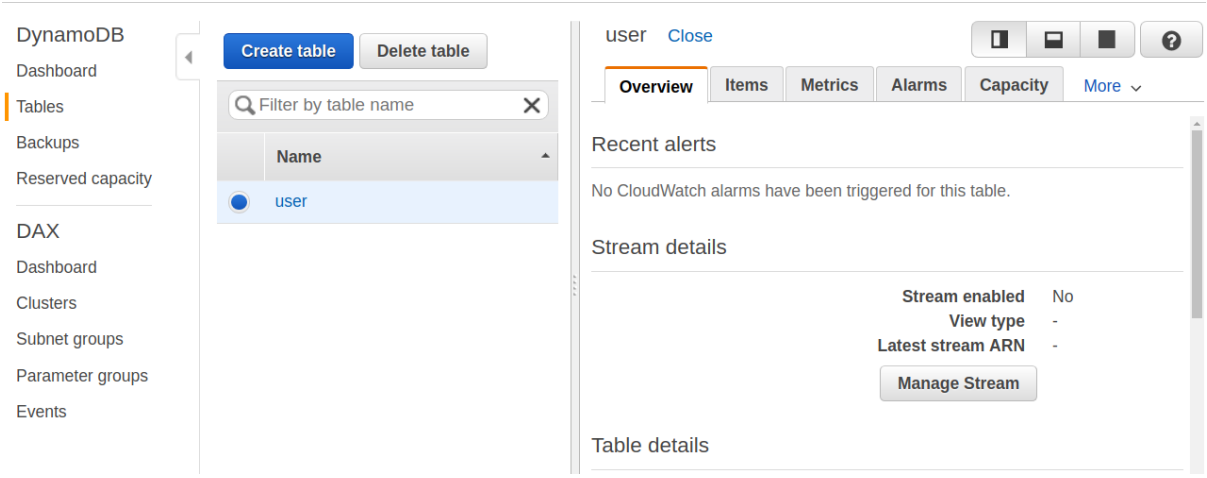
Read capacity units	Write capacity units
5	5

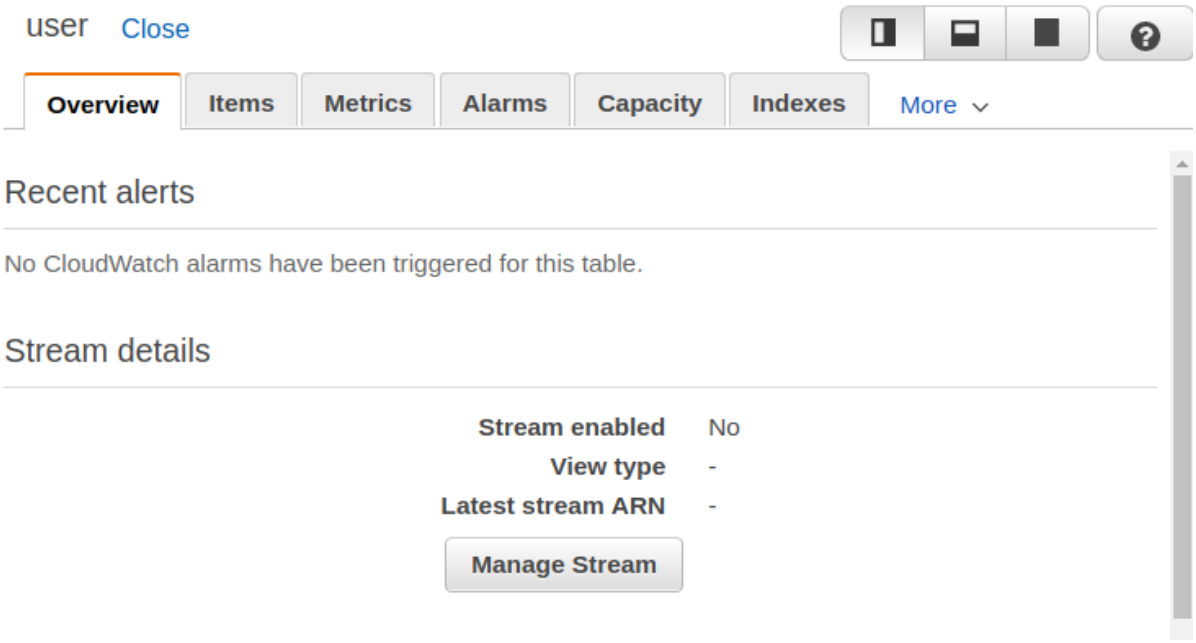
Estimated cost $2.91 / month (Capacity calculator)

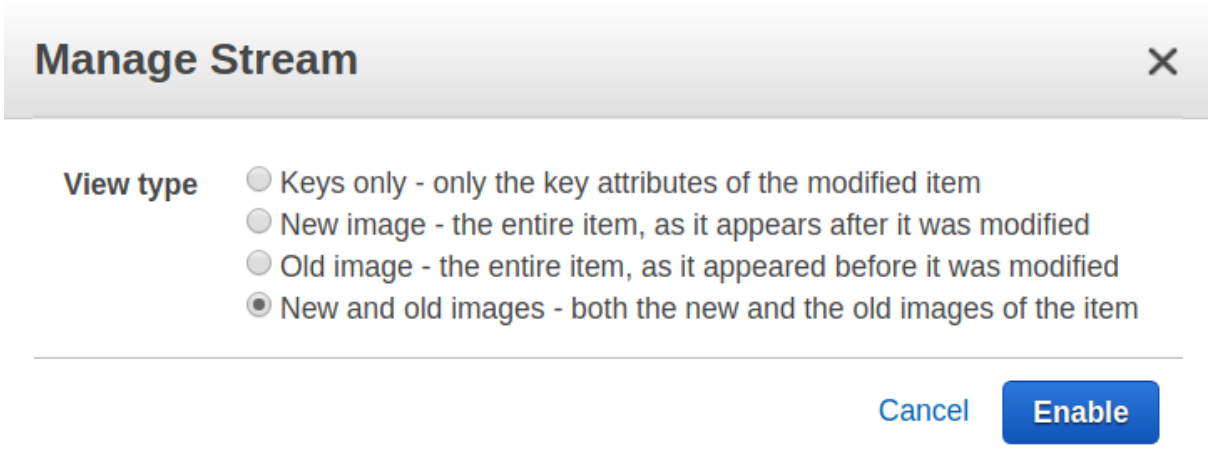
Approximate creation time is **5 minutes**. Additional write capacity may decrease creation time. A notification will be sent to the SNS topic dynamodb once the index creation is complete.
Basic Alarms with 80% upper threshold using SNS topic 'dynamodb' will be automatically created. Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced configuration for alarms can be done in the alarms tab.

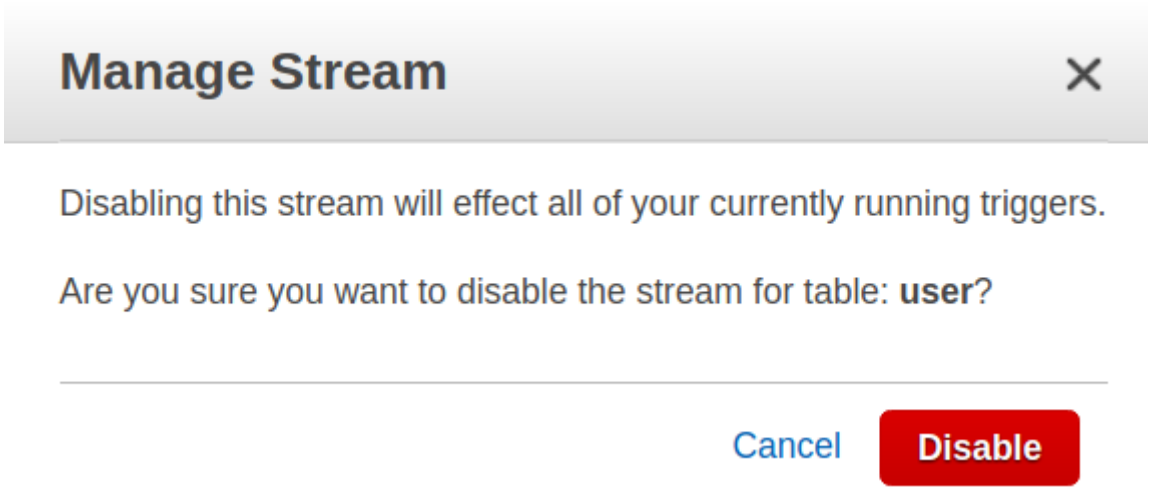
Cancel | Create index

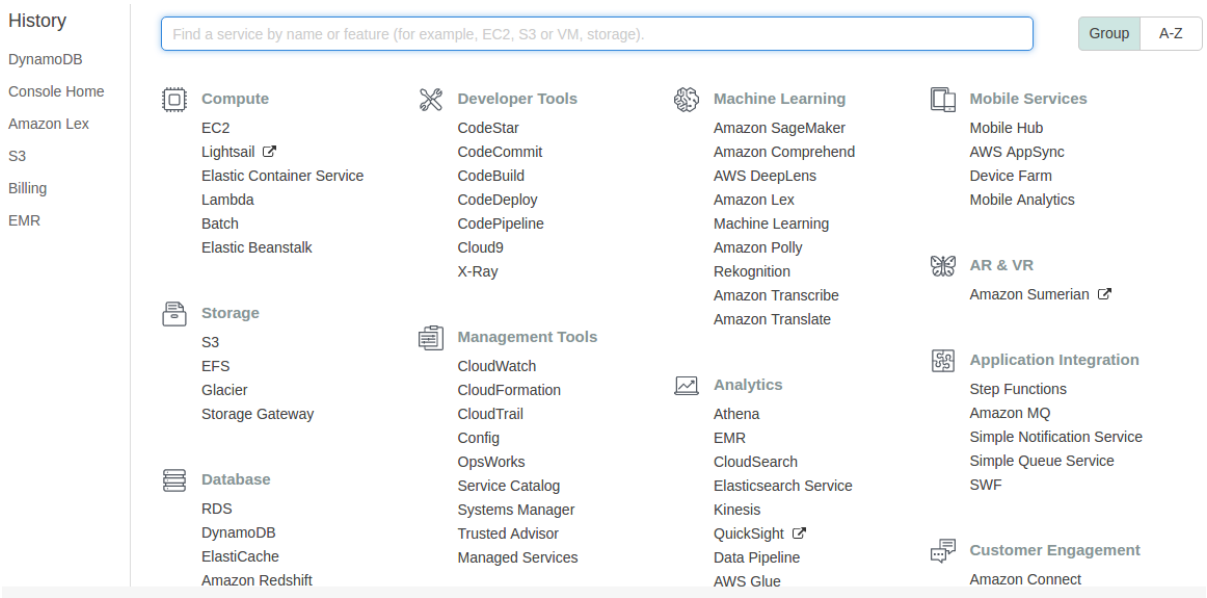

Application
Dynamo DB web API
Dynamo DB stream web API
Dynamo DB
Dynamo DB stream

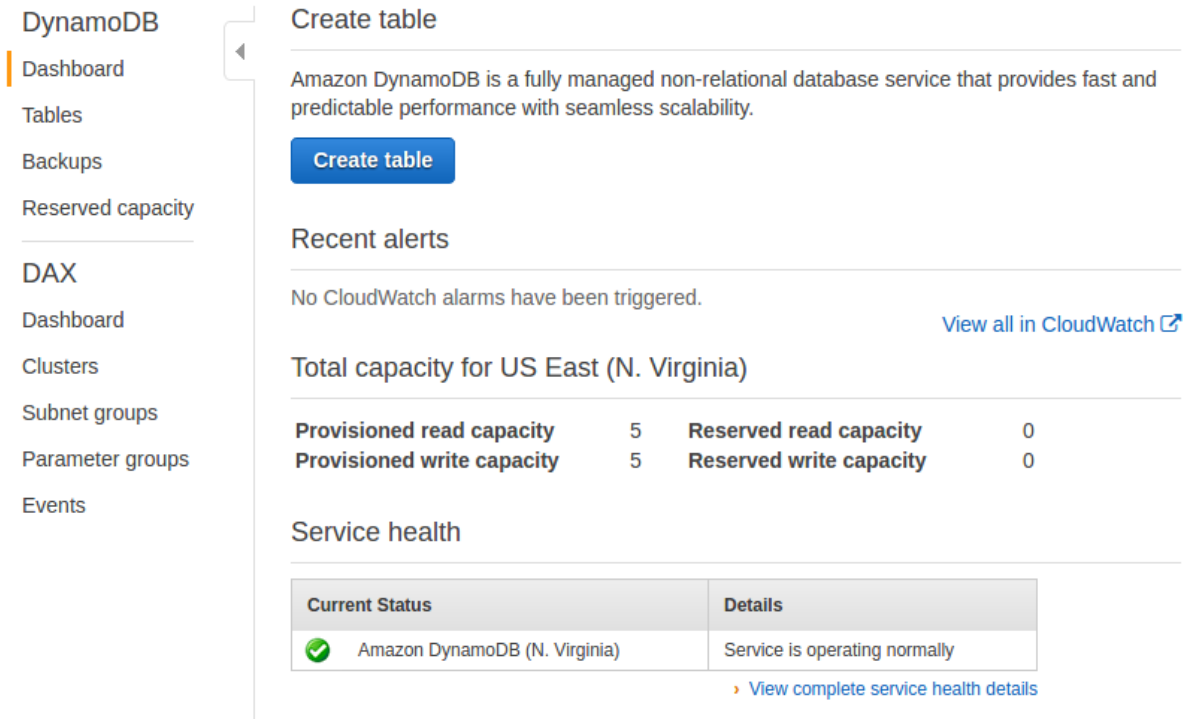

DynamoDB
Dashboard
Tables
Backups
Reserved capacity
DAX
Dashboard
Clusters
Subnet groups
Parameter groups
Events
Create table
Delete table
Filter by table name
Name
user
user
Close
Overview
Items
Metrics
Alarms
Capacity
More
Recent alerts
No CloudWatch alarms have been triggered for this table.
Stream details
Stream enabled
No
View type
-
Latest stream ARN
-
Manage Stream
Table details


user
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
More
Recent alerts
No CloudWatch alarms have been triggered for this table.
Stream details
Stream enabled
No
View type
-
Latest stream ARN
-
Manage Stream


Manage Stream
View type
Keys only - only the key attributes of the modified item
New image - the entire item, as it appears after it was modified
Old image - the entire item, as it appeared before it was modified
New and old images - both the new and the old images of the item
Cancel
Enable


Manage Stream
Disabling this stream will effect all of your currently running triggers.
Are you sure you want to disable the stream for table: user?
Cancel
Disable


History
DynamoDB
Console Home
Amazon Lex
S3
Billing
EMR
Find a service by name or feature (for example, EC2, S3 or VM, storage).
Group
A-Z
Compute
EC2
Lightsail
Elastic Container Service
Lambda
Batch
Elastic Beanstalk
Developer Tools
CodeStar
CodeCommit
CodeBuild
CodeDeploy
CodePipeline
Cloud9
X-Ray
Machine Learning
Amazon SageMaker
Amazon Comprehend
AWS DeepLens
Amazon Lex
Machine Learning
Amazon Polly
Rekognition
Amazon Transcribe
Amazon Translate
Mobile Services
Mobile Hub
AWS AppSync
Device Farm
Mobile Analytics
AR & VR
Amazon Sumerian
Storage
S3
EFS
Glacier
Storage Gateway
Management Tools
CloudWatch
CloudFormation
CloudTrail
Config
OpsWorks
Service Catalog
Systems Manager
Trusted Advisor
Managed Services
Analytics
Athena
EMR
CloudSearch
Elasticsearch Service
Kinesis
QuickSight
Data Pipeline
AWS Glue
Application Integration
Step Functions
Amazon MQ
Simple Notification Service
Simple Queue Service
SWF
Database
RDS
DynamoDB
ElastiCache
Amazon Redshift
Customer Engagement
Amazon Connect


DynamoDB
Dashboard
Tables
Backups
Reserved capacity
DAX
Dashboard
Clusters
Subnet groups
Parameter groups
Events
Create table
Amazon DynamoDB is a fully managed non-relational database service that provides fast and predictable performance with seamless scalability.
Create table
Recent alerts
No CloudWatch alarms have been triggered.
View all in CloudWatch
Total capacity for US East (N. Virginia)
Provisioned read capacity 5
Provisioned write capacity 5
Reserved read capacity 0
Reserved write capacity 0
Service health
Current Status
Details
Amazon DynamoDB (N. Virginia)
Service is operating normally
View complete service health details

Create DynamoDB table

Tutorial

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* customer

Primary key* Partition key

customer_id Number

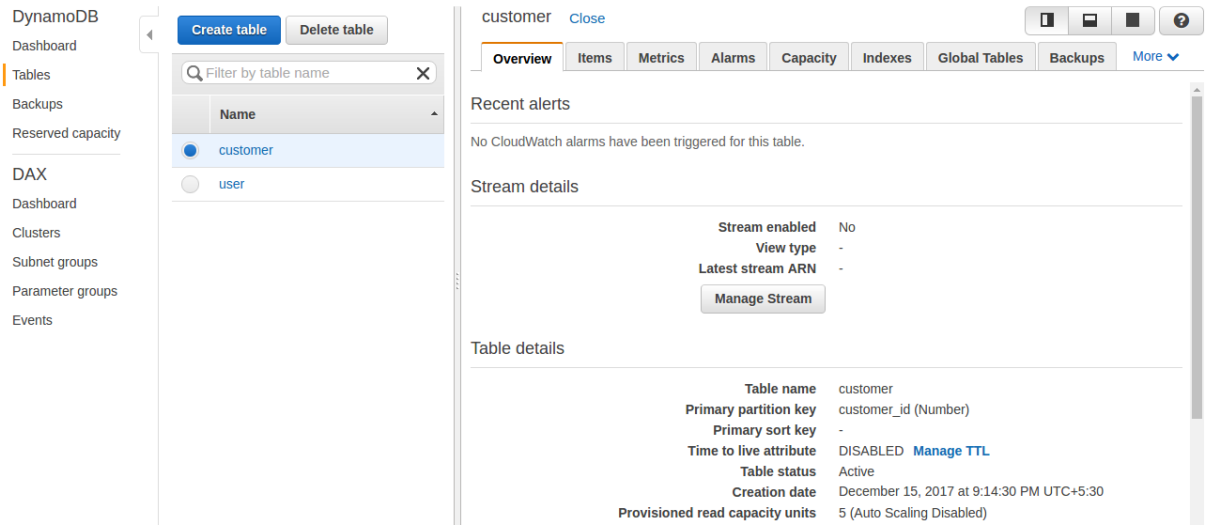
Add sort key

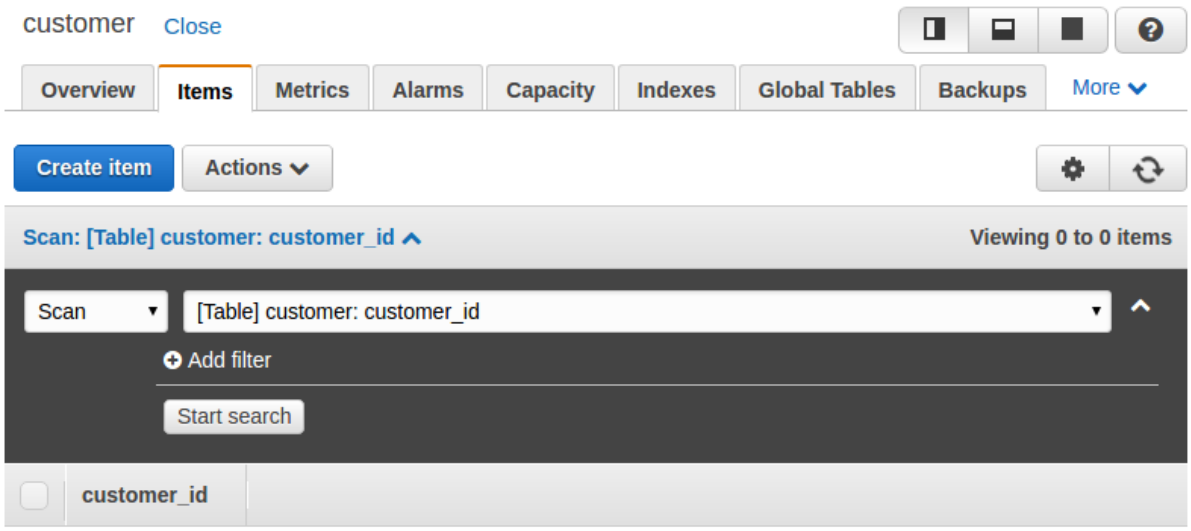
Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

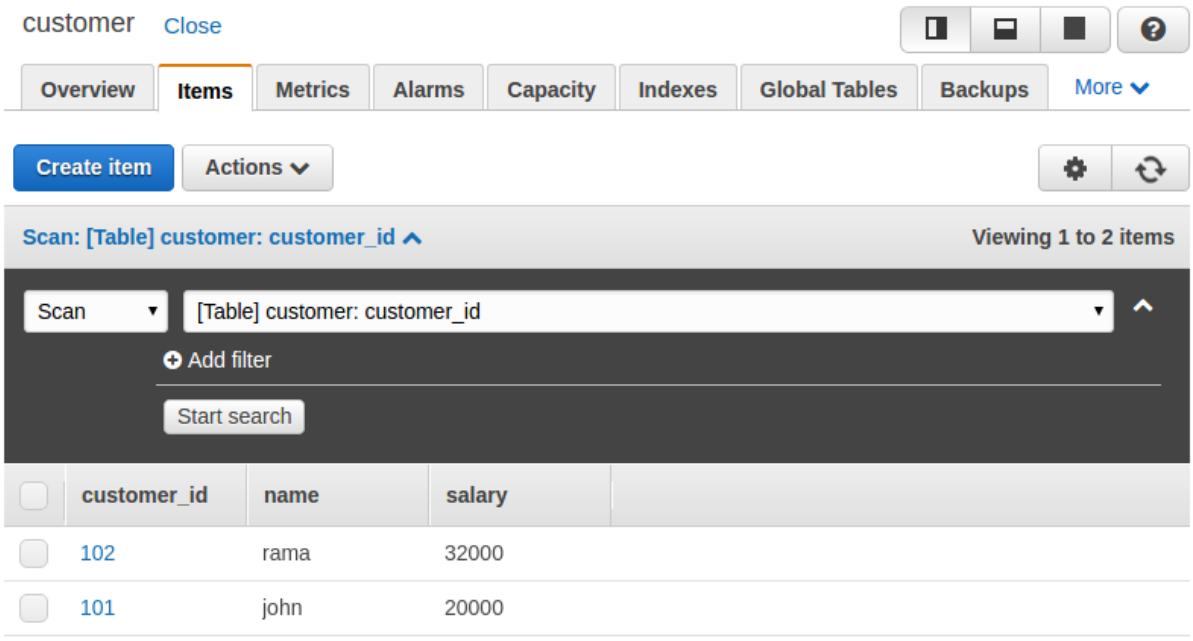
Use default settings

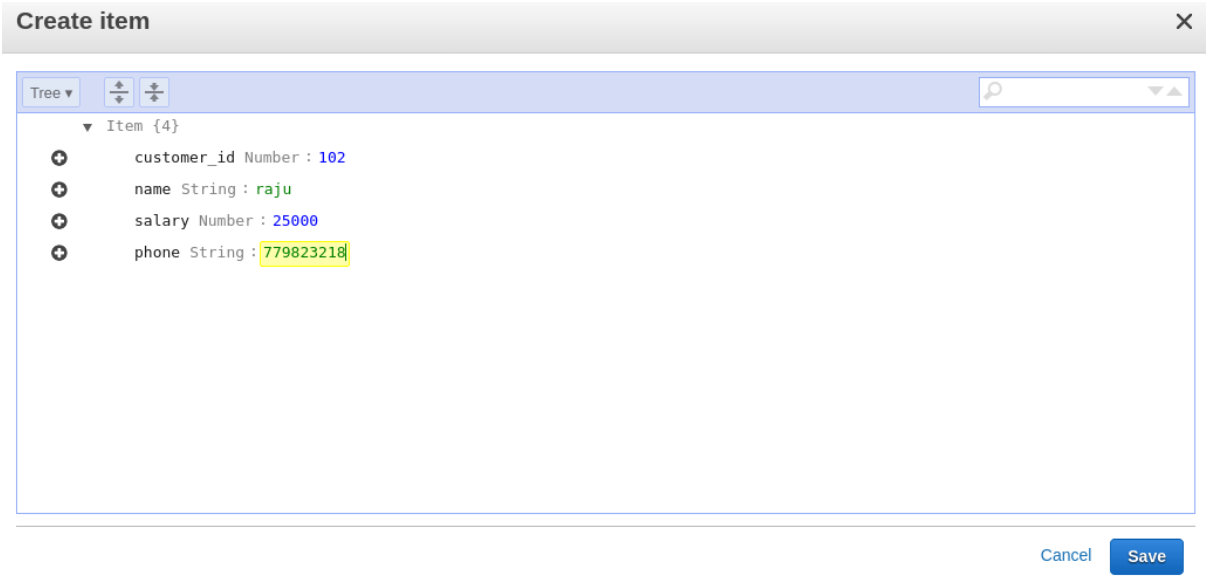
- No secondary indexes.
- Provisioned capacity set to 5 reads and 5 writes.
- Basic alarms with 80% upper threshold using SNS topic "dynamodb".
  On-Demand Backup and Restore Enabled NEW!

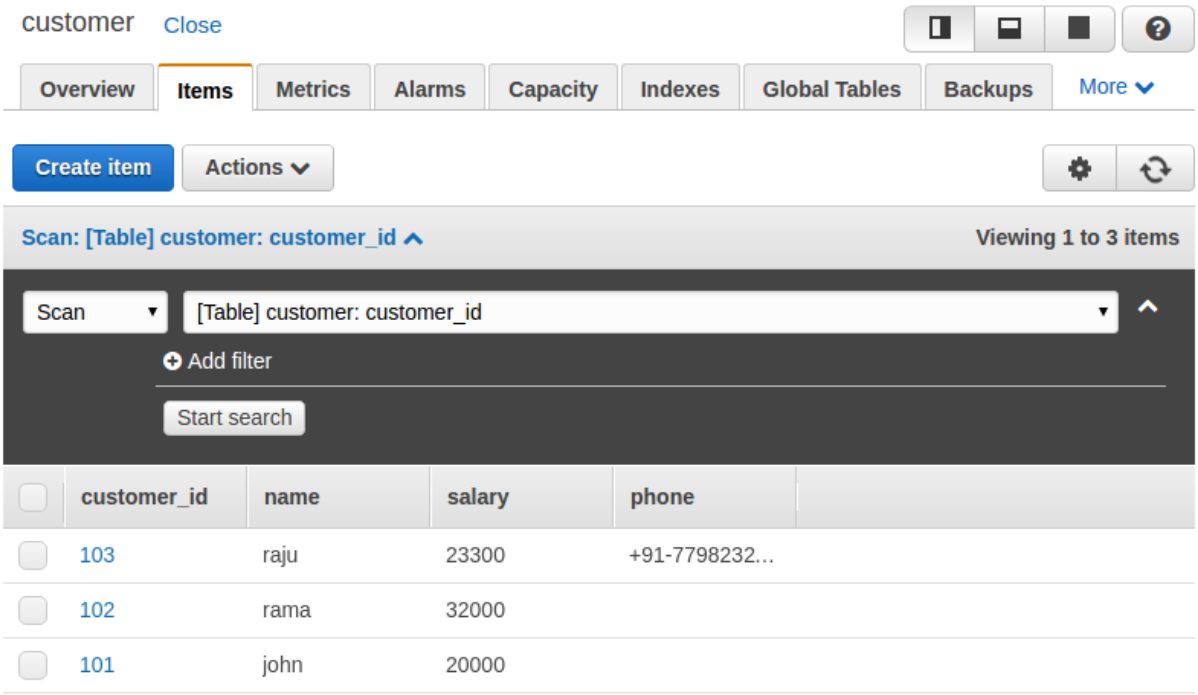


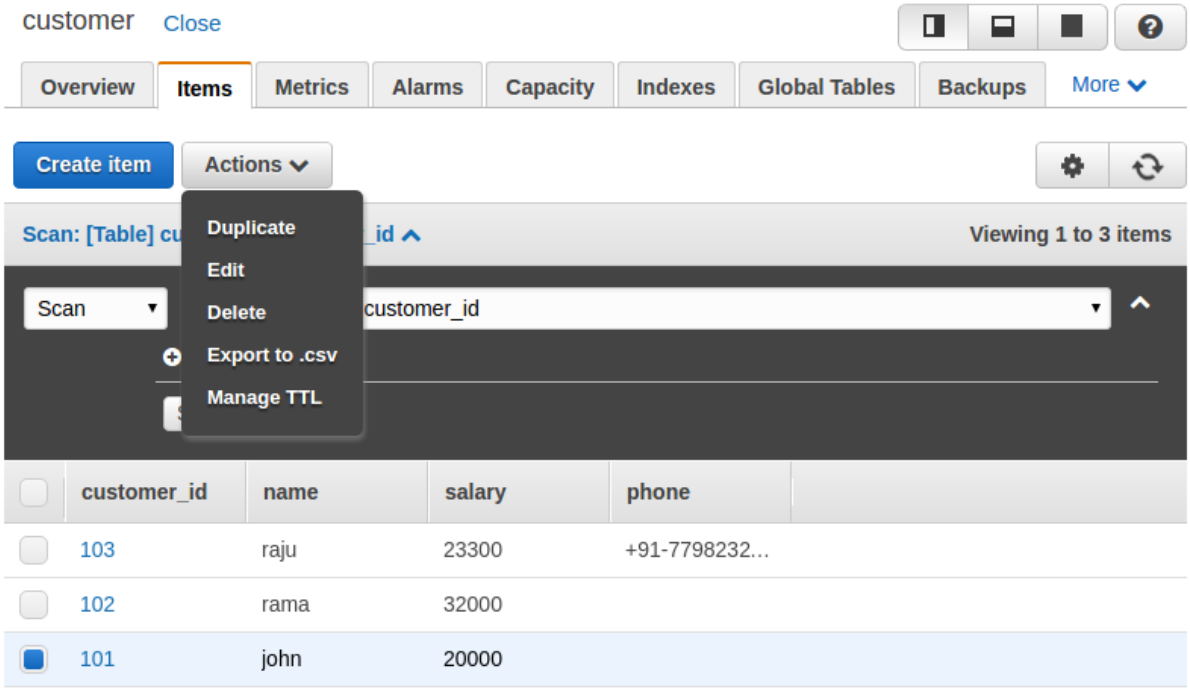


An item consists of one or more attributes. Each attribute consists of a name, a data type, and a value. When you read or write an item, the only attributes that are required are those that make up the primary key. More info

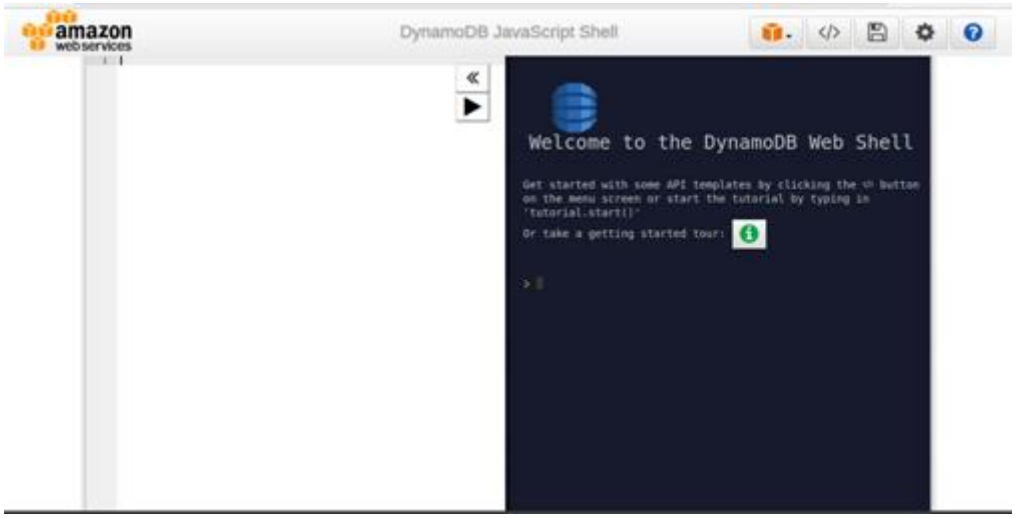

customer
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
More
Create item
Actions
Scan: [Table] customer: customer_id
Viewing 1 to 2 items
Scan
[Table] customer: customer_id
Add filter
Start search
customer_id
name
salary
102
rama
32000
101
john
20000

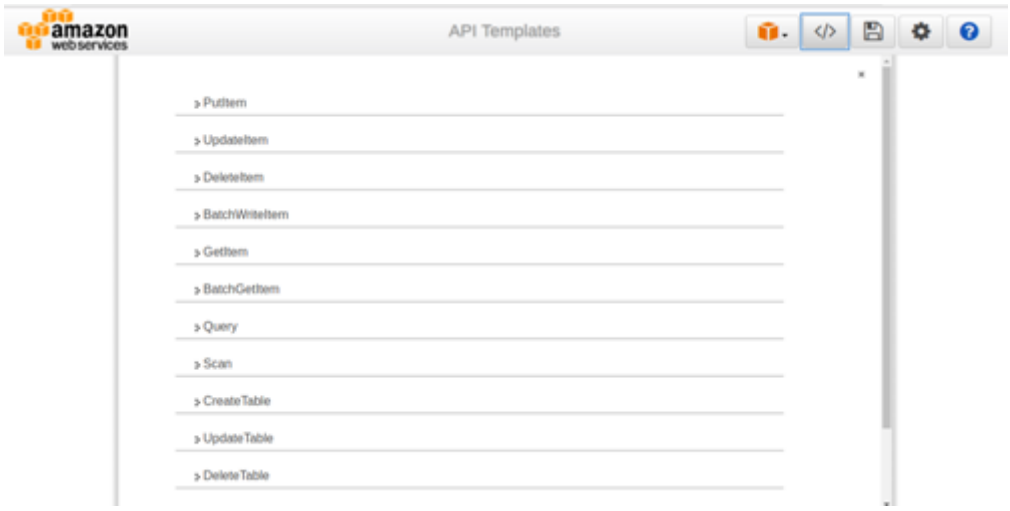

Create item
Tree
Item {4}
customer_id Number : 102
name String : raju
salary Number : 25000
phone String : 779823218
Cancel
Save


customer
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
More
Create item
Actions
Scan: [Table] customer: customer_id
Viewing 1 to 3 items
Scan
[Table] customer: customer_id
Add filter
Start search
customer_id
name
salary
phone
103
raju
23300
+91-7798232...
102
rama
32000
101
john
20000


customer
Close
Overview
Items
Metrics
Alarms
Capacity
Indexes
Global Tables
Backups
More
Create item
Actions
Duplicate
Edit
Delete
Export to .csv
Manage TTL
Viewing 1 to 3 items
Scan
customer_id
customer_id
name
salary
phone
103
raju
23300
+91-7798232...
102
rama
32000
101
john
20000


Edit item
Tree
Item {3}
customer_id Number : 101
name String : john
salary Number : 23000
Cancel
Save


amazon web services
DynamoDB JavaScript Shell
Welcome to the DynamoDB Web Shell
Get started with some API templates by clicking the </> button on the menu screen or start the tutorial by typing in 'tutorial.start()'
Or take a getting started tour:


amazon web services
API Templates
PutItem
UpdateItem
DeleteItem
BatchWriteItem
GetItem
BatchGetItem
Query
Scan
CreateTable
UpdateTable
DeleteTable

```
var params = {
    Limit: 100,
};
dynamodb.listTables(params, function(err, data) {
    if (err) ppJson(err);
    else ppJson(data);
});
```

```
"message":"Limit must be greater than or equal to
1"
"code":"ValidationException"
"time":"2018-03-08T01:09:09.801Z"
"statusCode":400
"retryable":false
```

```
var params = {
TableName: 'person',
KeySchema: [
    {
        AttributeName: 'PersonID',
        KeyType: 'HASH'
    },
],
AttributeDefinitions: [
    {
        AttributeName: 'PersonID',
        AttributeType: 'N',
    }
],
ProvisionedThroughput: {
    ReadCapacityUnits: 10,
    WriteCapacityUnits: 10,
},
};

dynamodb.createTable(params, function(err, data) {
    if (err) ppJson(err); // an error occurred
    else ppJson(data); // successful response

});
```

```
"TableDescription" {
  "AttributeDefinitions" [
    0: {
        "AttributeName":"PersonID"
        "AttributeType":"N"
  "TableName":"person"
  "KeySchema" [
    0: {
        "AttributeName":"PersonID"
        "KeyType":"HASH"
  "TableStatus":"ACTIVE"
  "CreationDateTime":"2018-03-08T01:14:17.341Z"
  "ProvisionedThroughput" {
      "LastIncreaseDateTime":"1970-01-
      01T00:00:00.000Z"
      "LastDecreaseDateTime":"1970-01-
      01T00:00:00.000Z"
      "NumberOfDecreasesToday":0
      "ReadCapacityUnits":10
      "WriteCapacityUnits":10
  "TableSizeBytes":0
  "ItemCount":0
  "TableArn":"arn:aws:dynamodb:ddblocal:0000000000
  00:table/person"
```

```
var params = {
    TableName: 'person',
    Item: {
        PersonID: 101,
        FirstName : "John",
        LastName : "Smith",
        Age : 27,
        Email : "john.smith@co.in"
    }
};
docClient.put(params, function(err, data) {
    if (err) ppJson(err);
    else ppJson(data);
});
```

```
var params = {};
docClient.put(params, function(err, data) {});
```

```
var params = {
    TableName: 'person',
    Key: {
        PersonID: 101
    },
};
docClient.get(params, function(err, data) {
    if (err) ppJson(err);
    else ppJson(data);
});
```

```
"Item" {
    "LastName":"Smith"
    "Email":"john.smith@co.in"
    "PersonID":101
    "FirstName":"John"
    "Age":27
```

```
var params = {
    TableName: 'person',
    Select: 'ALL_ATTRIBUTES'
};
dynamodb.scan(params, function(err, data) {
    if (err) ppJson(err); // an error occurred
    else ppJson(data); // successful response
});
```


```
var params = {
    TableName: 'person',
    Key: {
        PersonID: 101,
    },
    UpdateExpression: 'SET Age= :age',
    ExpressionAttributeValues:{
        ":age":32,
    },
};
docClient.update(params, function(err, data) {
    if (err) ppJson(err); // an error occurred
    else ppJson(data); // successful response
});
```


```
var params = {
    TableName: 'person',
    Key: {
        PersonID: 101,
    },
};
docClient.delete(params, function(err, data) {
    if (err) ppJson(err); // an error occurred
    else ppJson(data); // successful response
});
```

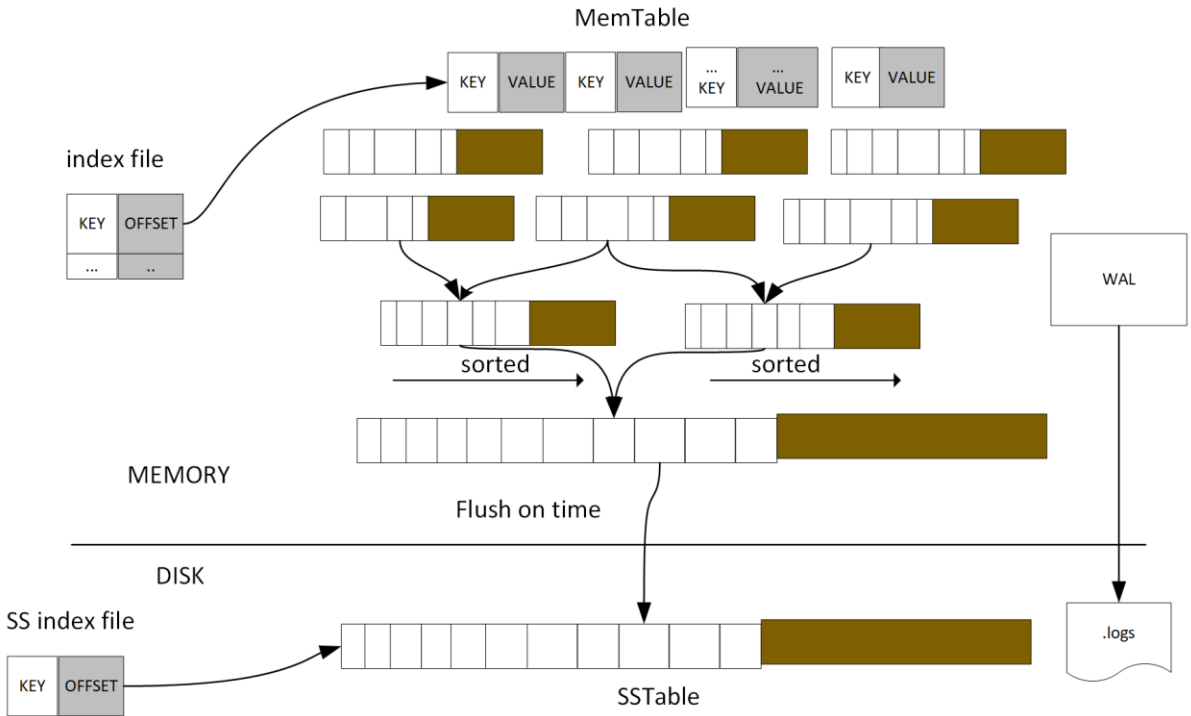
Chapter 8: InfluxDB

```
+----------------------+---------+--------------+---------+---------+---------+--------+--------+
| time                 | close   | company      | high    | low     | open    | ticker | volume |
+----------------------+---------+--------------+---------+---------+---------+--------+--------+
| 2017-11-22T14:30:00Z | 1051.16 | Alphabet Inc | 1051.16 | 1051.16 | 1051.16 | GOOGL  | 13622  |
| 2017-11-22T14:30:00Z |  173.45 | Apple Inc    |  173.45 |  173.34 |  173.36 | AAPL   | 300218 |
| 2017-11-22T14:31:00Z | 1051.46 | Alphabet Inc | 1051.78 | 1051.11 | 1051.42 | GOOGL  | 1404   |
| 2017-11-22T14:31:00Z |  173.70 | Apple Inc    |  173.71 |  173.35 |  173.39 | AAPL   | 165442 |
| 2017-11-22T14:32:00Z | 1051.07 | Alphabet Inc | 1051.57 | 1050.72 | 1051.53 | GOOGL  | 7014   |
| 2017-11-22T14:32:00Z |  173.56 | Apple Inc    |  173.80 |  173.51 |  173.70 | AAPL   | 175809 |
| 2017-11-22T14:33:00Z | 1051.35 | Alphabet Inc | 1051.53 | 1050.61 | 1051.00 | GOOGL  | 5973   |
| 2017-11-22T14:33:00Z |  173.62 | Apple Inc    |  173.75 |  173.56 |  173.57 | AAPL   | 109326 |
| 2017-11-22T14:34:00Z | 1051.33 | Alphabet Inc | 1051.75 | 1051.27 | 1051.75 | GOOGL  | 1500   |
| 2017-11-22T14:34:00Z |  173.91 | Apple Inc    |  173.93 |  173.61 |  173.62 | AAPL   | 218830 |
+----------------------+---------+--------------+---------+---------+---------+--------+--------+
```

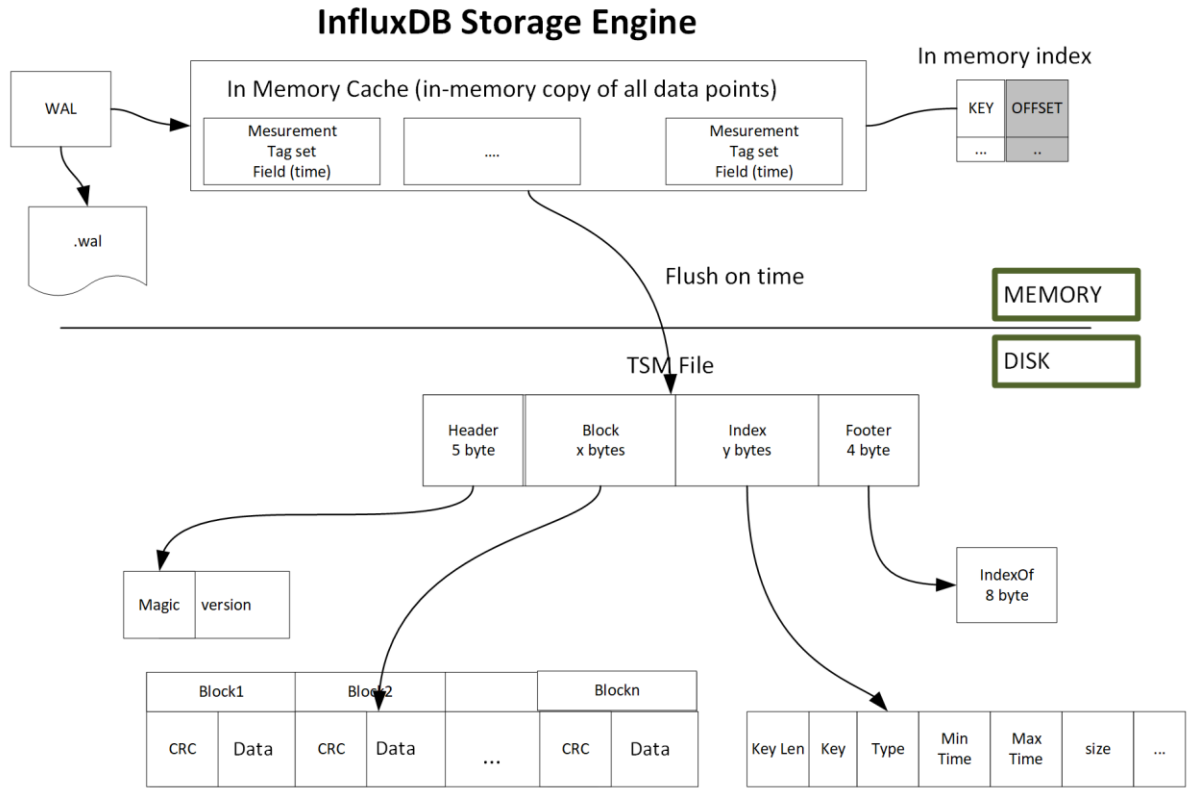
```
name: tickers
tags: company=Alphabet Inc, ticker=GOOGL
time                 close   high    low     open    volume
----                 -----   ----    ---     ----    ------
2017-11-22T14:30:00Z 1051.16 1051.16 1051.16 1051.16 13622
2017-11-22T14:31:00Z 1051.46 1051.78 1051.11 1051.42 1404
2017-11-22T14:32:00Z 1051.07 1051.57 1050.72 1051.53 7014
2017-11-22T14:33:00Z 1051.35 1051.53 1050.61 1051    5973
2017-11-22T14:34:00Z 1051.33 1051.75 1051.27 1051.75 1500

name: tickers
tags: company=Apple Inc, ticker=AAPL
time                 close  high   low    open   volume
----                 -----  ----   ---    ----   ------
2017-11-22T14:30:00Z 173.45 173.45 173.34 173.36 300218
2017-11-22T14:31:00Z 173.7  173.71 173.35 173.39 165442
2017-11-22T14:32:00Z 173.56 173.8  173.51 173.7  175809
2017-11-22T14:33:00Z 173.62 173.75 173.56 173.57 109326
2017-11-22T14:34:00Z 173.91 173.93 173.61 173.62 218830
```

```
time                 close  high   low    open   volume
----                 -----  ----   ---    ----   ------
2017-11-22T14:30:00Z 173.45 173.45 173.34 173.36 300218
```

```
name    duration shardGroupDuration replicaN default
----    -------- ------------------ -------- -------
autogen 0s       168h0m0s                 1    true
```

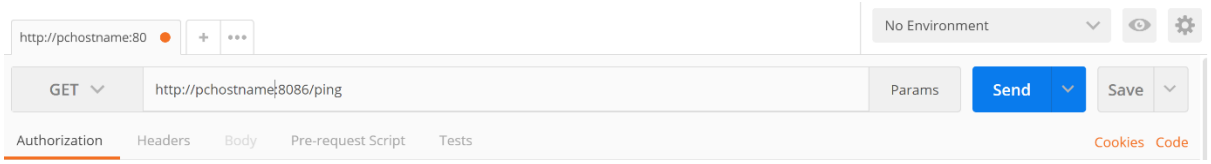


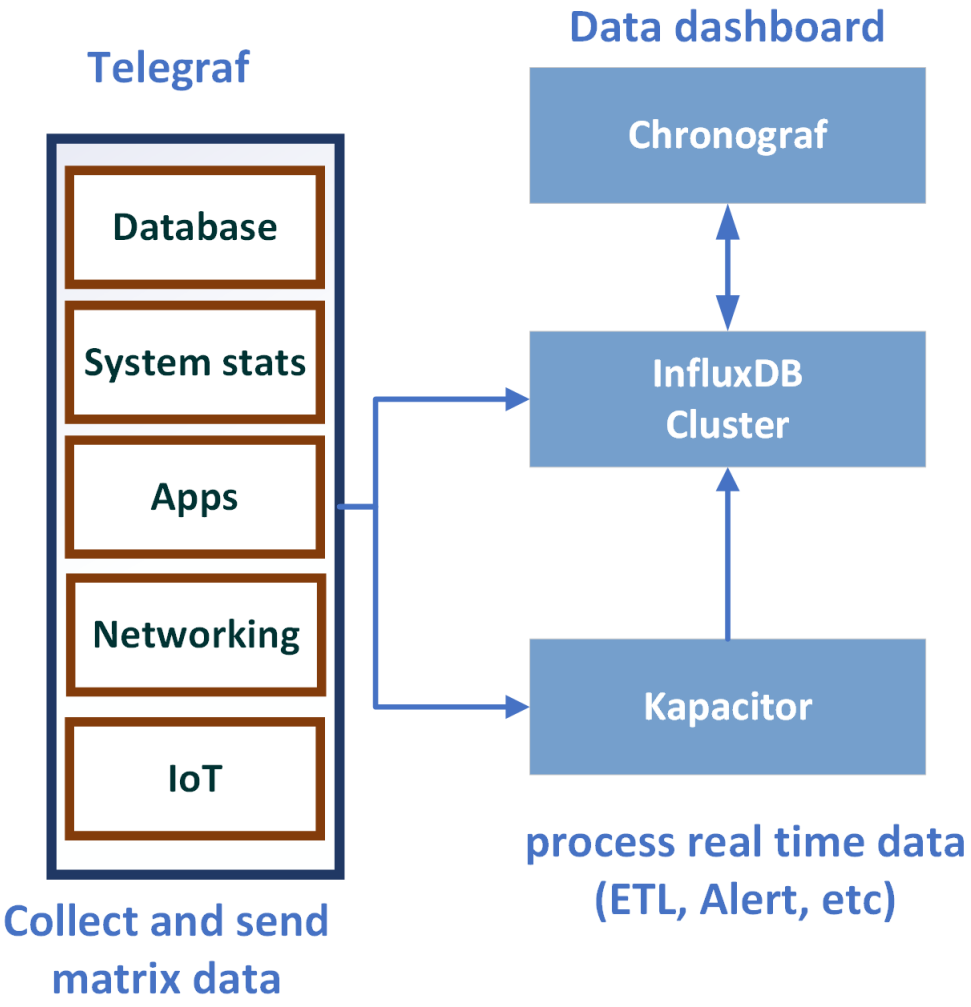
LSM - Continuous merge sort and create larger file

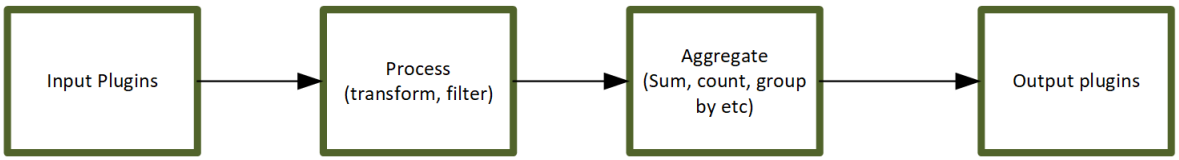


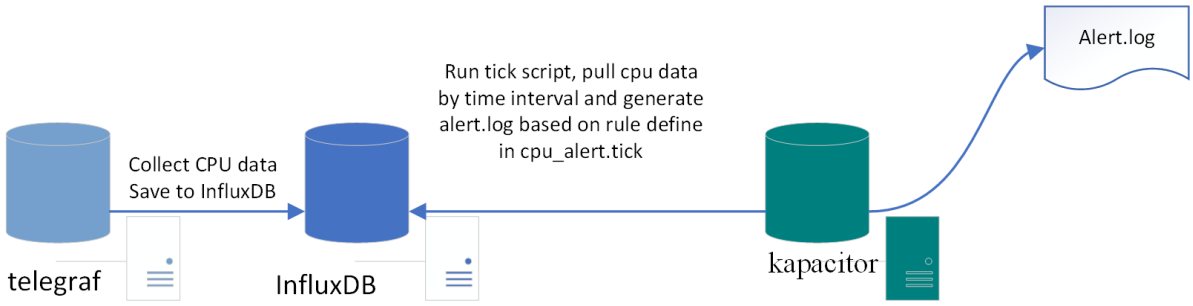
```
> SELECT HeapAlloc, HeapIdle, HeapInUse, NumGC, Mallocs, TotalAlloc from runtime limit 3;
name: runtime
time                 HeapAlloc HeapIdle HeapInUse NumGC Mallocs TotalAlloc
----                 --------- -------- --------- ----- ------- ----------
2017-11-27T04:39:00Z 2818392   196608   4521984   1     80921   4652552
2017-11-27T04:39:10Z 2835344   1081344  4685824   2     94765   5918456
2017-11-27T04:39:20Z 3265608   679936   5087232   2     96718   6348720
```

```
> create database market;
> exit
```










```
● kapacitor.service - Time series data processing engine.
   Loaded: loaded (/lib/systemd/system/kapacitor.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2017-12-02 20:04:18 EST; 4min 52s ago
     Docs: https://github.com/influxdb/kapacitor
 Main PID: 5071 (kapacitord)
   CGroup: /system.slice/kapacitor.service
           └─5071 /usr/bin/kapacitord -config /etc/kapacitor/kapacitor.conf

Dec 02 20:04:18 bwu-pc systemd[1]: Started Time series data processing engine..
Dec 02 20:04:18 bwu-pc kapacitord[5071]: '##:::'##:::'###:::'########:::::'###:::::'######::'####:'########::'#######::'########::
Dec 02 20:04:18 bwu-pc kapacitord[5071]:  ##::'##:::'## ##::: ##.... ##::'## ##::'##... ##:. ##::... ##..::'##.... ##: ##.... ##:
Dec 02 20:04:18 bwu-pc kapacitord[5071]:  ##:'##:::'##:. ##:: ##:::: ##::'##:. ##:: ##:::..:::: ##::::: ##:::: ##:: ##:::: ##:
Dec 02 20:04:18 bwu-pc kapacitord[5071]:  #####:::'##::. ##: ########::'##::. ##: ##::::::::: ##::::: ##:::: ##:: ##:::: ##:
Dec 02 20:04:18 bwu-pc kapacitord[5071]:  ##. ##::: ########: ##.....::: ########: ##::::::::: ##::::: ##:::: ##:: ##:::: ##:
Dec 02 20:04:18 bwu-pc kapacitord[5071]:  ##:. ##:: ##.... ##: ##::::::::: ##.... ##: ##::: ##:: ##::::: ##:::: ##:: ##:::: ##:
Dec 02 20:04:18 bwu-pc kapacitord[5071]:  ##::. ##: ##:::: ##: ##::::::::: ##:::: ##:. ######::'####::::: ##:::::. #######:: ##:::: ##:
Dec 02 20:04:18 bwu-pc kapacitord[5071]: ..::::..:::..:::::..::..:::::::::..:::::..:::......:::....:::::..:::::::..:::::::..:::::..::
Dec 02 20:04:18 bwu-pc kapacitord[5071]: 2017/12/02 20:04:18 Using configuration at: /etc/kapacitor/kapacitor.conf
```