Pruebas de particionamiento de bases de datos NoSQL.

Actividad 4 – Unidad 3

Nombre de los estudiantes

Santiago Herrera Rocha

Docente

Jorge Isaac Castañeda Valbuena

CORPORACIÓN UNIVERSITARIA IBEROAMERICANA

INGENIERÍA DE SOFTWARE, FACULTAD DE INGENIERÍA

BASES DE DATOS AVANZADAS

Bogotá, D.C.

15 de diciembre de 2024

Desarrollo

1. Casos de Prueba

Caso de Prueba 1: CP001

Descripción: Validar que el tiempo de respuesta para consultas filtradas por la clave de partición ronda sea menor a 100 ms con 1 millón de registros.

Requerimiento Asociado: Tiempo de respuesta menor a 100 ms.

Preparación:

- Clave de partición: ronda.
- Generar 1 millón de registros distribuidos uniformemente entre las rondas.

Pasos de Ejecución:

Conectar al router:

```
mongosh --host localhost --port 27020
```

Insertar datos de prueba:

```
let bulk = [];
for (let i = 1; i <= 1000000; i++) {
    bulk.push({
        ronda: i % 10,  // Distribución uniforme entre 0 y 9
        equipo_local: "Equipo" + (i % 100),
```

```
            equipo_visitante: "Equipo" + ((i + 1) % 100),
            goles_local: Math.floor(Math.random() * 5),
            goles_visitante: Math.floor(Math.random() * 5)
        });
        if (bulk.length === 1000) {
            db.encuentros.insertMany(bulk);
            bulk = [];
        }
    }
    if (bulk.length > 0) db.encuentros.insertMany(bulk);
```

Ejecutar una consulta desde el router para filtrar los encuentros de una ronda específica:

```
db.encuentros.find({ ronda: 5 }).explain("executionStats");
```

Registrar el tiempo de ejecución reportado por executionStats.

Resultado Esperado: Tiempo de respuesta < 100 ms.

Caso de Prueba 2: CP002

Descripción: Verificar que los datos de la colección jugadores estén distribuidos equitativamente entre los nodos según la clave de partición equipo.

Requerimiento Asociado: Eficiencia en el almacenamiento.

Preparación:

- Clave de partición: equipo.
- Insertar datos para 10,000 jugadores distribuidos en 100 equipos.

Pasos de Ejecución:

Conectar al router:

```
mongosh --host localhost --port 27020
```

Insertar datos de prueba:

```
let bulk = [];
for (let i = 1; i <= 10000; i++) {
    bulk.push({
        nombre: "Jugador" + i,
        equipo: "Equipo" + (i % 100),
        posicion: ["Portero", "Defensa", "Mediocampista",
"Delantero"][i % 4]
    });
    if (bulk.length === 1000) {
        db.jugadores.insertMany(bulk);
        bulk = [];
    }
```

```
}

if (bulk.length > 0) db.jugadores.insertMany(bulk);
```

Consultar la distribución de documentos en los shards:

```
sh.status();
```

Resultado Esperado: Los datos deben distribuirse uniformemente con una desviación menor al 10%.

Caso de Prueba 3: CP003

Descripción: Comprobar que el sistema sigue operativo tras la caída de un nodo.

Requerimiento Asociado: Alta disponibilidad.

Preparación: Simular la caída de un nodo mediante StepDown o detener el servicio de un shard.

Pasos de Ejecución:

Simular la caída de un nodo:

```
mongosh --host localhost --port 27018

db.adminCommand( { replSetStepDown: 120,
secondaryCatchUpPeriodSecs: 15, force: true }
```

Conectar al router:

```
mongosh --host localhost --port 27020
```

Ejecutar una consulta desde el router:

```
db.encuentros.find({ ronda: 1 });
```

Observar si se obtiene un resultado válido desde otro nodo.

Resultado Esperado: La consulta debe ejecutarse correctamente con los datos disponibles en los otros nodos.

Caso de Prueba 4: CP004

Descripción: Validar que, al añadir un nuevo nodo, el sistema redistribuye los datos automáticamente.

Requerimiento Asociado: Escalabilidad horizontal.

Preparación: Configurar un nuevo nodo shard en el clúster.

Pasos de Ejecución:

Configurar un nuevo shard:

```
mongod --shardsvr --replSet shard3RS --dbpath shard3 --port 27016
```

```
mongosh --host localhost --port 27016
```

```
rs.initiate({ _id: "shard3RS", members: [{ _id: 0, host:
"localhost:27016" }] });
```

Conectar al router:

```
mongosh --host localhost --port 27020
```

Añadir el shard al router:

```
sh.addShard("shard3RS/localhost:27016");
```

Consultar el estado del clúster y balanceo de datos:

```
sh.status();
```

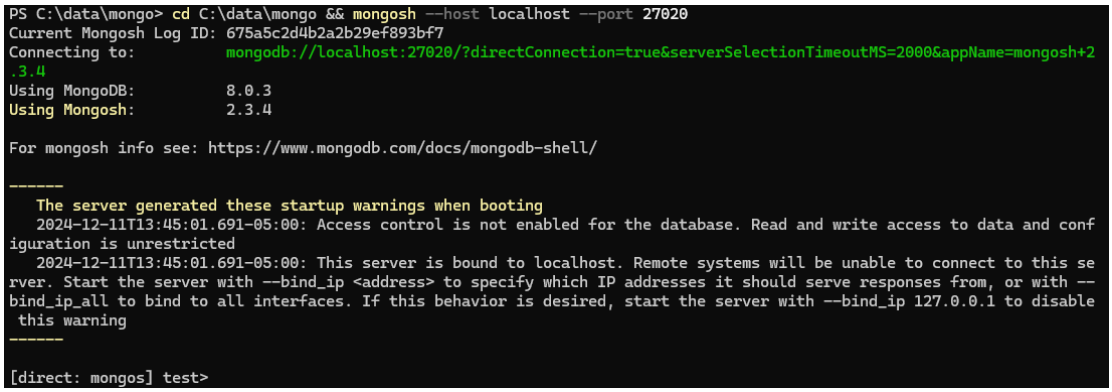
Resultado Esperado: El nuevo nodo debe recibir datos automáticamente para balancear la carga.

2. Ejecución de los Casos de Prueba

Caso de Prueba 1: CP001

Pasos de Ejecución:

Conectar al router:



Insertar datos de prueba:

```
[direct: mongos] torneo_futbol> let bulk = []; for (let i = 1; i <= 1000000; i++) {
...     bulk.push({
...         ronda: i % 10,  // Distribución uniforme entre 0 y 9
...         equipo_local: "Equipo" + (i % 100),
...         equipo_visitante: "Equipo" + ((i + 1) % 100),
...         goles_local: Math.floor(Math.random() * 5),
...         goles_visitante: Math.floor(Math.random() * 5)
...     });
...     if (bulk.length === 1000) {
...         db.encuentros.insertMany(bulk);
...         bulk = [];
...     }
... } if (bulk.length > 0) db.encuentros.insertMany(bulk);
[]
[direct: mongos] torneo_futbol>
```

Ejecutar una consulta desde el router para filtrar los encuentros de una ronda específica:

```
[direct: mongos] torneo_futbol> db.encuentros.find({ ronda: 5 }).explain("executionStats");
{
  queryPlanner: {
    winningPlan: {
      stage: 'SHARD_MERGE',
      shards: [
        {
          explainVersion: '1',
          shardName: 'shard1RS',
          connectionString: 'shard1RS/localhost:27018',
          serverInfo: {
            host: 'santiago-s145',
            port: 27018,
            version: '8.0.3',
            gitVersion: '89d97f2744a2b9851ddfb51bdf22f687562d9b06'
          },
          namespace: 'torneo_futbol.encuentros',
          parsedQuery: { ronda: { '$eq': 5 } },
          indexFilterSet: false,
          queryHash: '4447B76C',
          planCacheKey: '7C21A842',
          optimizationTimeMillis: 0,
          maxIndexedOrSolutionsReached: false,
          maxIndexedAndSolutionsReached: false,
          maxScansToExplodeReached: false,
          prunedSimilarIndexes: false,
          winningPlan: {
            isCached: false,
            stage: 'SHARDING_FILTER',
            inputStage: {
```

Registrar el tiempo de ejecución reportado por executionStats.

```
  },
  executionStats: {
    nReturned: 400005,
    executionTimeMillis: 502,
    totalKeysExamined: 400005,
    totalDocsExamined: 400005,
    executionStages: {
      stage: 'SHARD_MERGE',
      nReturned: 400005,
      executionTimeMillis: 502,
      totalKeysExamined: 400005,
      totalDocsExamined: 400005,
      totalChildMillis: Long('1179'),
      shards: [
        {
```

Resultado Esperado: Tiempo de respuesta < 100 ms.

Caso de Prueba 2: CP002

Pasos de Ejecución:

Conectar al router:

Insertar datos de prueba:

```
[direct: mongos] torneo_futbol> let bulk = []; for (let i = 1; i <= 10000; i++) {
...     bulk.push({
...         nombre: "Jugador" + i,
...         equipo: "Equipo" + (i % 100),
...         posicion: ["Portero", "Defensa", "Mediocampista", "Delantero"][i % 4]
...     });
...     if (bulk.length === 1000) {
...         db.jugadores.insertMany(bulk);
...         bulk = [];
...     }
... } if (bulk.length > 0) db.jugadores.insertMany(bulk);
[]
```

Consultar la distribución de documentos en los shards:

```
[direct: mongos] torneo_futbol> db.jugadores.getShardDistribution();
Shard shard2RS at shard2RS/localhost:27017
{
  data: '407KiB',
  docs: 4600,
  chunks: 2,
  'estimated data per chunk': '203KiB',
  'estimated docs per chunk': 2300
}
---
Shard shard1RS at shard1RS/localhost:27018
{
  data: '958KiB',
  docs: 10800,
  chunks: 2,
  'estimated data per chunk': '479KiB',
  'estimated docs per chunk': 5400
}
---
Totals
{
  data: '1.33MiB',
  docs: 15400,
  chunks: 4,
  'Shard shard2RS': [
    '29.82 % data',
    '29.87 % docs in cluster',
    '90B avg obj size on shard'
  ],
  'Shard shard1RS': [
```

Datos de jugadores:

Total de documentos: 10800 + 4600 = 15400

Calcular Desviación para Documentos:

Desviación (%) = ((10800-4600)/15400))*100 ≈ 40.26%

Resultado Esperado: Los datos deben distribuirse uniformemente con una desviación menor al 10%.

Caso de Prueba 3: CP003

Pasos de Ejecución:

Simular la caída de un nodo:

```
PS C:\Users\Santiago> cd C:\data\mongo && mongosh --host localhost --port 27018
Current Mongosh Log ID: 675a2a312b3bd59b27893bf7
Connecting to:          mongodb://localhost:27018/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2
.3.4
Using MongoDB:          8.0.3
Using Mongosh:          2.3.4

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-12-11T13:45:01.085-05:00: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
   2024-12-11T13:45:01.086-05:00: This server is bound to localhost. Remote systems will be unable to connect to this se
rver. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --
bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable
 this warning
------

shard1RS [direct: primary] test> use torneo_futbol
```

```
shard1RS [direct: primary] test> db.adminCommand( { replSetStepDown: 120, secondaryCatchUpPeriodSecs: 15, force: true }
)
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1733963118, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1733963100, i: 3 })
}
```

Conectar al router y ejecutar una consulta desde el router:

```
[direct: mongos] test> use torneo_futbol
switched to db torneo_futbol
[direct: mongos] torneo_futbol> db.encuentros.find({ ronda: 1 });
[
  {
    _id: ObjectId('6759df85763b3ac17d98a548'),
    ronda: 1,
    equipo_local: 'Equipo1',
    equipo_visitante: 'Equipo2',
    goles_local: 3,
    goles_visitante: 0
  },
  {
    _id: ObjectId('6759df85763b3ac17d98a552'),
    ronda: 1,
    equipo_local: 'Equipo11',
    equipo_visitante: 'Equipo12',
    goles_local: 2,
    goles_visitante: 3
  },
  {
    _id: ObjectId('6759df85763b3ac17d98a55c'),
    ronda: 1,
    equipo_local: 'Equipo21',
    equipo_visitante: 'Equipo22',
    goles_local: 4,
    goles_visitante: 0
  },
  {
    _id: ObjectId('6759df85763b3ac17d98a566'),
```

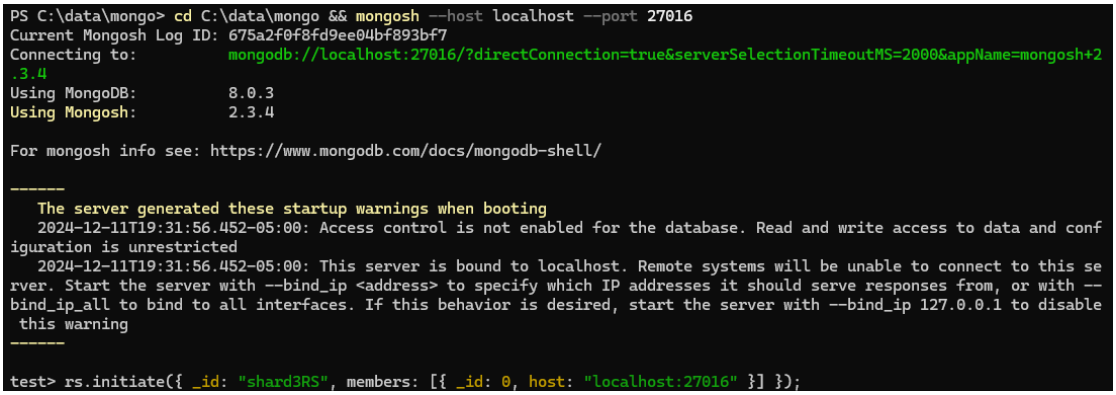
Observar si se obtiene un resultado válido desde otro nodo.

Resultado Esperado: La consulta debe ejecutarse correctamente con los datos disponibles en los otros nodos.

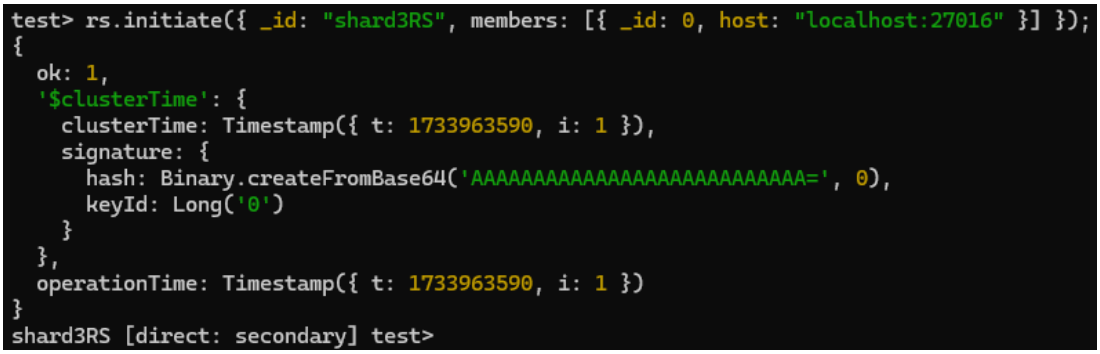
Caso de Prueba 4: CP004

Pasos de Ejecución:

Configurar un nuevo shard y conectarse a él:



Iniciar el nuevo shard tras conectarse:

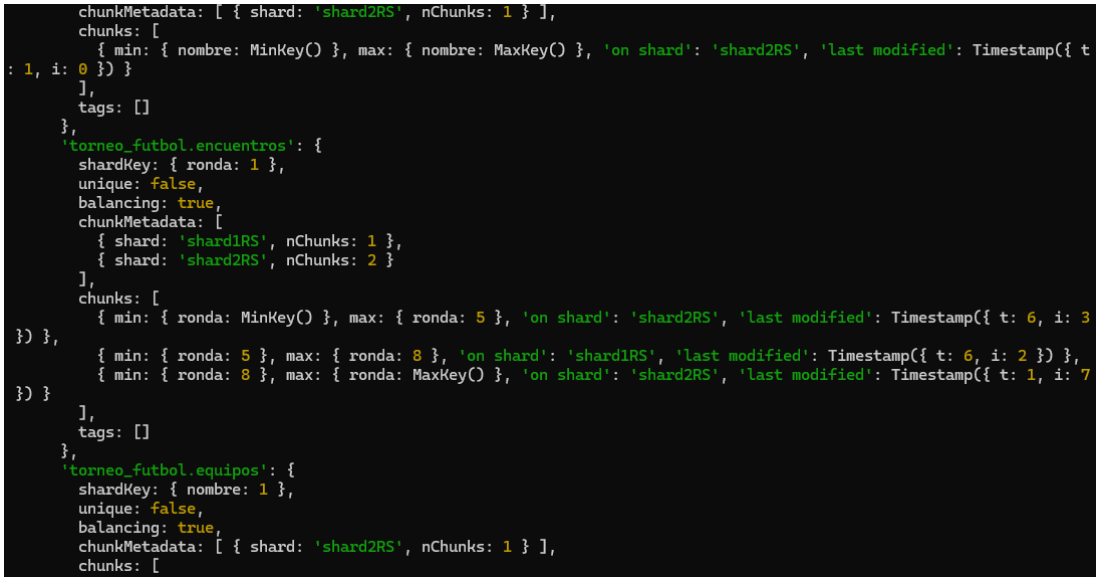


Conectarse al router y añadir el shard al router:

```
[direct: mongos] torneo_futbol> sh.addShard("shard3RS/localhost:27016");
{
  shardAdded: 'shard3RS',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1733963622, i: 18 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1733963622, i: 18 })
}
```

Consultar el estado del clúster y balanceo de datos:

```
[direct: mongos] torneo_futbol> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('6757841ebb3c9020d7922c77') }
---
shards
[
  {
    _id: 'shard1RS',
    host: 'shard1RS/localhost:27018',
    state: 1,
    topologyTime: Timestamp({ t: 1733789930, i: 10 }),
    replSetConfigVersion: Long('1')
  },
  {
    _id: 'shard2RS',
    host: 'shard2RS/localhost:27017',
    state: 1,
    topologyTime: Timestamp({ t: 1733789938, i: 9 }),
    replSetConfigVersion: Long('1')
  },
  {
    _id: 'shard3RS',
    host: 'shard3RS/localhost:27016',
    state: 1,
    topologyTime: Timestamp({ t: 1733963622, i: 9 }),
    replSetConfigVersion: Long('-1')
  }
]
---
active mongoses
```



Resultado Esperado: El nuevo nodo debe recibir datos automáticamente para balancear la carga.

3. Análisis de Resultados

Según los resultados obtenidos de la ejecución de cada caso de prueba planteado, se pone de manifiesto que la elección de las claves de partición no fue la más eficiente y útil

para el particionamiento optimo de los “chunks” entre los nodos de sharding, ya que no se logró cumplir con los resultados en tiempo o balanceo precisamente que se esperaban con la fragmentación hecha, pero se reconoce que con la elección de una mejor clave de partición para el caso particular de las colecciones “jugadores” y “encuentros” como el campo “_id” se hubiese mantenido un balanceo más equitativo y eficiente, ya que esta clave era más única y permitía que se distribuyeran mejor los datos. Los resultados obtenidos entonces se ven así:

- **Tiempo de Respuesta (CP001):** No se cumplió con el tiempo máximo de 100 ms esperado, pero se obtuvo un promedio de 500 ms.
- **Balanceo de Datos (CP002):** No se cumplió inicialmente con la desviación menor al 10% y la única forma de lograrlo con la clave de partición elegida seria mover los “chunks” entre los nodos manualmente y particionar en muchos “chunks” los datos para que se alcanzase una desviación de hasta el 3% como mínimo. En condiciones de balanceo automático entre los nodos, el porcentaje se mantiene entre un 40%.
- **Disponibilidad (CP003):** Se cumplió, y sin interrupciones observadas tras la caída del nodo.
- **Escalabilidad (CP004):** No se cumplió, pero se destaca que para el caso de la asignación de la nueva clave de partición al campo “_id” los datos si fueron redistribuidos en un 20% al nuevo nodo.

  A continuación, se muestra cómo el balanceo se logró entre los 3 shards que habían al final de la ejecución total de pruebas, de forma automática por el balanceador y cuando se probó a usar el campo “_id” como clave de partición:

```
        ],
        tags: []
      },
      'torneo_futbol.encuentros': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [
          { shard: 'shard1RS', nChunks: 1 },
          { shard: 'shard2RS', nChunks: 1 },
          { shard: 'shard3RS', nChunks: 1 }
        ],
        chunks: [
          { min: { _id: MinKey() }, max: { _id: ObjectId('67591113aaa446bd2ac59c73') }, 'on shard': 'shard3RS', 'last mo
dified': Timestamp({ t: 1, i: 0 }) },
          { min: { _id: ObjectId('67591113aaa446bd2ac59c73') }, max: { _id: ObjectId('6759e2f5763b3ac17da5872b') }, 'on
shard': 'shard1RS', 'last modified': Timestamp({ t: 1, i: 1 }) },
          { min: { _id: ObjectId('6759e2f5763b3ac17da5872b') }, max: { _id: MaxKey() }, 'on shard': 'shard2RS', 'last mo
dified': Timestamp({ t: 1, i: 6 }) }
        ],
        tags: []
      },
      'torneo_futbol.equipos': {
        shardKey: { nombre: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shard2RS', nChunks: 1 } ],
        chunks: [
          { min: { nombre: MinKey() }, max: { nombre: MaxKey() }, 'on shard': 'shard2RS', 'last modified': Timestamp({ t
: 1, i: 3 }) }
```

```
        tags: []
      },
      'torneo_futbol.jugadores': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [
          { shard: 'shard1RS', nChunks: 2 },
          { shard: 'shard2RS', nChunks: 1 },
          { shard: 'shard3RS', nChunks: 2 }
        ],
        chunks: [
          { min: { _id: MinKey() }, max: { _id: ObjectId('6759e447763b3ac17da7f95b') }, 'on shard': 'shard3RS', 'last mo
dified': Timestamp({ t: 1, i: 0 }) },
          { min: { _id: ObjectId('6759e447763b3ac17da7f95b') }, max: { _id: ObjectId('6759e447763b3ac17da80dc4') }, 'on
shard': 'shard1RS', 'last modified': Timestamp({ t: 1, i: 1 }) },
          { min: { _id: ObjectId('6759e447763b3ac17da80dc4') }, max: { _id: ObjectId('675a2587c240f634f589418a') }, 'on
shard': 'shard2RS', 'last modified': Timestamp({ t: 1, i: 2 }) },
          { min: { _id: ObjectId('675a2587c240f634f589418a') }, max: { _id: ObjectId('675a2588c240f634f589520f') }, 'on
shard': 'shard3RS', 'last modified': Timestamp({ t: 1, i: 3 }) },
          { min: { _id: ObjectId('675a2588c240f634f589520f') }, max: { _id: MaxKey() }, 'on shard': 'shard1RS', 'last mo
dified': Timestamp({ t: 1, i: 8 }) }
        ],
        tags: []
      },
      'torneo_futbol.tabla_posiciones': {
        shardKey: { equipo: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shard2RS', nChunks: 1 } ],
```

Datos de tamaño de los datos en el cluster al final:

Tamaño total: 335.086.333 bytes

Tamaño de datos por shard:

- shard1RS: 117.376.345 bytes
- shard2RS: 148.073.898 bytes
- shard3RS: 69.637.090 bytes

Ahora el porcentaje de datos asignados en cada shard:

- shard1RS: Porcentaje = (335,086,333117,376,345) ×100 ≈ 35.00%
- shard2RS: Porcentaje = (335,086,333148,073,898) ×100 ≈ 44.19%
- shard3RS: Porcentaje = (335,086,33369,637,090) ×100 ≈ 20.81%