Sistemas de Gestión de Datos y de la Información
Práctica Guiada 2.

Alberto Lorente y Hristo Ivanov

28 de enero de 2016

Resumen

Para la ejecución de estas preguntas se ha utilizado mongo versión 3.2.1 por lo que los resultados son distintos a los de los laboratorios con la versión 3.0.6.

1. Pregunta 1.

¿Cuál es el plan ganador para la misma consulta ordenada por nombre de usuario?
db.users.find('year':1980).sort('username':1)
El plan ganador para la consulta es COLLSCAN - SORT_KEY_GENERATOR - SORT.
Busca en toda la colección aquellos documentos que tengan el año 1980 (COLLSCAN), crea una clave en username para ordenar (SORT_KEY_GENERATOR) y devuelve el resultado ordenador (SORT).

```
"winningPlan" : {
  "stage" : "SORT",
  "sortPattern" : {
    "username" : 1
  },
  "inputStage" : {
    "stage" : "SORT_KEY_GENERATOR",
    "inputStage" : {
      "stage" : "COLLSCAN",
      "filter" : {
        "year" : {
          "$eq" : 1980
        }
      },
      "direction" : "forward"
    }
  }
}
```

2. Pregunta 2.

Considerando el índice year_1, ¿cuáles son las etapas del plan ganador para la consulta de la pregunta anterior?
db.users.find('year':1980).sort('username':1)
El plan ganador para la consulta es IXSCAN - FETCH - SORT_KEY_GENERATOR - KEEP_MUTATIONS - SORT.
Busca por el índice (IXSCAN) y coge aquellos que tengan el año 1980 (FETCH), crea una clave en username para ordenar (SORT_KEY_GENERATOR) y devuelve el resultado ordenador (SORT).

```
"winningPlan" : {
  "stage" : "SORT",
  "sortPattern" : {
    "username" : 1
  },
  "inputStage" : {
    "stage" : "KEEP_MUTATIONS",
    "inputStage" : {
      "stage" : "SORT_KEY_GENERATOR",
      "inputStage" : {
        "stage" : "FETCH",
        "inputStage" : {
          "stage" : "IXSCAN",
          "keyPattern" : {
            "year" : 1
          },
          "indexName" : "year_1",
          "isMultiKey" : false,
          "isUnique" : false,
          "isSparse" : false,
          "isPartial" : false,
          "indexVersion" : 1,
          "direction" : "forward",
          "indexBounds" : {
            "year" : [
              "[1980.0, 1980.0]"
            ]
          }
        }
      }
    }
  }
},
```

3. Pregunta 3.

Considerando el mismo índice year_1, ¿cuáles son las etapas del plan ganador para la siguiente consulta?
db.users.find('like':'deportes').sort('year':1)
¿Cuántos documentos y claves se examinan? ¿Qué rango de claves se recorre?
Las etapas del plan ganador para la consulta son IXSCAN - FETCH.
Escoge por el índice year_1 aquellos documentos que tengan año y extrae los que les gusta el deporte.
En la etapa IXSCAN se examínan 20000 keys del índice year_1 para ordenar y en la etapa FETCH se examínan 20000 documentos para buscar aquellos cuyo atributo 'like' sea 'deportes'.

```
"winningPlan" : {
  "stage" : "FETCH",
  "filter" : {
    "like" : {
      "$eq" : "deportes"
    }
  },
  "inputStage" : {
    "stage" : "IXSCAN",
    "keyPattern" : {
      "year" : 1
    },
    "indexName" : "year_1",
    "isMultiKey" : false,
    "isUnique" : false,
    "isSparse" : false,
    "isPartial" : false,
    "indexVersion" : 1,
    "direction" : "forward",
    "indexBounds" : {
      "year" : [
        "[MinKey, MaxKey]"
      ]
    }
  }
},

"totalKeysExamined" : 20000,
"totalDocsExamined" : 20000,
```

4. Pregunta 4.

Observa la evaluación de la consulta anterior tras crear el índice year_1__id_1 y responde a las siguientes cuestiones:
a) ¿Cuál es el plan ganador? ¿Qué índice usa?
b) ¿Cuántas claves y cuantos documentos consulta el plan ganador?
c) ¿Cuáles son los planes rechazados? ¿Qué índices usan?
a) El plan ganador es IXSCAN - PROJECTION. El índice year_1__id_1
b) El plan ganador examina 164 claves y 0 documentos.
c) El plan rechazado es IXSCAN - FETCH - PROJECTION. Supone la proyección del plan de la pregunta anterior.

```
"winningPlan" : {
  "stage" : "PROJECTION",
  "transformBy" : {
    "_id" : 1
  },
  "inputStage" : {
    "stage" : "IXSCAN",
    "keyPattern" : {
      "year" : 1,
      "_id" : 1
    },
    "indexName" : "year_1__id_1",
    "isMultiKey" : false,
    "isUnique" : false,
    "isSparse" : false,
    "isPartial" : false,
    "indexVersion" : 1,
    "direction" : "forward",
    "indexBounds" : {
      "year" : [
        "[1980.0, 1980.0]"
      ],
      "_id" : [
        "[MinKey, MaxKey]"
      ]
    }
  }
},

"totalKeysExamined" : 164,
"totalDocsExamined" : 0,

"rejectedPlans" : [
  {
```

```
    "stage" : "PROJECTION",
    "transformBy" : {
      "_id" : 1
    },
    "inputStage" : {
      "stage" : "FETCH",
      "inputStage" : {
        "stage" : "IXSCAN",
        "keyPattern" : {
          "year" : 1
        },
        "indexName" : "year_1",
        "isMultiKey" : false,
        "isUnique" : false,
        "isSparse" : false,
        "isPartial" : false,
        "indexVersion" : 1,
        "direction" : "forward",
        "indexBounds" : {
          "year" : [
            "[1980.0, 1980.0]"
          ]
        }
      }
    }
  }
]
},
```

5. Pregunta 5.

¿Qué servidor se ha convertido en primario? ¿Por qué?
El servidor primario es el 27102. Al ser este el primero en formar parte del *Replica Set*. Los demás al ser añadidos después, han tomado a este como primario.

6. Pregunta 6.

¿Los servidores secundarios han sufrido algún cambio? Si es así, ¿a qué es debido ese cambio?
Pista: Revisa rs.status() y comprueba si ha cambiado algo.
Sí. Al perder conexión con el servidor primario estos han realizado una votación para elegir el nuevo servidor primario. En este caso el nuevo servidor primario ha sido el 27101. El `uptime` tiene gran peso en la toma de esta decisión, pero también se toman otros factores en consideración.

7. Pregunta 7.

¿Qué servidores quedan activos y en qué estado están?
En nuestro caso el 27103 sigue vivo, en estado de segundario. El servidor 27103 sigue como segundario porque no sabe si el servidor primario sigue vivo o simplemente no esta alcanzable. En cambio en la configuración anterior teniamos dos servidores segundarios que no eran capaces de alcanzar el primario.

8. Pregunta 8.

¿En cuál de los 3 servidores mongod se ha almacenado la colección sgdi.users? Conecta directamente con ellos utilizando el cliente mongo y compruebas sus colecciones.
En el primero.

9. Pregunta 9.

¿En cuántos chunks se ha dividido la coleccion sgdi.users? ¿Cuántos chunks almacena cada shard? ¿Cuáles son los rangos de cada uno de los chunks creados?
Esta información la podéis encontrar en la base de datos config o mediante el comando sh.status(true) – el parámetro true sirve para activar el modo verboso –.
Chunks total : 14.

- Chunks 27101 : 5
- Chunks 27102 : 5
- Chunks 27103 : 4

Los rangos se pueden apreciar en la siguiente lista:

```
1  { "username" : { "$minKey" : 1 } } -->> { "username" : "DnGYx" } on : shard0001 Timestamp(2, 0)
2  { "username" : "DnGYx" } -->> { "username" : "HZGYA" } on : shard0002 Timestamp(3, 0)
3  { "username" : "HZGYA" } -->> { "username" : "LMwRR" } on : shard0001 Timestamp(4, 0)
4  { "username" : "LMwRR" } -->> { "username" : "PEeab" } on : shard0002 Timestamp(5, 0)
5  { "username" : "PEeab" } -->> { "username" : "Srtfj" } on : shard0001 Timestamp(6, 0)
6  { "username" : "Srtfj" } -->> { "username" : "WdqWk" } on : shard0002 Timestamp(7, 0)
7  { "username" : "WdqWk" } -->> { "username" : "aSneZ" } on : shard0001 Timestamp(8, 0)
8  { "username" : "aSneZ" } -->> { "username" : "eHexD" } on : shard0002 Timestamp(9, 0)
9  { "username" : "eHexD" } -->> { "username" : "hxwlf" } on : shard0001 Timestamp(10, 0)
10 { "username" : "hxwlf" } -->> { "username" : "lnVtN" } on : shard0000 Timestamp(10, 1)
11 { "username" : "lnVtN" } -->> { "username" : "peFcR" } on : shard0000 Timestamp(1, 10)
12 { "username" : "peFcR" } -->> { "username" : "tVaHf" } on : shard0000 Timestamp(1, 11)
13 { "username" : "tVaHf" } -->> { "username" : "xLoBa" } on : shard0000 Timestamp(1, 12)
14 { "username" : "xLoBa" } -->> { "username" : { "$maxKey" : 1 } } on : shard0000 Timestamp(1, 13)
```

10. Pregunta 10.

Para cada una de las siguientes consultas especifica: a) cuántos shards debe consultar mongos, b) cuántos resultados devuelve cada shard, c) qué tipo de búsqueda (escaneo o indexada) se realiza en cada shard, d) número total (contando todos los shards) de documentos devueltos y examinados.
Consulta 1: Usuario con identificador 67
Consulta 2: Usuarios con año 1997
Consulta 3: Usuario de nombre .^aron"

- `e.find('_id':67)`
  - Número de shards: 3
  - Resultados devueltos por shard:
    - Shard 27101 : 1
    - Shard 27102 : 0
    - Shard 27103 : 0
  - Tipo de busqueda por shard:
    - Shard 27101 : IDHACK - SHARDING_FILTER
    - Shard 27102 : IDHACK - SHARDING_FILTER
    - Shard 27103 : IDHACK - SHARDING_FILTER
  - Número total de documentos examinados : 1
- `e.find('year':1997)`
  - Número de shards: 3
  - Resultados devueltos por shard:
    - Shard 27101 : 294
    - Shard 27102 : 310
    - Shard 27103 : 254
  - Tipo de busqueda por shard:
    - Shard 27101 : COLLSCAN - SHARDING_FILTER
    - Shard 27102 : COLLSCAN - SHARDING_FILTER
    - Shard 27103 : COLLSCAN - SHARDING_FILTER
  - Número total de documentos examinados : 100000
- `e.find('username':'Aaron')`
  - Número de shards: 1
  - Resultados devueltos por shard:
    - Shard 27101 : X

    - Shard 27102 : 0
    - Shard 27103 : X
- Tipo de busqueda por shard:
    - Shard 27101 : X
    - Shard 27102 : IXSCAN - SHARDING_FILTER - FETCH
    - Shard 27103 : X
- Número total de documentos examinados : 0