MongoDB Fundamentos

Módulo 3

Explain: planes de ejecución

Permiten saber si se está usando un índice al ejecutar alguna de las siguientes funciones:

- `find`
- `count`
- `remove`
- `update`
- `group`

`Explain` se puede usar en tres modos: *queryPlanner*, *executionStats* y *allPlansExecution.* Se verá cada uno por separado.

1. ***queryPlanner***

Es el modo de ejecución por defecto. Para ver el plan se debe incluir una llamada al método **`explain()`** antes del método que se desea probar. Por ejemplo, utilizando una colección **`coord`** de documentos con coordenadas {x,y}, se puede preguntar:

```
db.coord.explain().find({x:15.23})
{
"queryPlanner" : {
"plannerVersion" : 1,
"namespace" : "banco.coord",
"indexFilterSet" : false,
"parsedQuery" : {
"x" : {
"$eq" : 15.23
}
},
```

```
"winningPlan" : {
 "stage" : "COLLSCAN"
 "filter" : {
 "x" : {
 "$eq" : 15.23
 }
 }
 "direction" : "forward"
 }
 "rejectedPlans" : [ ]
},
"serverInfo" : {
"host" : "puck",
"port" : 27017,
"version" : "3.2.1",
"gitVersion" : "a14d55980c2cdc565d4704a7e3ad37e4e535c1b2"
},
"ok" : 1
}
```

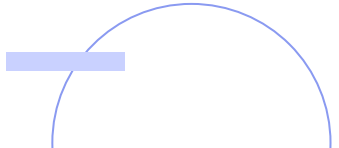
Es posible distinguir **4 partes dentro del plan:**

- **`queryPlanner`:** muestra la query en formato interno (`parsedQuery`), así como el nombre de la base de datos y la colección.
- **`winningPlan`:** es seguramente la parte más importante. Es el plan escogido e indica cómo ha organizado MongoDB la búsqueda de los documentos.
- **`rejectedPlans`:** indica otros planes de ejecución posibles que han sido desechados por ser menos eficientes.
- **`serverInfo`:** lugar donde se ha desarrollado la consulta, y datos sobre la versión de mongo. Finalmente el valor **ok** indica que la operación tuvo éxito.

El **`winningPlan`**, que es el plan elegido, a través de **`stage`** indica que se ha hecho un recorrido de la tabla sin índices (`COLLSCAN`).

Posibles valores de `stage`:

- **`IXSCAN`:** recorrido basado en índices.
- **`FETCH`:** se recorre un resultado anterior, normalmente buscando los valores que cumplen un cierto filtro.
- **`KEEP:MUTATION`:** análogo al anterior.
- **`COLLSCAN`:** se recorre una colección entera.
- **`PROJECTION`:** se hace una proyección de los datos obtenidos en una etapa anterior.
- **`SORT`:** se realiza una ordenación en memoria (muy lenta).
- **`SHARD_MERGED`:** sólo cuando se trabaja en clúster. Indica qué información está al principio en varias particiones. Examina por separado y al final mezcla los resultados.
- **`SHARDING_FILTER`:** filtro de una partición.
- **`SINGLE_SHARD`:** sólo se examinó un miembro de la partición.



Se puede usar **`explain`** como un objeto:

```
explica = db.coord.explain()
```

que es posible usar en cualquier momento:

```
explica.find({x:15.23})
```

y se obtiene la misma información.

2. ***executionStats***

Si se desea más información es necesario usar **`executionStats`**:

```
explica = db.coord.explain("executionStats")
```

y

```
explica.find({x:15.23})
```

ahora devuelve información mucho más detallada, con datos como el número de documento y el tiempo requerido.

```
"executionStats" : {
"executionSuccess" : true,
"nReturned" : 0,
"executionTimeMillis" : 125,
"totalKeysExamined" : 0,
"totalDocsExamined" : 420023,
"executionStages" : {
"stage" : "COLLSCAN",
"filter" : {
"x" : {
"$eq" : 15.23
}
},
"nReturned" : 0,
"executionTimeMillisEstimate" : 120,
```

La consulta ha examinado 420023 documentos en 125 milisegundos.

Veamos cómo cambia si se crea un índice:

```
explica.find({x:15.23})
{
"queryPlanner" : {
"plannerVersion" : 1,
"namespace" : "banco.coord",
"indexFilterSet" : false,
"parsedQuery" : {
"x" : {
"$eq" : 15.23
}
},
"winningPlan" : {
"stage" : "FETCH",
"inputStage" : {
"stage" : "IXSCAN",
"keyPattern" : {
"x" : 1
},
...
```

```
...
"indexName" : "x_1",
"isMultiKey" : false,
"isUnique" : false,
....
"executionStats" : {
"executionSuccess" : true,
"nReturned" : 0,
"executionTimeMillis" : 0,
"totalKeysExamined" : 0,
"totalDocsExamined" : 0,
```

```
"executionStages" : {
"stage" : "FETCH",
"nReturned" : 0,
"executionTimeMillisEstimate" : 0,
"works" : 1,
"advanced" : 0,
"needTime" : 0,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
```

En este caso, se usó el índice y ha tardado 0 ms (lo que significa menos de 1ms).

Es llamativo que ha necesitado revisar 0 documentos, es decir, el propio índice ha permitido llegar a la conclusión de que ningún documento cumple la condición `find`.

3. ***allPlansExecution***

En este modo de consulta, **MongoDB ejecuta el optimizador de consulta para elegir la mejor estrategia para la operación bajo evaluación.**

El método `explain` retorna un `queryPlanner` con información sobre el método evaluado.

Consejos sobre índices

- Cuanto más índices → consultas más rápidas.
- Cuanto más índices → escrituras más lentas.
- En el caso de importaciones masivas es más rápido crear los índices al final
- Índices tan grandes que no quepan en memoria pueden ser de menos utilidad.

Se puede encontrar esta información con el comando **`db.coleccion.stats()`**

o directamente

`db.coleccion.totalIndexSize()`

- Dada una consulta, si se debe elegir:
  - el primer candidato para indexar es alguna clave que contenga una igualdad, si la hay.
  - El segundo es alguna clave incluida en un **`sort`**.
  - Finalmente, las claves que impliquen un rango (mayor o menor).

Se debe tener cuidado al crear índices, hacerlo sólo en el caso de consultas realmente repetidas.

Cuando hay varios índices MongoDB hace una "simulación" de planes usando cada uno de ellos, a ver cuál gana (en realidad es un poco más complicado, hay varios criterios).

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!