Explain in mongodb

ANJU MUNOTH

db.collection.explain()

- To use db.collection.explain(), append one of the methods to db.collection.explain():

  db.collection.explain().<method(...)>

- For example,

  db.products.explain().remove({ category: "apparel" }, { justOne: true })

- For write operations, db.collection.explain() returns information about the write operation that would be performed but does not actually modify the database.

db.collection.explain()

Returns information on the query plan for the following methods:

Starting in MongoDB 3.0	Starting in MongoDB 3.2
•aggregate() •count() •find() •remove() •update()	•distinct() •findAndModify()

db.collection.explain()

- db.collection.explain() method has the following parameter:
- Verbosity: string
- Optional. Specifies the verbosity mode for the explain output.
- The mode affects the behavior of explain() and determines the amount of information to return.
- Possible modes are:
  - "queryPlanner" (Default)
  - "executionStats"
  - " allPlansExecution"

queryPlanner Mode (Default)

- By default, db.collection.explain() runs in queryPlanner verbosity mode.

- MongoDB runs the query optimizer to choose the winning plan for the operation under evaluation.
- db.collection.explain() returns the queryPlanner information for the evaluated method.
- The following example runs db.collection.explain() in "queryPlanner" verbosity mode to return the query planning information for the specified count() operation:

```
db.products.explain().count( { quantity: { $gt: 50 } } )
```

executionStats Mode

- MongoDB runs the query optimizer to choose the winning plan, executes the winning plan to completion, and returns statistics describing the execution of the winning plan.
- Following example runs db.collection.explain() in "executionStats" verbosity mode to return the query planning and execution information for the specified find() operation:

```
db.products.explain("executionStats").find(
   { quantity: { $gt: 50 }, category: "apparel" }
)
```

allPlansExecution Mode

- MongoDB runs the query optimizer to choose the winning plan and executes the winning plan to completion.
- In "allPlansExecution" mode, MongoDB returns statistics describing the execution of the winning plan as well as statistics for the other candidate plans captured during plan selection.
- db.collection.explain() returns the queryPlanner and executionStats information for the evaluated method. The executionStats includes the completed query execution information for the winning plan.
- If the query optimizer considered more than one plan, executionStats information also includes the partial execution information captured during the plan selection phase for both the winning and rejected candidate plans.

allPlansExecution Mode

- The following example runs db.collection.explain() in "allPlansExecution" verbosity mode.
- The db.collection.explain() returns the queryPlanner and executionStats for all considered plans for the specified update() operation:

```
db.products.explain("allPlansExecution").update(
   { quantity: { $lt: 1000}, category: "apparel" },
   { $set: { reorder: true } }
)
```

Explain find() with Modifiers

- db.collection.explain().find() construct allows for the chaining of query modifiers.
- For example, the following operation provides information on the find() method with sort() and hint() query modifiers.

```
db.products.explain("executionStats").find(
   { quantity: { $gt: 50 }, category: "apparel" }
).sort( { quantity: -1 } ).hint( { category: 1, quantity: -1 } )
```

Explain find() with Modifiers

- For a list of query modifiers available, run in the mongo shell:

- **db.collection.explain().find().help()**

Output

- db.collection.explain() operations can return information regarding:
  - queryPlanner, which details the plan selected by the query optimizer and lists the rejected plans;
  - executionStats, which details the execution of the winning plan and the rejected plans; and
  - serverInfo, which provides information on the MongoDB instance.
- The verbosity mode (i.e. queryPlanner, executionStats, allPlansExecution) determines whether the results include executionStats and whether executionStats includes data captured during plan selection.

Explain Output

- explain results present the query plans as a tree of stages.

```
"winningPlan" : {
   "stage" : <STAGE1>,
   ...
   "inputStage" : {
      "stage" : <STAGE2>,
      ...
      "inputStage" : {
         "stage" : <STAGE3>,
         ...
      }
   }
},
```

Explain Output

- Each stage passes its results (i.e. documents or index keys) to the parent node.
- The leaf nodes access the collection or the indices.
- The internal nodes manipulate the documents or the index keys that result from the child nodes.
- The root node is the final stage from which MongoDB derives the result set.
- Stages are descriptive of the operation; e.g.
  - COLLSCAN for a collection scan
  - IXSCAN for scanning index keys
  - FETCH for retrieving documents
  - SHARD_MERGE for merging results from shards
  - SHARDING_FILTER for filtering out orphan documents from shards

queryPlanner information details the plan selected by the query optimizer.

For unsharded collections, explain returns the following queryPlanner information:

```
"queryPlanner" : {
   "plannerVersion" : <int>,
   "namespace" : <string>,
   "indexFilterSet" : <boolean>,
   "parsedQuery" : {
      ...
   },
   "queryHash" : <hexadecimal string>,
   "planCacheKey" : <hexadecimal string>,
   "optimizedPipeline" : <boolean>, // Starting in MongoDB 4.2, only appears if true
   "winningPlan" : {
      "stage" : <STAGE1>,
      ...
      "inputStage" : {
         "stage" : <STAGE2>,
         ...
         "inputStage" : {          ...          }
      }
   },
   "rejectedPlans" : [      <candidate plan 1>,      ...   ]
}
```

queryPlanner- Ouput

- explain.queryPlanner -Contains information on the selection of the query plan by the query optimizer.
- explain.queryPlanner.namespace -A string that specifies the namespace (i.e., <database>.<collection>) against which the query is run.
- explain.queryPlanner.indexFilterSet - A boolean that specifies whether MongoDB applied an index filter for the query shape.
- explain.queryPlanner.queryHash - A hexadecimal string that represents the hash of the query shape and is dependent only on the query shapes. queryHash can help identify slow queries (including the query filter of write operations) with the same query shape

queryPlanner- Ouput

- explain.queryPlanner.planCacheKey - A hash of the key for the plan cache entry associated with the query.
- Unlike the queryHash, the planCacheKey is a function of both the query shape and the currently available indexes for that shape. That is, if indexes that can support the query shape are added/dropped, the planCacheKey value may change whereas the queryHash value would not change.
- explain.queryPlanner.optimizedPipeline -A boolean that indicates that the entire aggregation pipeline operation was optimized away, and instead, fulfilled by a tree of query plan execution stages.
- For example, starting in MongodB 4.2, the following aggregation operation can be fulfilled by the tree of query plan execution rather than using the aggregation pipeline.
- db.example.aggregate([{ $match: { someFlag: true } }])

queryPlanner- Ouput

- explain.queryPlanner.winningPlan -A document that details the plan selected by the query optimizer. MongoDB presents the plan as a tree of stages; i.e. a stage can have an inputStage or, if the stage has multiple child stages, inputStages.
- explain.queryPlanner.winningPlan.stage -A string that denotes the name of the stage.
- Each stage consists of information specific to the stage. For instance, an IXSCAN stage will include the index bounds along with other data specific to the index scan. If a stage has a child stage or multiple child stages, the stage will have an inputStage or inputStages.
- explain.queryPlanner.winningPlan.inputStage -A document that describes the child stage, which provides the documents or index keys to its parent. The field is present if the parent stage has only one child.

queryPlanner- Ouput

- explain.queryPlanner.winningPlan.inputStages -An array of documents describing the child stages. Child stages provide the documents or index keys to the parent stage. The field is present if the parent stage has multiple child nodes. For example, stages for $or expressions or index intersection consume input from multiple sources.

- explain.queryPlanner.rejectedPlans -Array of candidate plans considered and rejected by the query optimizer. The array can be empty if there were no other candidate plans.

executionStats

- The returned executionStats information details the execution of the winning plan. In order to include executionStats in the results, you must run the explain in either:

- executionStats or
- allPlansExecution verbosity mode. Use allPlansExecution mode to include partial execution data captured during plan selection.

```
"executionStats" : {
   "executionSuccess" : <boolean>,
   "nReturned" : <int>,
   "executionTimeMillis" : <int>,
   "totalKeysExamined" : <int>,
   "totalDocsExamined" : <int>,
   "executionStages" : {
      "stage" : <STAGE1>
      "nReturned" : <int>,
      "executionTimeMillisEstimate" : <int>,
      "works" : <int>,
      "advanced" : <int>,
      "needTime" : <int>,
      "needYield" : <int>,
      "saveState" : <int>,
      "restoreState" : <int>,
      "isEOF" : <boolean>,      ...
      "inputStage" : {
         "stage" : <STAGE2>,
         "nReturned" : <int>,
         "executionTimeMillisEstimate" : <int>,         ...
         "inputStage" : {           ...        }
      }
   },
```

Contd on next slide

```
"allPlansExecution" : [
    {
      "nReturned" : <int>,
      "executionTimeMillisEstimate" : <int>,
      "totalKeysExamined" : <int>,
      "totalDocsExamined" :<int>,
      "executionStages" : {
        "stage" : <STAGEA>,
        "nReturned" : <int>,
        "executionTimeMillisEstimate" : <int>,
        ...
        "inputStage" : {
          "stage" : <STAGEB>,
          ...
          "inputStage" : {          ...          }
        }
      }
    },
    ...
  ]
}
```

executionStats

- explain.executionStats - Contains statistics that describe the completed query execution for the winning plan. For write operations, completed query execution refers to the modifications that would be performed, but does not apply the modifications to the database.
- explain.executionStats.nReturned -Number of documents that match the query condition. nReturned corresponds to the n field returned by cursor.explain() in earlier versions of MongoDB.
- explain.executionStats.executionTimeMillis -Total time in milliseconds required for query plan selection and query execution. executionTimeMillis corresponds to the millis field returned by cursor.explain() in earlier versions of MongoDB.
- explain.executionStats.totalKeysExamined - Number of index entries scanned. totalKeysExamined corresponds to the nscanned field returned by cursor.explain() in earlier versions of MongoDB.

executionStats

- explain.executionStats.totalDocsExamined -Number of documents examined during query execution. Common query execution stages that examine documents are COLLSCAN and FETCH.
- totalDocsExamined refers to the total number of documents examined and not to the number of documents returned. For example, a stage can examine a document in order to apply a filter. If the document is filtered out, then it has been examined but will not be returned as part of the query result set.
- If a document is examined multiple times during query execution, totalDocsExamined counts each examination. That is, totalDocsExamined is not a count of the total number of unique documents examined.
- explain.executionStats.executionStages -Details the completed execution of the winning plan as a tree of stages; i.e. a stage can have an inputStage or multiple inputStages.
- Each stage consists of execution information specific to the stage.

executionStats

- explain.executionStats.executionStages.works - Specifies the number of “work units” performed by the query execution stage. Query execution divides its work into small units. A “work unit” might consist of examining a single index key, fetching a single document from the collection, applying a projection to a single document, or doing a piece of internal bookkeeping.

- explain.executionStats.executionStages.advanced -The number of intermediate results returned, or advanced, by this stage to its parent stage.

- explain.executionStats.executionStages.needTime - The number of work cycles that did not advance an intermediate result to its parent stage
- For instance, an index scan stage may spend a work cycle seeking to a new position in the index as opposed to returning an index key; this work cycle would count towards explain.executionStats.executionStages.needTime rather than explain.executionStats.executionStages.advanced.

executionStats

- explain.executionStats.executionStages.needYield - The number of times that the storage layer requested that the query stage suspend processing and yield its locks.
- explain.executionStats.executionStages.saveState - The number of times that the query stage suspended processing and saved its current execution state, for example in preparation for yielding its locks.
- explain.executionStats.executionStages.restoreState - The number of times that the query stage restored a saved execution state, for example after recovering locks that it had previously yielded.

executionStats

- explain.executionStats.executionStages.isEOF - Specifies whether the execution stage has reached end of stream:
- If true or 1, the execution stage has reached end-of-stream.
- If false or 0, the stage may still have results to return.
- For example, consider a query with a limit whose execution stages consists of a LIMIT stage with an input stage of IXSCAN for the query. If the query returns more than the specified limit, the LIMIT stage will report isEOF: 1, but its underlying IXSCAN stage will report isEOF: 0.

executionStats

- explain.executionStats.executionStages.inputStage.keysExamined - For query execution stages that scan an index (e.g. IXSCAN), keysExamined is the total number of in-bounds and out-of-bounds keys that are examined in the process of the index scan.
- If the index scan consists of a single contiguous range of keys, only in-bounds keys need to be examined. If the index bounds consists of several key ranges, the index scan execution process may examine out-of-bounds keys in order to skip from the end of one range to the beginning of the next.
- Consider the following example, where there is an index of field x and the collection contains 100 documents with x values 1 through 100:
- db.keys.find({ x : { $in : [3, 4, 50, 74, 75, 90] } }).explain("executionStats")
- The query will scan keys 3 and 4. It will then scan the key 5, detect that it is out-of-bounds, and skip to the next key 50.
- Continuing this process, the query scans keys 3, 4, 5, 50, 51, 74, 75, 76, 90, and 91. Keys 5, 51, 76, and 91 are out-of-bounds keys that are still examined. The value of keysExamined is 10.

executionStats

- explain.executionStats.executionStages.inputStage.docsExamined - Specifies the number of documents scanned during the query execution stage.
- Present for the COLLSCAN stage, as well as for stages that retrieve documents from the collection (e.g. FETCH)

- explain.executionStats.executionStages.inputStage.seeks - New in version 3.4: For index scan (IXSCAN) stages only.
- The number of times that we had to seek the index cursor to a new position in order to complete the index scan.

- explain.executionStats.allPlansExecution - Contains partial execution information captured during the plan selection phase for both the winning and rejected plans. The field is present only if explain runs in allPlansExecution verbosity mode.

serverInfo

- For unsharded collections, explain returns the following serverInfo information for the MongoDB instance:

```
"serverInfo" : {
   "host" : <string>,
   "port" : <int>,
   "version" : <string>,
   "gitVersion" : <string>
}
```

serverInfo

For sharded collections, explain returns the serverInfo for each accessed shard.

```
"queryPlanner" : {
   ...
   "winningPlan" : {
      "stage" : <STAGE1>,
      "shards" : [
         {
            "shardName" : <string>,
            "connectionString" : <string>,
            "serverInfo" : {
               "host" : <string>,
               "port" : <int>,
               "version" : <string>,
               "gitVersion" : <string>
            },
            ...
         }
         ...
      ]
```

Restrictions

- Starting in MongoDB 4.2, cannot run the explain command/db.collection.explain() in executionStats mode or allPlansExecution mode for an aggregation pipeline that contains the $out stage.
- Instead, you can either:
  - run the explain in queryPlanner mode or
  - run the explain in executionStats mode or allPlansExecution mode but without the $out stage to return information for the stages that precede the $out stage.