$group (aggregation)

Definition

`$group`

Groups documents by some specified expression and outputs to the next stage a document for each distinct grouping. The output documents contain an `_id` field which contains the distinct group by key. The output documents can also contain computed fields that hold the values of some accumulator expression grouped by the `$group`'s `_id` field. `$group` does *not* order its output documents.

The `$group` stage has the following prototype form:

```
{ $group: { _id: <expression>, <field1>: { <accumulator1> : <expression1> }, ... } }
```

The `_id` field is *mandatory*; however, you can specify an `_id` value of null to calculate accumulated values for all the input documents as a whole.

The remaining computed fields are *optional* and computed using the `<accumulator>` operators.

The `_id` and the `<accumulator>` expressions can accept any valid expression. For more information on expressions, see Expressions.

Considerations

Accumulator Operator

The `<accumulator>` operator must be one of the following accumulator operators:

Name	Description
`$sum`	Returns a sum of numerical values. Ignores non-numeric values. *Changed in version 3.2:* Available in both `$group` and `$project` stages.

Name	Description
`$avg`	Returns an average of numerical values. Ignores non-numeric values. *Changed in version 3.2:* Available in both `$group` and `$project` stages.
`$first`	Returns a value from the first document for each group. Order is only defined if the documents are in a defined order. Available in `$group` stage only.
`$last`	Returns a value from the last document for each group. Order is only defined if the documents are in a defined order. Available in `$group` stage only.
`$max`	Returns the highest expression value for each group. *Changed in version 3.2:* Available in both `$group` and `$project` stages.
`$min`	Returns the lowest expression value for each group. *Changed in version 3.2:* Available in both `$group` and `$project` stages.
`$push`	Returns an array of expression values for each group. Available in `$group` stage only.
`$addToSet`	Returns an array of *unique* expression values for each group. Order of the array elements is undefined. Available in `$group` stage only.
`$stdDevPop`	Returns the population standard deviation of the input values. *Changed in version 3.2:* Available in both `$group` and `$project` stages.

Name	Description
`$stdDevSamp`	Returns the sample standard deviation of the input values. *Changed in version 3.2:* Available in both `$group` and `$project` stages.

`$group` Operator and Memory

The `$group` stage has a limit of 100 megabytes of RAM. By default, if the stage exceeds this limit, `$group` will produce an error. However, to allow for the handling of large datasets, set the `allowDiskUse` option to `true` to enable `$group` operations to write to temporary files. See `db.collection.aggregate()` method and the `aggregate` command for details.

Changed in version 2.6: MongoDB introduces a limit of 100 megabytes of RAM for the `$group` stage as well as the `allowDiskUse` option to handle operations for large datasets.

Examples

Calculate Count, Sum, and Average

Given a collection `sales` with the following documents:

```
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-03-01T0
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-03-01T0
{ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-03-15T0
{ "_id" : 4, "item" : "xyz", "price" : 5, "quantity" : 20, "date" : ISODate("2014-04-04T1
{ "_id" : 5, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-04-04T
```

Group by Month, Day, and Year

The following aggregation operation uses the `$group` stage to group the documents by the month, day, and year and calculates the total price and the average quantity as well as counts the documents per each group:

```
db.sales.aggregate(
   [
      {
        $group : {
           _id : { month: { $month: "$date" }, day: { $dayOfMonth: "$date" }, year: { $ye
           totalPrice: { $sum: { $multiply: [ "$price", "$quantity" ] } },
           averageQuantity: { $avg: "$quantity" },
           count: { $sum: 1 }
        }
      }
   ]
)
```

The operation returns the following results:

```
{ "_id" : { "month" : 3, "day" : 15, "year" : 2014 }, "totalPrice" : 50, "averageQuantity
{ "_id" : { "month" : 4, "day" : 4, "year" : 2014 }, "totalPrice" : 200, "averageQuantity
{ "_id" : { "month" : 3, "day" : 1, "year" : 2014 }, "totalPrice" : 40, "averageQuantity"
```

Group by `null`

The following aggregation operation specifies a group `_id` of `null`, calculating the total price and the average quantity as well as counts for all documents in the collection:

```
db.sales.aggregate(
   [
      {
        $group : {
           _id : null,
           totalPrice: { $sum: { $multiply: [ "$price", "$quantity" ] } },
           averageQuantity: { $avg: "$quantity" },
           count: { $sum: 1 }
        }
      }
   ]
)
```

The operation returns the following result:

```
{ "_id" : null, "totalPrice" : 290, "averageQuantity" : 8.6, "count" : 5 }
```

Retrieve Distinct Values

Given a collection `sales` with the following documents:

```
{ "_id" : 1, "item" : "abc", "price" : 10, "quantity" : 2, "date" : ISODate("2014-03-01T0
{ "_id" : 2, "item" : "jkl", "price" : 20, "quantity" : 1, "date" : ISODate("2014-03-01T0
{ "_id" : 3, "item" : "xyz", "price" : 5, "quantity" : 10, "date" : ISODate("2014-03-15T0
{ "_id" : 4, "item" : "xyz", "price" : 5, "quantity" : 20, "date" : ISODate("2014-04-04T1
{ "_id" : 5, "item" : "abc", "price" : 10, "quantity" : 10, "date" : ISODate("2014-04-04T
```

The following aggregation operation uses the `$group` stage to group the documents by the item to retrieve the distinct item values:

```
db.sales.aggregate( [ { $group : { _id : "$item" } } ] )
```

The operation returns the following result:

```
{ "_id" : "xyz" }
{ "_id" : "jkl" }
{ "_id" : "abc" }
```

Pivot Data

A collection `books` contains the following documents:

```
{ "_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 }
{ "_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 }
{ "_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 }
{ "_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 }
{ "_id" : 7020, "title" : "Iliad", "author" : "Homer", "copies" : 10 }
```

Group `title` by `author`

The following aggregation operation pivots the data in the `books` collection to have titles grouped by authors.

```
db.books.aggregate(
   [
     { $group : { _id : "$author", books: { $push: "$title" } } }
   ]
)
```

The operation returns the following documents:

```
{ "_id" : "Homer", "books" : [ "The Odyssey", "Iliad" ] }
{ "_id" : "Dante", "books" : [ "The Banquet", "Divine Comedy", "Eclogues" ] }
```

Group Documents by `author`

The following aggregation operation uses the `$$ROOT` system variable to group the documents by authors. The resulting documents must not exceed the `BSON Document Size` limit.

```
db.books.aggregate(
   [
     { $group : { _id : "$author", books: { $push: "$$ROOT" } } }
   ]
)
```

The operation returns the following documents:

```
{
  "_id" : "Homer",
  "books" :
     [
       { "_id" : 7000, "title" : "The Odyssey", "author" : "Homer", "copies" : 10 },
       { "_id" : 7020, "title" : "Iliad", "author" : "Homer", "copies" : 10 }
     ]
}

{
  "_id" : "Dante",
  "books" :
     [
       { "_id" : 8751, "title" : "The Banquet", "author" : "Dante", "copies" : 2 },
       { "_id" : 8752, "title" : "Divine Comedy", "author" : "Dante", "copies" : 1 },
       { "_id" : 8645, "title" : "Eclogues", "author" : "Dante", "copies" : 2 }
     ]
}
```

SEE ALSO:

The Aggregation with the Zip Code Data Set tutorial provides an extensive example of the `$group` operator in a common use case.

Additional Resources

- MongoDB Analytics: Learn Aggregation by Example: Exploratory Analytics and Visualization Using Flight Data
- MongoDB for Time Series Data: Analyzing Time Series Data Using the Aggregation Framework and Hadoop
- The Aggregation Framework
- Webinar: Exploring the Aggregation Framework
- Quick Reference Cards