$ (projeção) ¶

Nesta página

Definição

$

O $operador posicional limita o conteúdo de um `<array>` dos resultados da consulta para conter apenas o **primeiro** elemento correspondente ao documento de consulta. Para especificar um elemento da matriz a ser atualizado, consulte o operador posicional $ para obter atualizações .

Use $no documento de projeção do `find()` método ou do `findOne()` método quando você precisar apenas de um elemento de matriz específico nos documentos selecionados.

Consulte o operador de agregação `$filter` para retornar uma matriz apenas com os elementos que correspondem à condição especificada.

Considerações de uso

O $operador e o `$elemMatch` operador projetam o **primeiro** elemento correspondente de uma matriz com base em uma condição.

O $operador projeta o primeiro elemento da matriz correspondente de cada documento em uma coleção com base em alguma condição da instrução de consulta.

O `$elemMatch` operador de projeção usa um argumento de condição explícito. Isso permite projetar com base em uma condição que não está na consulta ou se você precisar projetar com base em vários campos nos documentos incorporados da matriz. Consulte Limitações dos campos da matriz para obter um exemplo.

Behavior

Usage Requirements

Given the form:

```
db.collection.find( { <array>: <value> ... },
                    { "<array>.$": 1 } )
db.collection.find( { <array.field>: <value> ...},
                    { "<array>.$": 1 } )
```

The `<array>` field being limited **must** appear in the query document, and the `<value>` can be documents that contain query operator expressions.

Array Field Limitations

MongoDB requires the following when dealing with projection over arrays:

- Only one positional `$` operator may appear in the projection document.
- Only one array field, the one being limited with the `$` projection operator, should appear in the query document. Additional array fields in the query document may lead to undefined behavior.
- The query document should only contain a single condition on the array field being projected. Multiple conditions may override each other internally and lead to undefined behavior.

Under these requirements, the following query is **incorrect**:

```
db.collection.find( { <array>: <value>, <someOtherArray>: <value2> },
                    { "<array>.$": 1 } )
```

To specify criteria on multiple fields of documents inside that array, use the `$elemMatch` query operator. The following query returns the first document inside a `grades` array that has a `mean` of greater than 70 and a `grade` of greater than 90.

```
db.students.find( { grades: { $elemM
                                                  grade: { $gt:90 }
                                                } } },
                  { "grades.$": 1 } )
```

You must use the `$elemMatch` operator if you need separate conditions for selecting documents and for choosing fields within those documents.

Sorts and the Positional Operator

When the `find()` method includes a `sort()`, the `find()` method applies the `sort()` to order the matching documents **before** it applies the positional `$` projection operator.

If an array field contains multiple documents with the same field name and the `find()` method includes a `sort()` on that repeating field, the returned documents may not reflect the sort order because the sort was applied to the elements of the array before the `$` projection operator.

Examples

Project Array Values

A collection `students` contains the following documents:

```
{ "_id" : 1, "semester" : 1, "grades" : [ 70, 87, 90 ] }
{ "_id" : 2, "semester" : 1, "grades" : [ 90, 88, 92 ] }
{ "_id" : 3, "semester" : 1, "grades" : [ 85, 100, 90 ] }
{ "_id" : 4, "semester" : 2, "grades" : [ 79, 85, 80 ] }
{ "_id" : 5, "semester" : 2, "grades" : [ 88, 88, 92 ] }
{ "_id" : 6, "semester" : 2, "grades" : [ 95, 90, 96 ] }
```

In the following query, the projection `{ "grades.$": 1 }` returns only the first element greater than or equal to 85 for the `grades` field.

```
db.students.find( { semester: 1, grades: { $gte: 85 } },
                  { "grades.$": 1 } )
```

The operation returns the following documents:

```
{ "_id" : 1, "grades" : [ 87 ] }
{ "_id" : 2, "grades" : [ 90 ] }
{ "_id" : 3, "grades" : [ 85 ] }
```

Although the array field `grades` may contain multiple elements that are greater than or equal to `85`, the `$` projection operator returns only the first matching element from the array.

Project Array Documents

A `students` collection contains the following documents where the `grades` field is an array of documents; each document contain the three field names `grade`, `mean`, and `std`:

```
{ "_id" : 7, semester: 3, "grades" : [ { grade: 80, mean: 75, std: 8 },
                                       { grade: 85, mean: 90, std: 5 },
                                       { grade: 90, mean: 85, std: 3 } ] }

{ "_id" : 8, semester: 3, "grades" : [ { grade: 92, mean: 88, std: 8 },
                                       { grade: 78, mean: 90, std: 5 },
                                       { grade: 88, mean: 85, std: 3 } ] }
```

In the following query, the projection `{ "grades.$": 1 }` returns only the first element with the `mean` greater than `70` for the `grades` field:

```
db.students.find(
   { "grades.mean": { $gt: 70 } },
   { "grades.$": 1 }
)
```

The operation returns the following documents:

```
{ "_id" : 7, "grades" : [  {  "grade" : 80,  "mean" : 75,  "std" : 8 } ] }
{ "_id" : 8, "grades" : [  {  "grade" : 92,  "mean" : 88,  "std" : 8 } ] }
```

Further Reading

`$elemMatch (projection)`