MongoDB 在查询中利用 $expr 来实现聚合表达

With MongoDB 3.6 the query language gains a new level of expressivity: you can now make use of aggregation expressions in a query using the $expr operator. This feature allows you to take full advantage of all expression operators within all queries, much of which previously had to be done within application logic or was restricted to the aggregation pipeline. $expr offers better performance than the $where operator, which while still available isn't as desirable because it uses the server-side JavaScript engine.

Use aggregation expressions with $expr

The $expr operator can be used in the query portion of all find, update, and delete operations, as well as in the $match stage in aggregations. A valid expression can contain:

- field paths (e.g. "$user.name"),
- system variables (e.g. "$$CURRENT.<field>"),
- literals (e.g. 5),
- expression objects
- expression operators ($gt, $lt, $abs, $avg, $multiply, $in, $cond, etc.)

For example, an application may want to give users an amount of free credits per month and keep track of all expenses for the same account in an array.

```
> db.accounts.insertMany([
  {"_id": 1, "credits": 5000, "expenses": [2000, 2000]},
  {"_id": 2, "credits": 4000, "expenses": [1000, 4000, 2000]},
  {"_id": 3, "credits": 3000, "expenses": [1500, 750]},
  {"_id": 4, "credits": 2000, "expenses": [2500, 750]}
])
```

To find the accounts whose overall expenses have exceeded the credits for that account you can run this query:

```
> db.accounts.find({
    $expr : {
      $gt : [
        { $sum : ["$expenses"] },
        "$credits"
      ]
    }
```

```
 })
{ "_id" : 2, "credits" : 4000, "expenses" : [ 1000, 4000, 2000 ] }
{ "_id" : 4, "credits" : 2000, "expenses" : [ 2500, 750 ] }
```

Use $expr with caution

Prior to v3.6, the $gt operator could be used in aggregation queries to compare fields with given values or with other fields in the same documents. However, comparing fields with other fields was previously not possible in a query. To do this, users had to do the comparisons within their application logic.

Usage of $where came with the caveat that the query would evaluate documents, not index entries. Unfortunately, index entries can't always be used to satisfy $expr filters either, so be conscious of how much of the collection's data will be processed. You can get a sense of the number of processed documents by running MongoDB's .explain() command. Depending on the memory needs of an $expr operation, you may want to consider carefully whether to use a $expr query or to perform equivalent operations within the application. In this expanded example, we limit our query to a specific day's worth of data using the {date: 1} index, rather than operating on the whole collection at once:

```
> db.accountsWithDate.insertMany([
  {"_id": 1, "credits": 5000, "expenses": [2000, 2000], "date": new ISODate("2018-05-01T12:42:10.318Z")},
  {"_id": 2, "credits": 4000, "expenses": [1000, 4000, 2000], "date":  new ISODate("2018-05-01T22:21:50.015Z")},
  {"_id": 3, "credits": 3000, "expenses": [1500, 750], "date": new ISODate("2018-05-02T07:01:20.259Z")},
  {"_id": 4, "credits": 2000, "expenses": [2500, 750], "date": new ISODate("2018-05-03T16:39:05.120Z")}
 ])

> db.accountsWithDate.createIndex({"date": 1})

> db.accountsWithDate.find({
    date: {$gte: ISODate("2018-05-01T00:00:00.000Z"),
          $lt: ISODate("2018-05-02T00:00:00.000Z")},
    $expr : {
       $gt : [
          { $sum : ["$expenses"] },
          "$credits"
       ]
    }
 })
```

{ "_id" : 2, "credits" : 4000, "expenses" : [1000, 4000, 2000], "date" : ISODate("2018-05-01T22:21:50.015Z")