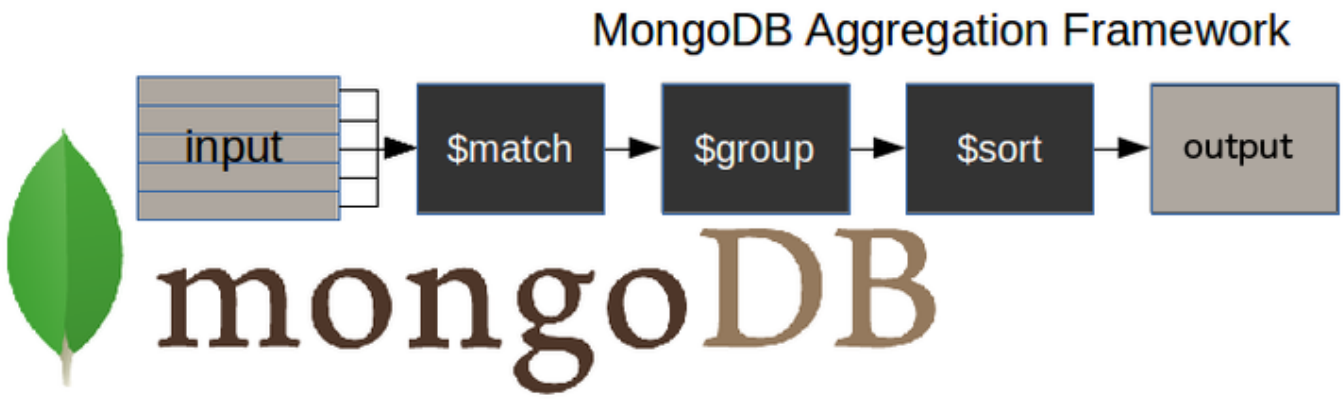
MongoDB 聚合管道如何工作?

下面的图表展示了典型的 MongoDB 聚合管道。



- `$match` 阶段 - 过滤我们需要使用的文档，那些符合我们需求的文档
- `$group` stage – 进行聚合工作
- `$sort` stage – 按照我们需要的方式对结果文档进行排序（升序或降序）

管道的输入可以是单个集合，稍后可以将其他集合合并到管道中。

然后管道对数据执行连续的转换，直到实现我们的目标。

这样，我们可以将复杂的查询分解为更简单的阶段，在每个阶段中我们完成对数据的不同操作。因此，在查询管道结束时，我们将实现我们想要的一切。

这种方法允许我们通过检查查询的输入和输出来检查查询在每个阶段是否正常运行。每个阶段的输出将作为下一个阶段的输入。

> Studio 3T 等工具可让您在构建聚合查询时检查阶段输入和输出。
>
> 下载适用于 Mac、Windows 或 Linux 的免费 Studio 3T！

查询中使用的阶段数量或组合方式没有限制。

为了实现最佳查询性能，需要考虑许多最佳实践。我们将在本文后面讨论这些内容。

MongoDB 聚合管道语法

文章导航

- 其中*collectionName* – 是集合的名称，
- *pipeline* – 是一个包含聚合阶段的数组，
- *options* – 聚合的可选参数

这是聚合管道语法的示例：

```
管道=[
        { $match : { ... } },
        { $group : { ... } },
        { $排序：{ ... } }
       ]
```

MongoDB 聚合阶段限制

聚合在内存中进行。每个阶段最多可使用 100 MB RAM。如果超过此限制，您将从数据库收到错误。

如果它成为一个不可避免的问题，您可以选择分页到磁盘，唯一的缺点是您将等待更长的时间，因为在磁盘上工作比在内存中工作要慢。要选择页面到磁盘方法，您只需将该选项设置 `allowDiskUse` 为 true，如下所示：

```
D b。集合名称.aggregate(管道,{allowDiskUse:true})
```

请注意，此选项并不总是可用的共享服务。例如 Atlas M0、M2 和 M5 集群禁用此选项。

聚合查询返回的文档（无论是作为游标还是通过 `$out` 另一个集合存储）限制为 16MB。也就是说，它们不能大于 MongoDB 文档的最大大小。

如果您可能会超出此限制，那么您应该指定聚合查询的输出将作为游标而不是文档。

我们用于 MongoDB 聚合示例的数据

文章导航

```
{
  国家 ： '西班牙',
  城市 ： '萨拉曼卡',
  名称：“美国航空”,
  地点 ： {
    类型：'点',
    坐标：[-5.6722512,17, 40.9607792]
  },
  学生 ： [
    {年份：2014年，编号：24774},
    {年份：2015年，编号：23166},
    {年份：2016年，编号：21913},
    { 年份：2017 年，编号：21715 }
  ]
}
{
  国家 ： '西班牙',
  城市 ： '萨拉曼卡',
  名称：“UPSA”,
  地点 ： {
    类型：'点',
    坐标：[-5.6691191,17, 40.9631732]
  },
  学生 ： [
    {年份：2014年，数量：4788},
    {年份：2015年，编号：4821},
    { 年份：2016 年，数量：6550 },
    { 年份：2017 年，编号：6125 }
  ]
}
```

如果您想在自己的安装中测试这些示例，可以使用下面的批量命令插入它们，或将其导入为 JSON 文件：

```
使用3tdb
db.universities.insert([
```

文章导航

```
      类型: '点',
      坐标: [-5.6722512,17, 40.9607792]
    },
    学生 :  [
      {年份: 2014年, 编号: 24774},
      {年份: 2015年, 编号: 23166},
      {年份: 2016年, 编号: 21913},
      { 年份: 2017 年, 编号: 21715 }
    ]
  },
  {
    国家 : '西班牙',
    城市 : '萨拉曼卡',
    名称: "UPSA",
    地点 :  {
      类型: '点',
      坐标: [-5.6691191,17, 40.9631732]
    },
    学生 :  [
      {年份: 2014年, 数量: 4788},
      {年份: 2015年, 编号: 4821},
      { 年份: 2016 年, 数量: 6550 },
      { 年份: 2017 年, 编号: 6125 }
    ]
  }
])
```

第二个也是最后一个集合被调用 `'courses'` ，如下所示：

```
{
  大学: 'USAL',
  名称: "计算机科学",
  级别: "优秀"
}
{
  大学: 'USAL',
  名称: '电子',
```

文章导航

```
  名称 : '通讯',
  级别：“优秀”
}
```

同样，您可以使用以下代码或通过导入为 JSON 文件以相同的方式插入它们：

```
db.courses.insert([
{
  大学：'USAL',
  名称：“计算机科学”,
  级别：“优秀”
},
{
  大学：'USAL',
  名称：'电子',
  级别：“中级”
},
{
  大学：'USAL',
  名称 : '通讯',
  级别：“优秀”
}
])
```

> 跳至本文末尾的附件部分，您将在其中找到可供下载的 JSON 文件。

MongoDB 聚合示例

MongoDB $匹配

该 `$match` 阶段允许我们从集合中选择我们想要使用的文档。它通过过滤掉那些不符合我们要求的内容来实现这一点

文章导航

为了获得可读的输出，我将 .pretty() 在所有命令的末尾添加。

```
db.universities.aggregate([
  { $match : { 国家 : '西班牙', 城市 : '萨拉曼卡' } }
]）。漂亮的（）
```

输出是...

```
{
"_id" : ObjectId("5b7d9d9efbc9884f689cdba9"),
“国家”：“西班牙”，“城市”：“萨拉曼卡”，
“名称”：“美国航空”，
“地点” ： {
      “类型”：“点”，
      “坐标”：[
            -5.6722512，
             17、
             40.9607792
       ]
},
“学生” ： [
                  {
                        “年份”：2014年，
                        “号码”：24774
                  },
                  {
                        “年份”：2015年，
                        “号码”：23166
                  },
                  {
                        “年份”：2016年，
                        “号码”：21913
                  },
                  {
                        “年份”：2017 年，
                        “号码”：21715
```

文章导航

```
            "国家": "西班牙",
            "城市": "萨拉曼卡",
            "名称": "UPSA",
            "地点" :  {
                    "类型": "点",
                    "坐标": [
                            -5.6691191,
                            17、
                            40.9631732
                    ]
            },
            "学生" :  [
                    {
                            "年份": 2014年,
                            "号码": 4788
                    },
                    {
                            "年份": 2015年,
                            "号码": 4821
                    },
                    {
                            "年份": 2016年,
                            "数量": 6550
                    },
                    {
                            "年份": 2017 年,
                            "号码": 6125
                    }
            ]
    }
```

MongoDB$项目

您很少需要检索文档中的所有字段。最好只返回您需要的字段，以避免处理不必要的数据。

该 `$project` 阶段用于执行此操作并添加您需要的任何计算字段。

在此示例中，我们只需要字段 country、city 和 name

文章导航

- 除了 _id 字段之外，只指定我们需要作为查询结果获取的那些字段就足够了

这个阶段……

```
db.universities.aggregate([
  { $project : { _id : 0, 国家 : 1 , 城市 : 1 , 名称 : 1 } }
]) 。漂亮的（）
```

..将给出结果...

```
{“国家”：“西班牙”，“城市”：“萨拉曼卡”，“名称”：“美国”}
{“国家”：“西班牙”，“城市”：“萨拉曼卡”，“名称”：“UPSA”}
```

> 这是MongoDB $project 的另一个示例。

MongoDB$组

通过该 $group 阶段，我们可以执行所需的所有聚合或汇总查询，例如查找计数、总计、平均值或最大值。

在此示例中，我们想知道“ ”集合中每所大学的文档数量 universities ：

查询...

```
db.universities.aggregate([
  { $group : { _id : '$name',totaldocs : { $sum : 1 } } }
]) 。漂亮的（）
```

..将产生这个结果...

```
{“ id”：“UPSA”，“总文档”：1 }
```

文章导航

操作员	意义
$计数	计算给定组中的文档数量。
最高$	显示集合中文档字段的最大值。
分钟$	显示集合中文档字段的最小值。
平均$	显示集合中文档字段的平均值。
$总和	汇总集合中所有文档的指定值。
$推	将额外的值添加到结果文档的数组中。

查看其他MongoDB 运算符 并了解有关此主题的更多信息。

MongoDB $out

这是一种不寻常的阶段类型，因为它允许您将聚合结果转移到新集合中，或者在删除集合后转移到现有集合中，甚至将它们添加到现有文档中（4.1.2 版本中的新功能）。

该 `$out` 阶段必须是管道中的最后阶段。

我们第一次使用具有多个阶段的聚合。我们现在有两个， a `$group` 和 an `$out` ：

```
db.universities.aggregate([
  { $group : { _id : '$name',totaldocs : { $sum : 1 } } },
  { $out : 'aggResults' }
])
```

现在，我们检查新的“ ”集合的内容 `aggResults` ：

```
db.aggResults.find().pretty()
{“_id”: “UPSA”, “总文档”: 1 }
{ "_id" : "USAL", "totaldocs" : 1 }
```

文章导航

现在我们已经生成了多阶段聚合，我们可以继续构建管道。

MongoDB $unwind

MongoDB 中的阶段 `$unwind` 通常出现在管道中，因为它是达到目的的一种手段。

您无法直接处理具有诸如 之类的阶段的文档中的数组元素 `$group` 。该 `$unwind` 阶段使我们能够处理数组中字段的值。

如果输入文档中存在数组字段，则有时需要多次输出文档，针对该数组的每个元素一次。

文档的每个副本都将数组字段替换为连续元素。

在下一个示例中，我将仅将阶段应用于字段 `name` 包含值 的文档 `USAL` 。

这是文件：

```
{
  国家 ： '西班牙',
  城市 ： '萨拉曼卡',
  名称：“美国航空”,
  地点 ：  {
    类型：'点',
    坐标：[-5.6722512,17, 40.9607792]
  },
  学生 ：  [
    {年份：2014年，编号：24774},
    {年份：2015年，编号：23166},
    {年份：2016年，编号：21913},
    { 年份：2017 年，编号：21715 }
  ]
}
```

现在，我们将 `$unwind` 舞台应用到学生的数组上，并检查是否为数组的每个元素获取了一个文档。

第一个文档由数组第一个元素中的字段和其余公共字段组成。

文章导航

```
    { $unwind : '$students' }
]）。漂亮的（）
{
        "_id" : ObjectId("5b7d9d9efbc9884f689cdba9"),
        “国家”：“西班牙”，
        “城市”：“萨拉曼卡”，
        “名称”：“美国航空”，
        “地点” ： {
                “类型”：“点”，
                “坐标”：[
                        -5.6722512，
                        17、
                        40.9607792
                ]
        }，
        “学生” ： {
                “年份”：2014年，
                “号码”：24774
        }
}
{
        "_id" : ObjectId("5b7d9d9efbc9884f689cdba9"),
        “国家”：“西班牙”，
        “城市”：“萨拉曼卡”，
        “名称”：“美国航空”，
        “地点” ： {
                “类型”：“点”，
                “坐标”：[
                        -5.6722512，
                        17、
                        40.9607792
                ]
        }，
        “学生” ： {
                “年份”：2015年，
                “号码”：23166
        }
}
{
```

文章导航

```
            "地点" :  {
                    "类型": "点",
                    "坐标": [
                            -5.6722512,
                            17、
                            40.9607792
                    ]
            },
            "学生" :  {
                    "年份": 2016年,
                    "号码": 21913
            }
    }
    {
            "_id" : ObjectId("5b7d9d9efbc9884f689cdba9"),
            "国家": "西班牙",
            "城市": "萨拉曼卡",
            "名称": "美国航空",
            "地点" :  {
                    "类型": "点",
                    "坐标": [
                            -5.6722512,
                            17、
                            40.9607792
                    ]
            },
            "学生" :  {
                    "年份": 2017 年,
                    "号码": 21715
            }
    }
```

MongoDB $排序

您需要 `$sort` 阶段根据特定字段的值对结果进行排序。

例如，我们将阶段结果获得的文档 `$unwind` 按学生人数降序排序。

文章导航

```
    { $unwind : '$students' },
    { $project : { _id : 0, '学生年份' : 1, '学生人数' : 1 } },
    { $sort : { '学生人数' : -1 } }
]) 。漂亮的（）
```

这给出了结果……

```
{ "学生" : { "年份" : 2014, "人数" : 24774 } }
{ "学生" : { "年份" : 2015, "人数" : 23166 } }
{ "学生" : { "年份" : 2016, "人数" : 21913 } }
{ "学生" : { "年份" : 2017, "人数" : 21715 } }
```

> 该 `$sort` 阶段可以与其他阶段一起使用，以将MongoDB 集合中的数据减少到恰好满足您的需要。

MongoDB 美元限制

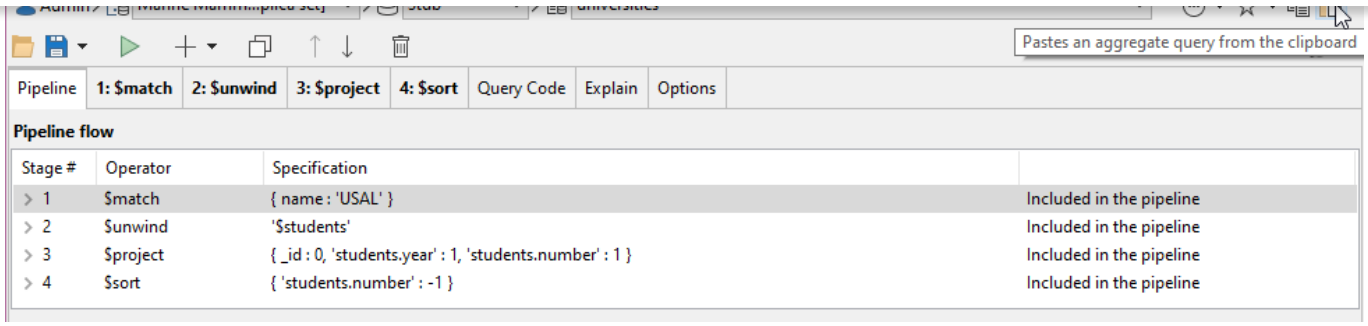
如果您只对查询的前两个结果感兴趣怎么办？它很简单：

```
db.universities.aggregate([
    { $match : { name : 'USAL' } },
    { $unwind : '$students' },
    { $project : { _id : 0, '学生年份' : 1, '学生人数' : 1 } },
    { $sort : { '学生人数' : -1 } },
    { $限制：2 }
]) 。漂亮的（）
{ "学生" : { "年份" : 2014, "人数" : 24774 } }
{ "学生" : { "年份" : 2015, "人数" : 23166 } }
```

请注意，当您需要限制排序文档的数量时，**必须** `$limit` 使用. `$sort`

现在我们已经有了完整的管道。

文章导航

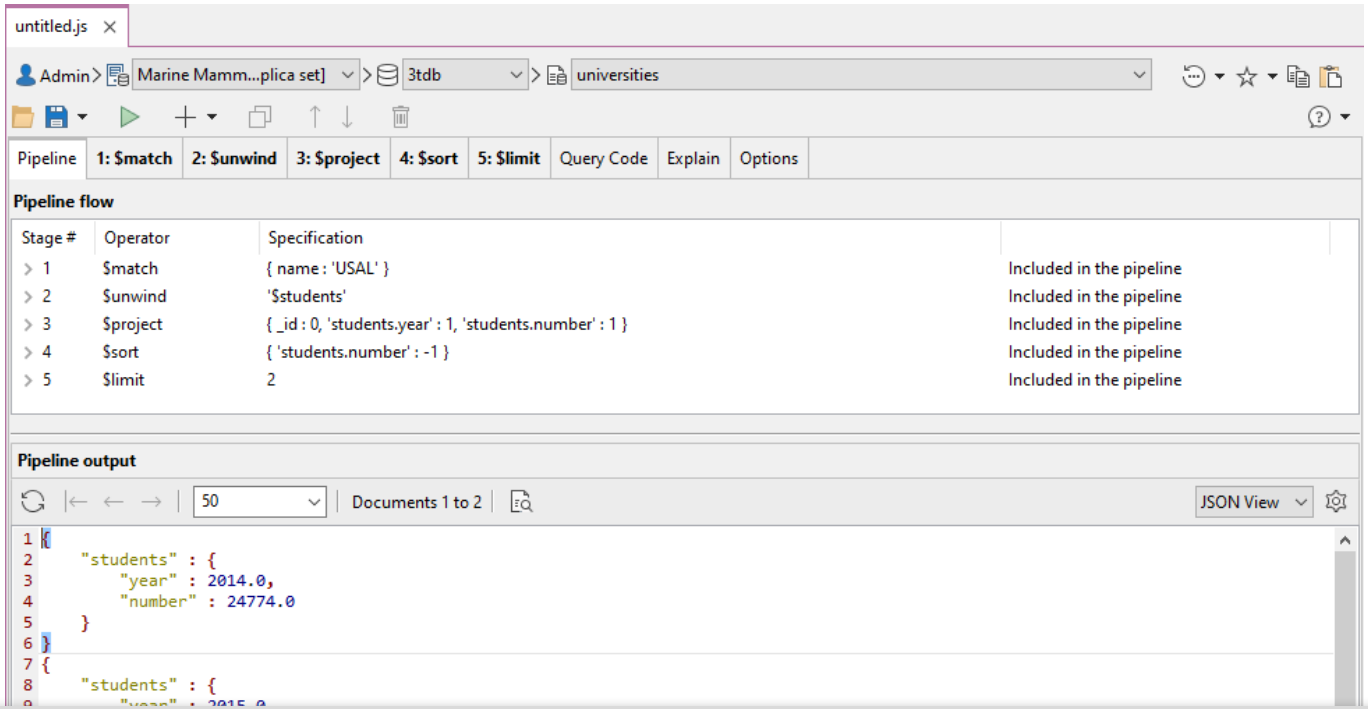


阅读有关聚合编辑器（Studio 3T 的逐步 MongoDB 聚合查询生成器）的更多信息。

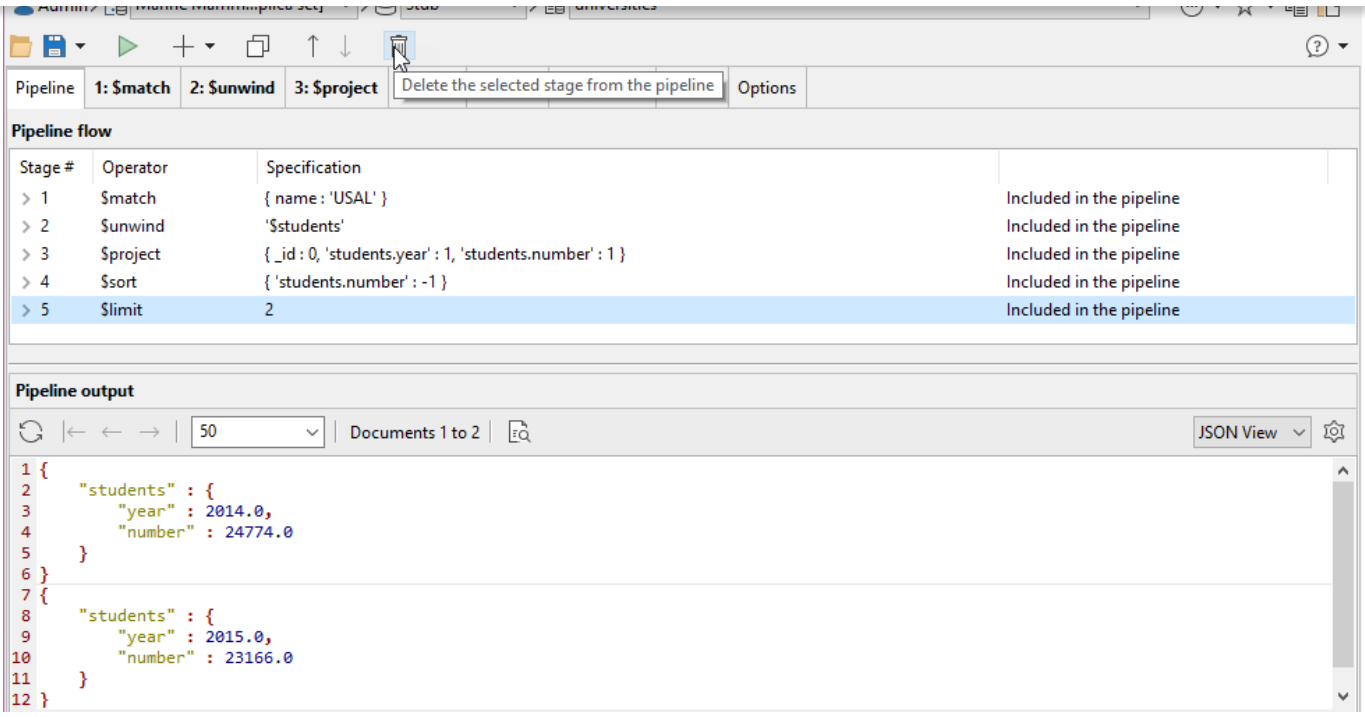
只需复制并粘贴如下所示的部分即可

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $unwind : '$students' },
  { $project : { _id : 0, '学生年份' : 1, '学生人数' : 1 } },
  { $sort : { '学生人数' : -1 } }
])
```

在下一个屏幕截图中，我们可以看到 Studio 3T 中的完整流程及其输出。



文章导航



> Studio 3T 的聚合编辑器支持这些 MongoDB 聚合运算符和阶段。

$addFields

您可能需要以新字段的方式对输出进行一些更改。在下一个示例中，我们要添加大学的成立年份。

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $addFields : { Foundation_year : 1218 } }
]) 。漂亮的（）
```

这给出了结果.....

```
{
        "_id" : ObjectId("5b7d9d9efbc9884f689cdba9"),
        “国家”：“西班牙”，
        “城市”：“萨拉曼卡”，
```

文章导航

```
                        17,
                        40.9607792
                ]
        },
        “学生” :  [
                {
                        “年份”: 2014年,
                        “号码”: 24774
                },
                {
                        “年份”: 2015年,
                        “号码”: 23166
                },
                {
                        “年份”: 2016年,
                        “号码”: 21913
                },
                {
                        “年份”: 2017 年,
                        “号码”: 21715
                }
        ],
        “基础年”: 1218
}
```

MongoDB$计数

该 `$count` 阶段提供了一种简单的方法来检查管道前一阶段的输出中获得的文档数量。

让我们看看它的实际效果：

```
db.universities.aggregate([
  { $unwind : '$students' },
  { $count : 'total_documents' }
]）。漂亮的（）
```

MongoDB $lookup

因为 MongoDB 是基于文档的，所以我们可以按照我们需要的方式塑造文档。然而，通常需要使用来自多个集合的信息。

使用 `$lookup` ，这是一个聚合查询，它合并两个集合中的字段。

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $项目：{ _id：0，名称：1 } },
  { $查找：{
      来自：'课程',
      localField : '名称',
      外国领域：'大学',
      如：“课程”
  } }
]）。漂亮的（）
```

> 需要另一个 `$lookup` 例子吗？这是一个。

如果您希望此查询快速运行，则需要对集合 `name` 中的字段 `universities` 和集合 `university` 中的字段建立索引 `courses` 。

换句话说，不要忘记对 `$lookup` .

```
{
        “名称”：“美国航空”，
        “培训班”：  [
                {
                        "_id" : ObjectId("5b7d9ea5fbc9884f689cdbab"),
                        “大学”：“USAL”，
                        "name" : "计算机科学",
                        “级别”：“优秀”
```

文章导航

```
                    "级别": "中级"
                },
                {
                        "_id" : ObjectId("5b7d9ea5fbc9884f689cdbad"),
                        "大学": "USAL",
                        "名称": "通讯",
                        "级别": "优秀"
                }
        ]
}
```

> 这是在 MongoDB 中创建索引的最快方法。

MongoDB $sortByCount

此阶段是对字段中不同值的数量进行分组、计数然后按降序排序的快捷方式。

假设您想知道每个级别的课程数量（按降序排列）。以下是您需要构建的查询：

```
db.courses.aggregate([
  { $sortByCount : '$level' }
]) 。漂亮的（）
```

这是输出：

```
{ "_id" : "优秀", "计数" : 2 }
{ "_id" : "中级", "计数" : 1 }
```

MongoDB $facet

有时，在创建数据报表时，您会发现需要对多个报表进行相同的初步处理，并且面临必须创建和维护中间集合的情况。

文章导航

多亏了这个阶段，我们现在可以在单个管道中完成它 $facet 。

看一下这个例子：

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $查找：{
      来自：'课程',
      localField : '名称',
      外国领域：'大学',
      如：“课程”
  } },
  { $facet : {
      '计数级别':
      [
         { $unwind : '$courses' },
         { $sortByCount : '$courses.level' }
      ],
      “学生较少的一年”：
      [
         { $unwind : '$students' },
         { $project : { _id : 0, 学生 : 1 } },
         { $sort : { '学生人数' : 1 } },
         { $限制：1 }
      ]
  } }
]）。漂亮的（）
```

我们所做的是从我们的大学课程数据库创建两份报告。**计算水平**和**学生较少的年份。**

他们都使用前两个阶段的输出，即 $match 和 $lookup 。

对于大型集合，这可以通过避免重复来节省大量处理时间，并且我们不再需要编写中间临时集合。

```
{
        “计数级别”：[
```

```
                {
                        "_id" : "中级",
                        "计数": 1
                }
        ],
        "年少学生": [
                {
                        "学生" :  {
                                "年份": 2017 年,
                                "号码": 21715
                        }
                }
        ]
}
```

> 详细了解 MongoDB 聚合中 $facet 阶段的其他用例。

锻炼

现在，尝试自己解决下一个练习。

我们如何获得曾经属于每一所大学的学生总数?

```
db.universities.aggregate([
  { $unwind : '$students' },
  { $group : { _id : '$name', 总校友 : { $sum : '$students.number' } }
}
]）。漂亮的（）
```

输出：

```
{"_id": "UPSA", "总校友": 22284 }
{"_id": "USAL", "总校友": 91568 }
```

文章导航

```
db.universities.aggregate([
  { $unwind : '$students' },
  { $group : { _id : '$name', 总校友 : { $sum : '$students.number' } } }
},
  { $sort : { 校友总数 : -1 } }
]) 。漂亮的（）
```

是的，我们需要 `$sort()` 在 的输出处应用阶段 `$group()` 。

检查我们的聚合查询

我之前提到过，检查查询的各个阶段是否按照我们需要的方式执行非常简单，而且确实很重要。

使用 Studio 3T，您可以使用两个专用面板来检查任何特定阶段的输入和输出文档。

表现

聚合管道会自动重塑查询，以提高其性能。

如果您同时拥有 `$sort` 和 `$match` 阶段，则最好在 the `$match` 之前使用 `$sort` ，以最大程度地减少阶段 `$sort` 必须处理的文档数量。

要利用索引，您必须在管道的第一阶段执行此操作。在这里，您必须使用 `$match` 或 `$sort` 阶段。

我们可以通过该方法检查查询是否使用了索引 `explain()` 。

```
管道 = [...]
db.<collectionName>.aggregate( 管道, { 解释 : true })
```

> 您始终可以 `explain()` 通过单击“解释”选项卡以图表或 JSON 形式查看任何聚合查询