

# ELASTICSEARCH NESTED QUERIES: HOW TO SEARCH FOR EMBEDDED DOCUMENTS



ElasticSearch is annoyingly complicated at times. You can run a search and it runs the wrong results and you are not made aware of that.

This can happen when, for example, you have a nested JSON document, i.e., one JSON document inside another. This is because Lucene (i.e., ElasticSearch) query has no understanding of object hierarchy in a JSON document.

## The Problem with Searching for nested JSON objects

To illustrate the problem and the solution, download [this program massAdd.py](#) and change the URL to match your ElasticSearch environment. Then run it.

Then look at loaded data. You can see from the brackets that **classes** is a JSON array. But the index, as we will see, does not reflect that.

```
curl -XGET --header 'Content-Type: application/json'
http://parisx:9200/universities/_search?pretty}
```

```
"_index" : "universities",
  "_type" : "universities",
  "_id" : "428b71b0b935eaf58bdf874c819f263a919da5f3",
  "_score" : 0.0,
  "_source" : {
```

```

        "firstName" : "Stephen",
        "classes" : ,
        "lastName" : "Shakespeare",
        "school" : "Arizona State University"
    }
}
]
}

```

Notice the index mapping does not show the JSON array. This is not because there are no brackets []. Rather the **nested** word is missing.

```

curl -XGET --header 'Content-Type: application/json'
http://parisx:9200/universities?pretty

```

...

```

"classes" : {
  "properties" : {
    "grades" : {
      "type" : "long"
    },
    "name" : {
      "type" : "text",
      "fields" : {
        "keyword" : {
          "type" : "keyword",
          "ignore_above" : 256
        }
      }
    }
  }
},

```

...

So run a search using what you would think would be the logical way to find students taking physics who got a grade of 1:

```

curl -XGET --header 'Content-Type: application/json'
http://parisx:9200/universities/_search/?pretty=true -d '{
  "query": {
    "bool" : {
      "must" :
    }
  }
}'

```

Produces this incorrect result. This is because Lucene flattens the whole document and finds both a **classes.grades** 1 and classes.name **physics**.

```
{
  "_index" : "universities",
  "_type" : "universities",
  "_id" : "5846c01f90d80448ab087bf4f476cd3f8ab6f683",
  "_score" : 1.621972,
  "_source" : {
    "classes" : ,
    "firstName" : "Stephen",
    "lastName" : "Rowe",
    "school" : "University of South Carolina - Columbia"
  }
}
```

## Making the Index Explicit

You can create an index in Elasticsearch just by loading data. That creates it on-the-fly. But you cannot control the way the index is created if you do that. So let's make the index creation explicit and make clear that classes is a JSON array.

First delete the index, which will also delete all the data.

```
curl -XDELETE http://parisx:9200/universities
```

Then create a new index and make it explicit that **classes** is an object inside of the universities object by using **"type" : "nested"**. Note: that we are not filling in university data, just students. So ignore those fields. (We will use university data in another post and [used it in the previous post](#)).

```
curl -XPUT --header 'Content-Type: application/json'
http://parisx:9200/universities -d '{
  "mappings" : {
    "universities" : {
      "properties" : {
        "Address" : { "type" : "text"},
        "AdminEmail" : { "type" : "text"},
        "AdminName" : { "type" : "text" },
        "AdminPhone" : { "type" : "text" },
        "DapipId" : { "type" : "text" },
        "Fax" : { "type" : "text" },
        "GeneralPhone" : { "type" : "text" },
        "LocationName" : { "type" : "text" },
        "LocationType" : { "type" : "text" },
        "OpeId" : { "type" : "text" },
        "ParentDapipId" : { "type" : "text" },
        "ParentName" : { "type" : "text" },
        "UpdateDate" : { "type" : "text" },
        "classes" : {
          "type" : "nested",
          "properties" : {
            "name" : { "type" : "text"},
```

```

        "grades" : { "type" : "integer" }
    },
    "firstName" : { "type" : "text" },
    "lastName" : { "type" : "text" },
    "school" : { "type" : "text" }
}
}
}'

```

Then load the data again by running massAdd.py again:

Now add the terms **nested** and **path** to the query to find students who got a grade of 1 in physics. The **path** is the parent object in the JSON, which in this case is **classes**. And use classes.name to search by class name:

```

curl -XGET --header 'Content-Type: application/json'
http://parisx:9200/universities/_search/?pretty=true -d '{
  "query": {
    "nested" : {
      "path" : "classes",
      "score_mode" : "avg",
      "query" : {
        "bool" : {
          "must" :
        }
      }
    }
  }
}'

```

results in the correct result.

```

{
  "_index" : "universities",
  "_type" : "universities",
  "_id" : "20ce682e16d0b88487addc9ceba2a695d57692aa",
  "_score" : 2.188588,
  "_source" : {
    "lastName" : "Shakespeare",
    "firstName" : "Julie",
    "classes" : ,
    "school" : "Arizona State University"
  }
}
]

```