# Querying nested objects

codingexplained.com

youtube.com/@codingexplained

CodingExplained

CODE. SLEEP. REPEAT.

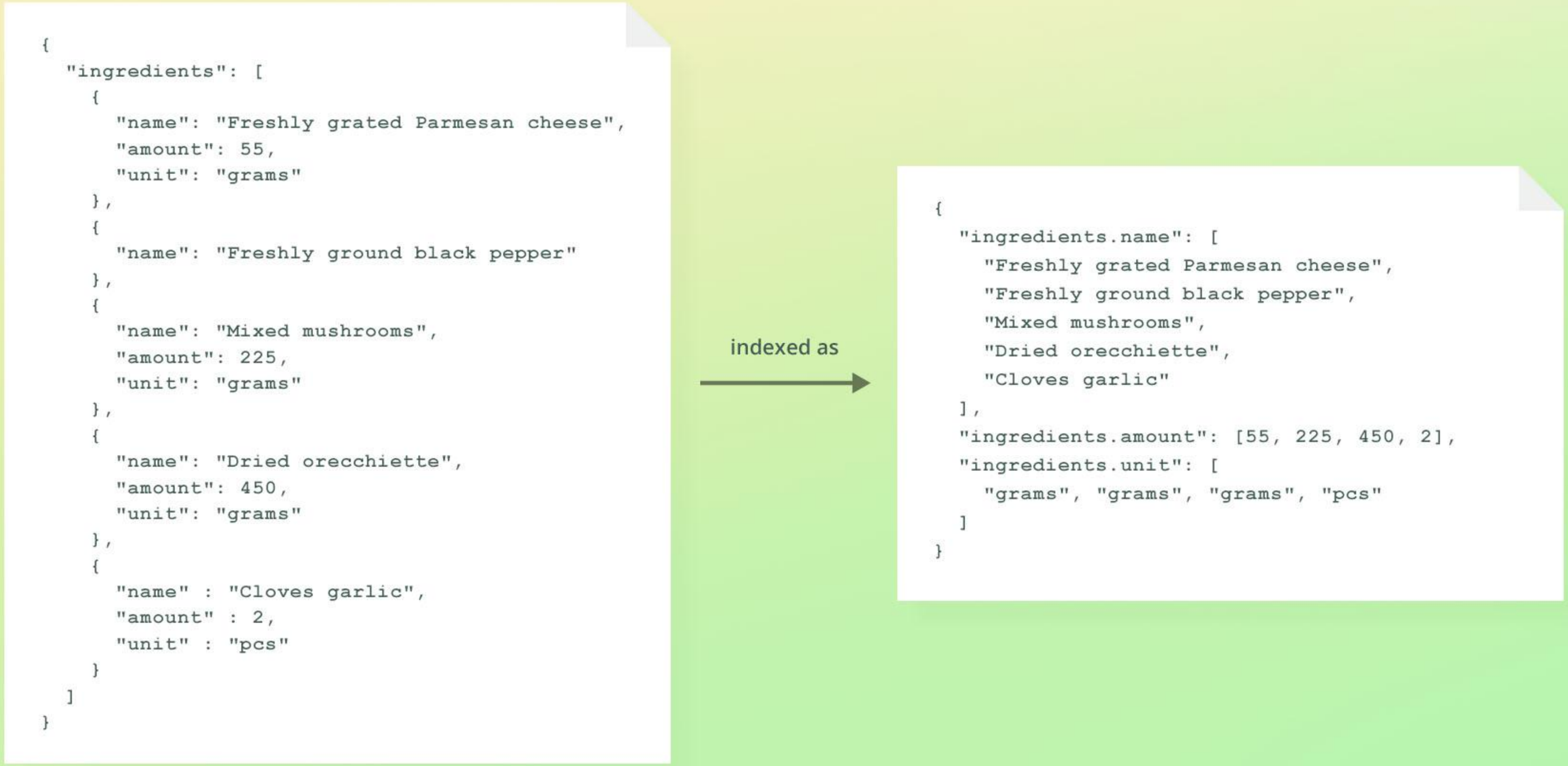
## The `nested` data type

- We covered the `nested` data type earlier in the course
- This lecture recaps on the concept of nested objects
  - Feel free to revisit the *“Overview of data types”* lecture

```json
{
  "title": "Pasta With Mushrooms, Brussels Sprouts, and Parmesan",
  "description": "Description of the recipe.",
  "preparation_time_minutes": 25,
  "servings": {
    "min": 4,
    "max": 4
  },
  "steps": [
    "Boil the pasta.",
    "Cut the mushrooms.",
    "Do some magic."
  ],
  "ingredients": [
    {
      "name": "Freshly grated Parmesan cheese",
      "amount": 55,
      "unit": "grams"
    },
    {
      "name": "Freshly ground black pepper"
    },
    {
      "name": "Mixed mushrooms",
      "amount": 225,
      "unit": "grams"
    },
    {
      "name": "Dried orecchiette",
      "amount": 450,
      "unit": "grams"
    },
    {
      "name" : "Cloves garlic",
      "amount" : 2,
      "unit" : "pcs"
    }
  ],
  "created": "2002-10-21T15:07:53Z",
  "ratings": [4.0, 3.5]
}
```

## Units

- `"grams"`
- `"cups"`
- `"handful"`
- `"cans"`
- `"pcs"` **(pieces)**
- `"ml"` **(milliliter)**
- `"tsp"` **(teaspoon)**
- `"tbsp"` **(tablespoon)**



```
{
  "ingredients": [
    {
      "name": "Freshly grated Parmesan cheese",
      "amount": 55,
      "unit": "grams"
    },
    {
      "name": "Freshly ground black pepper"
    },
    {
      "name": "Mixed mushrooms",
      "amount": 225,
      "unit": "grams"
    },
    {
      "name": "Dried orecchiette",
      "amount": 450,
      "unit": "grams"
    },
    {
      "name" : "Cloves garlic",
      "amount" : 2,
      "unit" : "pcs"
    }
  ]
}
```

```
{
  "ingredients.name": [
    "Freshly grated Parmesan cheese",
    "Freshly ground black pepper",
    "Mixed mushrooms",
    "Dried orecchiette",
    "Cloves garlic"
  ],
  "ingredients.amount": [55, 225, 450, 2],
  "ingredients.unit": [
    "grams", "grams", "grams", "pcs"
  ]
}
```

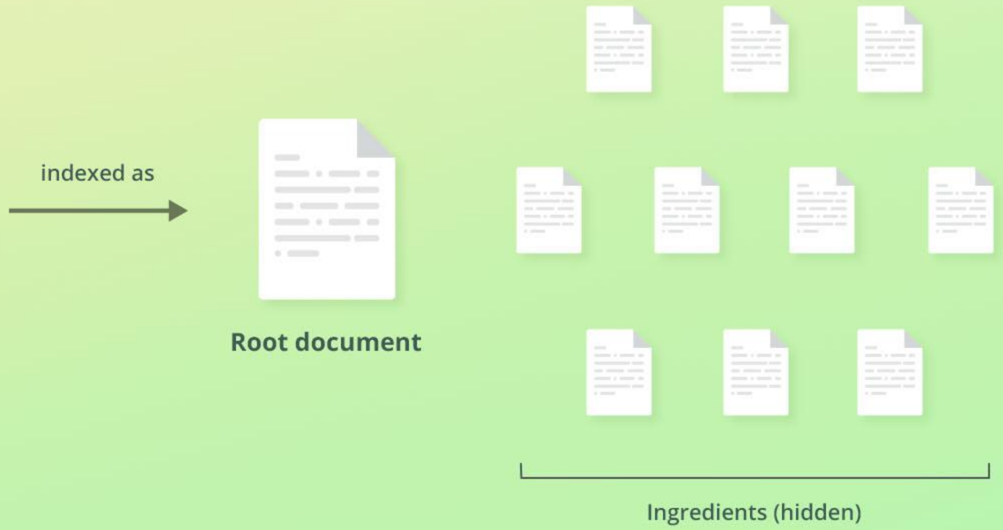
```
GET /recipes/_search
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "ingredients.name": "parmesan"
          }
        },
        {
          "range": {
            "ingredients.amount": {
              "gte": 100
            }
          }
        }
      ]
    }
  }
}
```

```
{
  "ingredients.name": [
    "Freshly grated Parmesan cheese",
    "Freshly ground black pepper",
    "Mixed mushrooms",
    "Dried orecchiette",
    "Cloves garlic"
  ],
  "ingredients.amount": [55, 225, 450, 2],
  "ingredients.unit": ["grams", "grams", "grams", "pcs"]
}
```

CodingExplained
CODE. SLEEP. REPEAT.

# How `nested` fields are indexed

```
POST /recipes/_doc
{
  "title": "Pasta With Mushrooms",
  "description": "Very good!",
  "ingredients": [
    # 10 ingredients
  ]
}
```



CodingExplained
CODE. SLEEP. REPEAT.

## The problem & the solution

**Problem:**

- When indexing arrays of objects, the relationships between values are not maintained
- Queries can yield “unpredictable” results

**Solution:**

- Use the `nested` data type and the `nested` query
- Create a new index to update the field mapping & reindex documents

## How documents are scored

- Matching child objects affect the parent document’s relevance score
- Elasticsearch calculates a relevance score for each matching child object
    - This is because each nested object is a Lucene document
- Relevance scoring can be configured with the `score_mode` parameter

## Adjusting relevance scores with `score_mode`

| `score_mode` | **Parent document's relevance score** |
|---|---|
| `avg` (default) | The average relevance score of matching child objects. |
| `min` | The minimum relevance score of matching child objects. |
| `max` | The maximum relevance score of matching child objects. |
| `sum` | The sum of all matching child objects' relevance scores. |
| `none` | Ignore relevance scores for matching child objects (i.e. 0.0). |

## Lecture summary

- Use the `nested` data type if you want to query objects independently
    - Otherwise the relationships between object values are not maintained
    - Each nested object is indexed as a hidden Lucene document
- Use the `nested` query on fields with the `nested` data type
    - Elasticsearch then handles everything automatically
- Use the `score_mode` parameter to adjust relevance scoring

CodingExplained
CODE. SLEEP. REPEAT.