Search.Api

Полнотекстовый поиск

Поиск осуществляется с помощью следующих endpoint-ов:

- `POST /api/v1/search` — полнотекстовый поиск по одному или нескольким индексам Elastic. Принимает единственный JSON-объект запроса `SearchRequest`. Возвращает JSON-объект `SearchResponse`.
- `POST /api/v1/multi_search` — мультиплексирование нескольких независимых поисковых запросов. Принимает массив JSON-объектов `SearchRequest[]`. Возвращает объект массив JSON-объектов `SearchResponse[]` той же длины, что и массив запросов. Запросы и ответы сопоставляются по порядковому номеру в массиве.

Для получения набора документов, нужно указать один или несколько индексов в поле `$from`:

```
{
  "$from": ["qp.news", "qp.textpages"]
}
```

Допустимы полные имена индексов или wildcard-паттерны.

В результате будет возвращен массив документов:

```
{
  "status": 200,
  "totalCount": 143505,
  "documents": [
    {
      "_id": "12345",
      "_index": "qp.textpages",
      "Title": "Помощь абоненту",
      // ...
    },
    // ...
  ]
}
```

Полнотекстовый поиск фраз

Поисковый ввод пользователя задается в поле `$query`:

```
{
  "$from": "media.materials",
```

```
  "$query": "мобильные приложения"
}
```

При выполнении запроса используются синонимы, морфология, стоп-слова и шинглы. Искать можно только по полям, проиндексированным как текст.

Веса полей

При этом можно указать веса различных полей документа в поле `$weights`:

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$weights": {
    "HeaderTitle": 5,
    "MainTag.Title": 10,
    "Tags": { "Title": 2 }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

Минимальное кол-во найденных слов

С помощью поля `$requiredWordsCount` можно указать минимальное количество найденных слов из `$query`, при котором документ попадает в выдачу. По-умолчанию необходимы ВСЕ слова. Например:

- `3` — должны быть найдены не менее трех слова
- `-1` — должны быть найдены все слова кроме одного
- `"80%"` — должны быть найдены не менее 80% слов

Выбор полей

Каждый возвращаемый документ содержит служебные поля:

- `_id: string` — Строковый идентификатор документа в Elastic
- `_index: string` — Сокращенное имя индекса, которому принадлежит документ
- `_score: number` — Мера релевантности документа в рамках запроса

Остальные поля можно выбрать с помощью поля `$select` (по-умолчанию возвращаются все поля):

```
{
  "$from": "media.materials",
  "$select": ["Id", "Content", "Tags.Title", "Category.*"]
}
```

Допустимы имена полей вложенных объектов через точку или wildcard-паттерны.

Подсвеченные HTML-фрагменты

С помощью поля `$snippets` можно указать, по каким полям документа нужно сгенерировать подсвеченные сниппеты, где `$count` — количество сниппетов (default 5), а `$length` — максимальная длина (default 100):

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": {
    "HeaderTitle": { "$count": 1, "$length": 100 },
    "Tags": { "Title": { "$count": 2, "$length": 50 } }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

В результате полученные сниппеты будут добавлены в каждый документ в спец. поле `_snippets`:

```
{
  // ...
  "documents": [
    {
      "_id": "101391",
      "_snippets": { "HeaderTitle": ["<b>мобильный</b> телефон"] }
    }
    // ...
  ]
}
```

Также можно указать `$count` в сокращенной форме, в виде числа:

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": { "HeaderTitle": 1, "Tags": 2 }
}
```

Можно не указывать конкретного поля для сниппетов,

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": { "$count": 1, "$length": 100 }
}
```

Тогда они будут построены по объединению полей `_all`:

```
{
  // ...
  "documents": [
    {
      "_id": "101391",
      "_snippets": { "_all": ["<b>мобильный</b> телефон"] }
    }
    // ...
  ]
}
```

Для того, чтобы поле не разбивалось на несколько сниппетов, нужно задать `$count: 0`:

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$snippets": { "HeaderTitle": { "$count": 0 } }
}
```

Тогда сниппет будет построен по всему полю целиком.

Сортировка

Сортировка результатов задается в поле `$orderBy`:

- в виде имени поля `"PublishDate"`
- объекта с указанием направления сортировки `{ "PublishDate": "desc" }`
- или набора из нескольких полей: `["Id", { "PublishDate": "desc" }]`

Также можно сортировать по спец. полю `_score`, которое соответствует релевантности документа

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$orderBy": [{ "PublishDate": "desc" }, "_score"]
}
```

По-умолчанию выдача сортируется по `_score`

Постраничный вывод

Ограничить размер выдачи можно с помощью полей `$limit` — размер страницы (по-умолчанию 50) и `$offset` — отступ от начала выдачи:

```
{
  "$from": "media.materials",
  "$limit": 10,
  "$offset": 20
}
```

Исправление поискового ввода

Если при поиске не найдено ни одного результата (или найдено мало), поисковая система может предложить исправление запроса. Чтобы включить эту функциональность нужно задать условия в поле `$correct`:

```
{
  "$from": "media.materials",
  "$query": "могильные приложения",
  "$correct": {
    "$query": { "$ifFoundLte": 5 },
    "$results": { "$ifFoundLte": 2 }
  }
}
```

В данном примере — предложить пользователю исправление запроса, если найдено не более 5 документов. Применять это исправление при поиске результатов, если изначально было найдено не более 2 документов.

В этом случае в ответе API будет присутствовать поле `queryCorrection`

```
{
  // ...
  "queryCorrection": {
    "text": "мобильные приложения",
    "snippet": "<b>мобильные</b> приложения",
    "resultsAreCorrected": true
  }
  // ...
}
```

Флаг `resultsAreCorrected` указывает на то, что исправленный запрос уже был применен при поиске результатов.

Роли пользователя

Если в проекте присутствют роли пользователей и включена индексация полей, а так же в конфигурации проекта API включена поддержка ролей, то можно добавить роль или список ролей через параметр `$roles` (массив).
На стороне API будет произведена выборка разрешенных индексов куда можно делать запросы с

указанной ролью/ролями, после чего содержимое `$from` будет заменено на список доступных индексов.
При указании роли/ролей можно в `$from` использовать `*`, оно всё равно будет заменено на полученный в результате проверки набор индексов.
Пример запроса:

```
{
  "$from": "media.materials",
  "$query": "мобильные приложения",
  "$roles": ["Manager", "Reader"]
}
```

Фильтрация по полям

Фильтрация документов по значениям полей задается в поле `$where`. Можно задать условия на одно или несколько различных полей:

```
{
  "$from": "qp.news",
  "$where": {
    "Regions": { "Alias": ["moskva", "spb"] },
    "Groups.Title": "Новости Абонентам"
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

Условия на разные поля документа объединяются через AND. Условия внутри массива значений одного поля объединяются через OR. Таким образом, в примере выше мы выбираем все новости, которые имеют регион "moskva" или "spb" и принадлежат к группе "Новости Абонентам".

Если поле, для которого задано условие, в исходных документах представлено не скалярным значением, а списком (или является полем одного из объектов в списке), то документ попадет в выдачу, когда условие выполняется хотя бы для одного из элементов этого списка.

Например, данный документ будет удовлетворять условию выше:

```
{
  "Title": "Какая-то новость",
  "Regions": [{ "Alias": "moskva" }, { "Alias": "tula" }],
  "Groups": [
    { "Title": "Новости Абонентам" },
    { "Title": "Новости корпоративным клиентам" }
  ]
}
```

Фильтровать можно по всем полям документа, включая служебные `_id` и `_index`.

Расширенные условия

Если необходимо фильтровать по каким-то более сложным условиям, чем явные значения полей, можно воспользоваться **расширенным синтаксисом** условий:

```
{
  "$from": "qp.news",
  "$where": {
    "Regions": { "Alias": { "$all": ["moskva", "spb"] } },
    "Groups.Title": { "$ne": "Новости Абонентам" },
    "PublishDate": {
      "$gte": "2018-01-01T00:00:00",
      "$lt": "2019-01-01T00:00:00"
    }
  }
}
```

В примере выше: выбираем все новости, которые имеют регион "moskva" **и** "spb", **не** принадлежат к группе "Новости Абонентам" и имеют дату публикации с 2018 по 2019 год.

Каждое расширенное условие на поле представляет собой объект со следующими полями:

- `$eq: scalar` — равно,
- `$ne: scalar` — не равно,
- `$in: scalar[]` — содержит одно из `@alias $any`,
- `$any: scalar[]` — содержит одно из `@alias $in`,
- `$all: scalar[]` — содержит все из,
- `$none: scalar[]` — не содержит ни одного из,
- `$lt: scalar` — меньше,
- `$lte: scalar` — меньше или равно,
- `$gt: scalar` — больше,
- `$gte: scalar` — больше или равно.

Предикаты внутри одного объекта объединяются через AND. Так же, как и с массивом явных значений, мы можем объединить несколько расширенных условий на одно поле через OR:

```
{
  "$from": "qp.*",
  "$where": {
    "Regions.Alias": [{ "$eq": null }, { "$in": ["moskva", "spb"] }]
  }
}
```

Комбинирование условий

Если необходимо задать фильтр, когда условия на одно поле зависят от условий на другое, мы можем воспользоваться булевскими комбинаторами:

- `$every: Condition[]` — должны быть выполнены все условия из списка,
- `$some: Condition[]` — должно быть выполнено хотя бы одно условие из списка,
- `$not: Condition` — условие не должно быть выполнено.

Например, ищем тарифы с ценой до 500 в Калуге или до 1000 в Москве:

```
{
  "$from": "dpc.tariff",
  "$where": {
    "$some": [
      {
        "Regions.Alias": "kaluga",
        "ParametersByAlias.SubscriptionFee.NumValue": { "$lte": 500 }
      },
      {
        "Regions.Alias": "moskva",
        "ParametersByAlias.SubscriptionFee.NumValue": { "$lte": 1000 }
      }
    ]
  }
}
```

Все комбинаторы (`$every`, `$some`, `$not`) могут быть вложены друг в друга произвольным образом:

```
{
  "$from": "dpc.tariff",
  "$where": {
    "$every": [
      {
        "$some": [{ "Regions.Alias": "kaluga" }, { "Regions.Alias": "moskva" }]
      },
      {
        "$not": {
          "ParametersByAlias.SubscriptionFee.NumValue": { "$gt": 1000 }
        }
      }
    ]
  }
}
```

Контекстные поля

Поле документа является контекстным, если при разных условиях фильтрации для одного и того же документа Elastic необходимо выдавать разные значения этого поля.

Запрос для поиска документов также имеет поле `$context`, в котором указан фильтр для контекстных полей документа. Если этот фильтр явно отсутствует, его значение берется из поля `$where`.

Пример: поле `"SearchUrl"`, содержащее Url страницы, который начинается с поддомена. Поддомен в свою очередь зависит от одного из регионов документа, хранящихся в массиве `"Regions"`.

Для этого при индексации в контекстное поле добавляется массив объектов, каждый из которых содержит единственное значение контекстного поля, а также выбранные значения тех полей документа, по которым должна проходить контекстная фильтрация.

Пример:

```
{
  "SearchUrl": [
    {
      "SearchUrl": "http://moskva.domain.ru",
      "Regions": { "Id": 123, "Alias": "moskva" }
    },
    {
      "SearchUrl": "http://spb.domain.ru",
      "Regions": { "Id": 456, "Alias": "spb" }
    }
  ],
  "Regions": [{ "Id": 123, "Alias": "moskva" }, { "Id": 456, "Alias": "spb" }]
  // ... other fields
}
```

Таким образом, в каждом объекте контекстного массива **ОБЯЗАТЕЛЬНО** содержится поле с тем же названием, что и у исходного контекстного поля, плюс поля документа, от которых она зависит.

Далее, для всех имен полей из фильтра `$context` или `$where`, если такое поле содержится в объектах контекстного массива, то по этому полю применяется фильтрация на application-сервере, после загрузки документа из Elastic.

Если в фильтре не указано ни одного подходящего поля, то будет выбран первый попавшийся объект контекстного массива.

Пример: если в фильтре указано

```
{
  "$where": {
    "Regions.Alias": "moskva",
    "Tags": ["foo", "bar"]
  }
}
```

То фильтрация будет производиться только по `"Regions.Alias"`, т.к. `"Tags"` не содержится в контекстных объектах.

После нахождения единственного контекстного объекта из него выбирается значение контекстного поля.

Результат:

```
{
  "SearchUrl": "http://moskva.domain.ru",
  "Regions": [{ "Id": 123, "Alias": "moskva" }, { "Id": 456, "Alias": "spb" }]
  // ... other fields
}
```

Контекстная фильтрация полей поддерживает все выражения кроме булевских комбинаторов `$every`, `$some`, `$not`. Эти комбинаторы **будут проигнорированы** при поиске значения контекстного поля.

Поиск по префиксам слов

Поиск осуществляется с помощью следующих endpoint-ов:

- `POST /api/v1/suggest` — полнотекстовый поиск по одному или нескольким индексам Elastic. Принимает единственный JSON-объект запроса `SuggestRequest`. Возвращает JSON-объект `SuggestResponse`.
- `POST /api/v1/multi_suggest` — мультиплексирование нескольких независимых поисковых запросов. Принимает массив JSON-объектов `SuggestRequest[]`. Возвращает объект массив JSON-объектов `SuggestResponse[]` той же длины, что и массив запросов. Запросы и ответы сопоставляются по порядковому номеру в массиве.

Для получения набора документов, нужно указать один или несколько индексов в поле `$from` и поисковый ввод пользователя в поле `$query`:

```
{
  "$from": ["qp.news", "qp.textpages"],
  "$query": "абон"
}
```

Допустимы полные имена индексов или wildcard-паттерны.

При выполнении запроса используются морфология и префиксы слов. Искать можно только по полям, проиндексированным как текст.

В результате будет возвращен массив документов:

```
{
  "status": 200,
  "totalCount": 110,
  "documents": [
    {
```

```
      "_id": "12345",
      "_index": "qp.textpages",
      "Title": "Помощь абоненту",
      // ...
    },
    // ...
  ]
}
```

Веса полей

При этом можно указать веса различных полей документа в поле `$weights`:

```
{
  "$from": "media.materials",
  "$query": "абон",
  "$weights": {
    "HeaderTitle": 5,
    "MainTag.Title": 10,
    "Tags": { "Title": 2 }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

Минимальное кол-во найденных слов

С помощью поля `$requiredWordsCount` можно указать минимальное количество найденных слов из `$query`, при котором документ попадает в выдачу. По-умолчанию необходимы ВСЕ слова. Например:

- `3` — должны быть найдены не менее трех слова
- `-1` — должны быть найдены все слова кроме одного
- `"80%"` — должны быть найдены не менее 80% слов

Выбор полей

Каждый возвращаемый документ содержит служебные поля:

- `_id: string` — Строковый идентификатор документа в Elastic
- `_index: string` — Сокращенное имя индекса, которому принадлежит документ
- `_score: number` — Мера релевантности документа в рамках запроса

Остальные поля можно выбрать с помощью поля `$select` (по-умолчанию возвращаются все поля):

```
{
  "$from": "media.materials",
  "$select": ["Id", "Content", "Tags.Title", "Category.*"]
}
```

Допустимы имена полей вложенных объектов через точку или wildcard-паттерны.

Подсвеченные HTML-фрагменты

С помощью поля `$snippets` можно указать, по каким полям документа нужно сгенерировать подсвеченные сниппеты, где `$count` — количество сниппетов (default 5), а `$length` — максимальная длина (default 100):

```
{
  "$from": "media.materials",
  "$query": "абон",
  "$snippets": {
    "HeaderTitle": { "$count": 1, "$length": 100 },
    "Tags": { "Title": { "$count": 2, "$length": 50 } }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

В результате полученные сниппеты будут добавлены в каждый документ в спец. поле `_snippets`:

```
{
  // ...
  "documents": [
    {
      "_id": "101391",
      "_snippets": { "HeaderTitle": ["звонки <b>абоненту</b>"] }
    }
    // ...
  ]
}
```

Также можно указать `$count` в сокращенной форме, в виде числа:

```
{
  "$from": "media.materials",
  "$query": "абон",
  "$snippets": { "HeaderTitle": 1, "Tags": 2 }
}
```

Можно не указывать конкретного поля для сниппетов,

```
{
  "$from": "media.materials",
```

```
  "$query": "абон",
  "$snippets": { "$count": 1, "$length": 100 }
}
```

Тогда они будут построены по объединению полей `_all`:

```
{
  // ...
  "documents": [
    {
      "_id": "101391",
      "_snippets": { "_all": ["звонки <b>абоненту</b>"] }
    }
    // ...
  ]
}
```

Для того, чтобы поле не разбивалось на несколько сниппетов, нужно задать `$count: 0`:

```
{
  "$from": "media.materials",
  "$query": "абон",
  "$snippets": { "HeaderTitle": { "$count": 0 } }
}
```

Тогда сниппет будет построен по всему полю целиком.

Сортировка

Сортировка результатов задается в поле `$orderBy`:

- в виде имени поля `"PublishDate"`
- объекта с указанием направления сортировки `{ "PublishDate": "desc" }`
- или набора из нескольких полей: `["Id", { "PublishDate": "desc" }]`

Также можно сортировать по спец. полю `_score`, которое соответствует релевантности документа

```
{
  "$from": "media.materials",
  "$query": "абон",
  "$orderBy": [{ "PublishDate": "desc" }, "_score"]
}
```

По-умолчанию выдача сортируется по `_score`

Размер выдачи

Ограничить размер выдачи можно с помощью полей `$limit` — размер страницы (по-умолчанию 50):

```
{
  "$from": "media.materials",
  "$limit": 10
}
```

Роли пользователя

Если в проекте присутствуют роли пользователей и включена индексация полей, а так же в конфигурации проекта API включена поддержка ролей, то можно добавить роль или список ролей через параметр `$roles` (массив).
На стороне API будет произведена выборка разрешенных индексов куда можно делать запросы с указанной ролью/ролями, после чего содержимое `$from` будет заменено на список доступных индексов.
При указании роли/ролей можно в `$from` использовать `*`, оно всё равно будет заменено на полученный в результате проверки набор индексов.
Пример запроса:

```
{
  "$from": ["qp.news", "qp.textpages"],
  "$query": "абон",
  "$roles": ["Manager", "Reader"]
}
```

Дополнение поискового запроса

Дополнение осуществляется с помощью следующих endpoint-ов:

- **`POST /api/v1/completion`** — дополнение по одному или нескольким индексам Elastic. Принимает единственный JSON-объект запроса `CompletionRequest`. Возвращает JSON-объект `CompletionResponse`.
- **`POST /api/v1/multi_completion`** — мультиплексирование нескольких независимых запросов. Принимает массив JSON-объектов `CompletionRequest[]`. Возвращает объект массив JSON-объектов `CompletionResponse[]` той же длины, что и массив запросов. Запросы и ответы сопоставляются по порядковому номеру в массиве.

Для получения набора дополненных фраз, нужно указать один или несколько индексов в поле `$from` и поисковый ввод пользователя в поле `$query`:

```
{
  "$from": ["qp.news", "qp.textpages"],
  "$query": "москва",
  "$weights": {
    // ...
```

```
  }
}
```

Допустимы полные имена индексов или wildcard-паттерны.

Искать можно только по полям, проиндексированным как текст.

В результате будут возвращены несколько предлагаемых фраз:

```
{
  "status": 200,
  "phrases": ["москва", "москва и область"]
}
```

Веса полей

При этом **необходимо** указать веса полей документа, по которым будет строиться дополнение, в поле `$weights`:

```
{
  "$from": "media.materials",
  "$query": "москва",
  "$weights": {
    "HeaderTitle": 5,
    "MainTag.Title": 10,
    "Tags": { "Title": 2 }
  }
}
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме.

Если конкретные значения весов не важны, а важны только сами поля, можно просто указать все веса равными `1`:

```
{
  "$from": "media.materials",
  "$query": "москва",
  "$weights": {
    "HeaderTitle": 1,
    "MainTag.Title": 1,
    "Tags.Title": 1
  }
}
```

Роли пользователя

Если в проекте присутствют роли пользователей и включена индексация полей, а так же в конфигурации проекта API включена поддержка ролей, то можно добавить роль или список ролей через параметр `$roles` (массив).
На стороне API будет произведена выборка разрешенных индексов куда можно делать запросы с указанной ролью/ролями, после чего содержимое `$from` будет заменено на список доступных индексов.
При указании роли/ролей можно в `$from` использовать `*`, оно всё равно будет заменено на полученный в результате проверки набор индексов.
Пример запроса:

```
{
  "$from": "media.materials",
  "$query": "москва",
  "$roles": ["Manager", "Reader"],
  "$weights": {
    "HeaderTitle": 5,
    "MainTag.Title": 10,
    "Tags": { "Title": 2 }
  }
}
```

Фасетный поиск

Задать фасеты по различным полям можно с помощью поля `$facets`.

Например, интервал для даты публикации и 5 наиболее популярных рубрик:

```
{
  "$from": "qp.news",
  "$facets": {
    "PublishDate": "$interval",
    "Rubrics": { "Title": { "$samples": 5 } }
  }
}
```

В результате будет получен объект со значениями фасетов:

```
{
  "status": 200,
  // ...
  "facets": {
    "PublishDate": {
      "interval": { "from": "1999-05-05T00:00:00", "to": "2019-07-09T17:18:00" }
    },
    "Rubrics.Title": {
      "samples": [
        { "value": "Домашний Интернет и ТВ", "count": 8649 },
```

```
          { "value": "Спецпредложения", "count": 4495 },
          { "value": "Услуги мобильной связи", "count": 4077 },
          { "value": "Тарифы и скидки на звонки", "count": 4031 },
          { "value": "Обслуживание абонентов", "count": 3597 }
        ]
      }
    }
  }
```

Поля вложенных объектов можно объявлять как через точку, так и во вложенной форме. Но в результатах названия фасетов будут всегда представлять собой полные пути к вложенному полю через точку.

Фасеты строятся вместе с выполнением поиска и фильтрации по всему диапазону найденных документов. Таким образом, диапазон построения фасетов ограничивается строкой запроса `$query` и фильтром `$where`. Если при выполнении запроса нам нужны только фасеты (а не найденные документы), то мы можем установить поле `$limit: 0`.

```
{
  "$from": "qp.news",
  "$where": {
    "Regions": { "Alias": ["moskva", "spb"] }
  },
  "$limit": 0,
  "$facets": {
    // ...
  }
}
```

Доступны четыре типа фасетов: **Interval**, **Samples**, **Ranges** и **Percentiles**. Результатом построения каждого фасета является объект с одним полем, где имя поля это тип фасета, а значение это результаты.

Interval Facet

Interval Facet служит для определения минимального и максимального значения поля в выборке. Задается с помощью ключевого слова: `"$interval"`:

```
{
  "$from": "qp.news",
  "$facets": {
    "PublishDate": "$interval"
  }
}
```

Результатом является объект с полями `from` и `to`:

```
{
  "facets": {
    "PublishDate": {
      "interval": { "from": "1999-05-05T00:00:00", "to": "2019-07-09T17:18:00" }
    }
  }
}
```

Samples Facet

Samples Facet служит для нахождения наиболее популярных значений поля в выборке. Задается как объект `{ "$samples": <count> }`, где `<count>` — максимальное количество популярных значений. Или в сокращенной форме, как ключевое слово `"$samples"` (тогда `<count>` = 100).

```
{
  "$from": "qp.news",
  "$facets": {
    "Rubrics.Title": { "$samples": 2 }
  }
}
```

Результатом является массив объектов с полями `value` — значение поля и `count` — кол-во документов, соответствующее этому значению:

```
{
  "facets": {
    "Rubrics.Title": {
      "samples": [
        { "value": "Домашний Интернет и ТВ", "count": 8649 },
        { "value": "Спецпредложения", "count": 4495 }
      ]
    }
  }
}
```

Если мы ищем сразу по нескольким индексам, можно построить фасет по спец. полю `_index`:

```
{
  "$from": "media.*",
  "$facets": {
    "_index": "$samples"
  }
}
```

В результате мы получим распределение документов по индексам:

```
{
  "facets": {
    "_index": {
      "samples": [
        { "value": "media.materials", "count": 2115 },
        { "value": "media.menuandpages", "count": 35 }
      ]
    }
  }
}
```

Ranges Facet

Ranges Facet позволяет разбить значения поля на именованные диапазоны. Задается как объект, содержащий список диапазонов:

```
{
  "$ranges": [
    { "$name": "<range_name>", "$from": "<min_value>", "$to": "<max_value>" }
    // ...
  ]
}
```

Где `$name` — имя диапазона, `$from` — нижняя граница (включая указанное значение), `to` — верхняя граница (не включая указанное значение).

Например:

```
{
  "$from": "dpc.internettariff",
  "$facets": {
    "ParametersByAlias.MaxSpeed.NumValue": {
      "$ranges": [
        { "$name": "high", "$from": 101 },
        { "$name": "mid", "$from": 30, "$to": 101 },
        { "$name": "low", "$to": 30 }
      ]
    }
  }
}
```

Результатом является массив объектов с полями `name` — имя диапазона, `from`, `to` — границы диапазона и `count` — кол-во документов, попавшее в диапазон:

```
{
  "facets": {
    "ParametersByAlias.MaxSpeed.NumValue": {
      "ranges": [
        { "name": "low", "to": 30, "count": 99 },
        { "name": "mid", "from": 30, "to": 101, "count": 165 },
        { "name": "high", "from": 101, "count": 11 }
      ]
    }
  }
}
```

Percentiles Facet

Percentiles Facet служит для построения доверительных интервалов и медианных значений. Задается как объект `{ "$percentiles": number[] }`;, содержащий список процентных значений.

Например, доверительный интервал 5-95 %:

```
{
  "$from": "dpc.device",
  "$limit": 0,
  "$facets": {
    "ParametersByAlias.SalePrice.NumValue": {
      "$percentiles": [5, 95]
    }
  }
}
```

Результатом является массив объектов с полями `percent` — процент распределения вероятностей и `value` — соответствующее значение поля:

```
{
  "facets": {
    "ParametersByAlias.SalePrice.NumValue": {
      "percentiles": [
        { "percent": 5, "value": 1300 },
        { "percent": 95, "value": 6500 }
      ]
    }
  }
}
```

Другие примеры Percentiles Facet:

- `{ "$percentiles": [50] }` — медиана
- `{ "$percentiles": [0, 100] }` — минимум и максимум

- `{ "$percentiles": [90, 99, 99.9, 99.99] }` — уровни доверия в процентах