Руководство пользователя
РЕД Вектор Данных

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

РЕД Вектор Данных – это поисковая система векторного сходства, разработанная компанией РЕД СОФТ ЦЕНТР.

Данный продукт предоставляет готовый к использованию сервис с удобным API для хранения, поиска и управления векторами с дополнительной полезной нагрузкой. Полезную нагрузку можно рассматривать как дополнительные фрагменты информации, уточняющие поиск и имеющие возможность получить полезную информацию.

В этом руководстве представлено описание наиболее важных объектов системы: коллекций и векторов. Благодаря эффективным методам индексирования и поиска, векторные базы данных позволяют быстрее и точнее извлекать неструктурированные данные, уже представленные в виде векторов, что помогает показывать пользователям наиболее релевантные результаты их запросов.

1 ПОДКЛЮЧЕНИЕ К СИСТЕМЕ

1.1 Запуск РЕД Вектор Данных

1.1.1 Запуск сервера

Для запуска контейнера с сервером необходимо указать:

```
docker run -d --name red-vector-db --restart=unless-stopped -p 6333:6333 -p 6334:6334 red-vector-db:latest
```

где:

- **-d** запуск контейнера в фоновом режиме;
- **--name red-v-server** задание имени контейнеру;
- **--restart=unless-stopped** автоматический перезапуск контейнера в случае остановки;
- **-p 6333:6333 -p 6334:6334** проброс портов 6333 и 6334 из контейнера на хост.

1.1.2 Проверка успешного запуска сервера

Чтобы убедиться, что сервер успешно запущен, необходимо открыть веб-браузер и перейти по следующему адресу:

```
http://localhost:6333/
```

Если сервер запущен успешно, то откроется соответствующая веб-страница с информацией о сервере.

1.2 Авторизация пользователя в Ред Вектор Данных

1.2.1 Виды доступа пользователей

По умолчанию в РЕД Вектор Данных доступ открыт для всех пользователей. РЕД Вектор Данных при этом поддерживает простую форму аутентификации клиента с использованием статического ключа API, что можно использовать для защиты вашего экземпляра РЕД Вектор Данных.

Для включения аутентификации на основе API-ключа в вашем собственном экземпляре РЕД Вектор Данных необходимо указать ключ в конфигурации либо использовать переменную окружения (порядок настройки в Руководстве администратора, раздел «Аутентификация»).

В более сложных случаях РЕД Вектор Данных поддерживает детальный контроль доступа с помощью JSON Web Tokens (JWT). Это позволяет создавать токены, ограничивающие доступ к данным, хранящимся в вашем кластере, и на их основе

строить систему управления доступом на основе ролей к различным сущностям системы.

1.2.2 Процесс аутентификации с помощью API токена

После активации ключа все запросы к REST и gRPC интерфейсам должны содержать этот ключ. Ключ передается в HTTP-заголовке **api-key**:

```
curl -X GET "http://localhost:6333/collections" \
    -H "api-key: <ваш_API_токен>"
```

РЕД Вектор Данных проверяет наличие заголовка **api-key** в каждом входящем пакете.

Если заголовок отсутствует или ключ не совпадает с настроенным, сервер возвращает ошибку **403 Forbidden**. Если ключ совпадает с **read_only_api_key**, сервер разрешает только GET-запросы и блокирует любые изменения данных.

1.2.3 Процесс аутентификации с помощью JWT токена

После получения токена пользователь (или клиентское приложение) передает его в каждом запросе одним из двух способов:

HTTP Header (рекомендуется):

```
curl -X GET "http://localhost:6333/collections" \
    -H "Authorization: Bearer <ваш_JWT_токен>"
```

API-Key Header:

```
curl -X GET "http://localhost:6333/collections" \
    -H "api-key: <ваш_JWT_токен>"
```

Сервер получает запрос и извлекает токен из заголовка. РЕД Вектор Данных проверяет подпись токена, используя свой **api_key**. Токен должен содержать уровень доступа и срок действия токена. Если подпись верна и срок действия (**exp**) не истек, сервер анализирует права из **payload** («полезной нагрузки») и разрешает или отклоняет операцию.

Токен JWT позволяет передавать в **payload** токена список разрешенных коллекций и тип операций (**ro** – только чтение или **rw** – чтение и запись).

2 УПРАВЛЕНИЕ КОЛЛЕКЦИЯМИ

В большинстве случаев следует использовать только одну коллекцию с разделением данных на основе полезной нагрузки. Создавать несколько коллекций следует при ограниченном количестве пользователей и необходимости изоляции. Этот подход может привести к перегрузке ресурсов. Кроме того, необходимо убедиться, что коллекции никак не влияют друг на друга, в том числе с точки зрения производительности.

2.1 Создать коллекцию

2.1.1 Создание коллекции

Для хранения векторов сначала нужно создать коллекцию, указав размер вектора (size) и метрику расстояния (distance).

```
curl -X PUT http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
    "vectors": {
      "size": 100,
      "distance": "Cosine"
    }
  }'
```

В этом примере создается коллекция collection_name для векторов размерностью 100 с использованием косинусного расстояния.

2.1.2 Параметры новой коллекции

2.1.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.1.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

2.1.2.3 Параметры запроса

timeout *integer Optional*

Если время ожидания истечет, запрос вернет ошибку обслуживания.

2.1.2.4 Запрос

vectors *object or map from strings to objects Optional*

Разделитель векторных параметров для одно- и многовекторных режимов. Одновекторный режим:

{ “size”: 128, “distance”: “Cosine” }

Многовекторный режим:

{ “default”: { “size”: 128, “distance”: “Cosine” } }

shard_number *uint or null Optional*

Для автоматического шардирования: Количество шардов в коллекции. По умолчанию 1 для автономного режима, в противном случае равно количеству узлов. Минимальное значение - 1.

Для пользовательского шардирования: количество шардов в коллекции на группу шардов. По умолчанию равно 1, это означает, что каждый ключ шарда будет сопоставлен с одним шардом. Минимальное значение - 1.

on_disk_payload *boolean or null Optional*

Определяет, где хранить данные полезной нагрузки. Если **true** — полезная нагрузка будет храниться только на диске. Может быть полезно для ограничения использования оперативной памяти в случае больших объемов данных. По умолчанию: **true**.

hnsw_config *object or any Optional*

Пользовательские параметры для индекса HNSW. Если параметров нет, используются значения из файла конфигурации сервиса.

2.1.2.5 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

2.2 Коллекция с несколькими векторами

В одной записи может быть несколько векторов. Эта функция позволяет хранить несколько векторов в одной коллекции. Для различения векторов в одной записи необходимо задать уникальное имя при создании коллекции. В этом режиме каждый именованный вектор имеет свое расстояние и размер:

```
curl -X PUT http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
    "vectors": {
      "image": {
        "size": 4,
        "distance": "Dot"
      },
      "text": {
        "size": 8,
        "distance": "Cosine"
      }
    }
  }'
```

В редких случаях можно создать коллекцию без использования векторного хранилища.

2.3 Коллекция с разреженными векторами

В коллекциях могут содержаться разреженные векторы в качестве дополнительных именованных векторов наряду с обычными плотными векторами в одной точке.

В отличие от плотных векторов, разреженные векторы должны иметь имена. Кроме того, разреженные и плотные векторы должны иметь разные имена в рамках одной коллекции.

```
curl -X PUT http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
    "sparse_vectors": {
      "text": { }
    }
  }'
```

Помимо уникального имени, для разреженных векторов не требуется никаких дополнительных параметров конфигурации.

Функция расстояния для разреженных векторов всегда задана Dot и не требует указания.

2.4 Создать коллекцию из другой коллекции

Для создания коллекции из другой коллекции используйте инструмент миграции. С его помощью можно скопировать коллекцию как внутри одного экземпляра РЕД Вектор Данных, так и в другой экземпляр.

Например, чтобы скопировать коллекцию из локального экземпляра в экземпляр РЕД Вектор Данных, выполните следующую команду:

```
docker run --net=host --rm -it registry.cloud.red_vector_db.io/library/red_vector_db-migration red_vector_db \
  --source.url 'http://localhost:6334' \
  --source.collection 'source-collection' \
  --target.url 'https://example.cloud-region.cloud-provider.cloud.red_vector_db.io:6334' \
  --target.api-key 'red_vector_db-key' \
  --target.collection 'target-collection' \
  --migration.batch-size 64
```

2.5 Проверить существование коллекции

2.5.1 Проверка существования коллекции

```
curl -X GET http://localhost:6333/collections/{collection_name}/exists
```

2.5.2 Параметры проверки

2.5.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.5.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

2.5.2.3 Ответ

Успешная операция:

usage *object or any or null*

time *double or null*

status *string or null*
result *object or null*

2.6 Обновление параметров коллекции

2.6.1 Обновление параметров коллекции

Следующая команда включает индексирование сегментов, содержащих более 10000 КБ векторов:

```
PATCH /collections/{collection_name}
{
   "optimizers_config": {
      "indexing_threshold": 10000
   }
}
```

2.6.2 Параметры обновления

2.6.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.6.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

2.6.2.3 Параметры запроса

timeout *integer Optional*

Ожидайте истечения времени ожидания операции подтверждения в секундах. Если время ожидания истечет, запрос вернет ошибку обслуживания.

2.6.2.4 Запрос

vectors *map from strings to objects or any Optional*

Карта параметров векторных данных для обновления каждого именованного вектора. Для обновления параметров в коллекции, содержащей один безымянный вектор, используйте пустую строку в качестве имени.

optimizers_config *object or any Optional*

hnsw_config *object or any Optional*

Параметры HNSW, которые необходимо обновить для индекса коллекции. Если таких параметров нет, они остаются без изменений.

quantization_config *object or enum or any Optional*

Параметры квантования для обновления. Если параметров нет, они остаются без изменений.

sparse_vectors *map from strings to objects or any Optional*

Карта параметров разреженных векторных данных, которые необходимо обновить для каждого разреженного вектора.

2.6.2.5 Ответ

Успешная операция:

usage *object or any or null*

time *double or null*

status *string or null*

result *object or null*

2.7 Удалить коллекцию

2.7.1 Удаление коллекции

```
curl -X DELETE http://localhost:6333/collections/{collection_name}
```

Удаляет указанную коллекцию и все связанные с ней данные.

2.7.2 Параметры удаления

2.7.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.7.2.2 Параметры пути

Название коллекции для удаления:

collection_name *string Required*

2.7.2.3 Параметры запроса

timeout *integer Optional*

Ожидайте истечения времени ожидания операции подтверждения в секундах. Если время ожидания истечет, запрос вернет ошибку обслуживания.

2.7.2.4 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *boolean or null*

2.8 Список всех коллекций

2.8.1 Список всех коллекций

```
CURL -X GET http://localhost:6333/collections
```

2.8.2 Параметры списка коллекций

2.8.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.8.2.2 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

2.9 Проверить существование коллекции

2.9.1 Проверить существование коллекции

Проверяет, существует ли указанная коллекция:

```
CURL -X GET http://localhost:6333/collections/{collection_name}/EXISTS
```

2.9.2 Параметры существования коллекции

2.9.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.9.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

2.9.2.3 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

2.10 Информация о коллекции

2.10.1 Информация о коллекции

Ред Вектор Данных позволяет определять параметры конфигурации существующей коллекции, чтобы лучше понимать, как точки распределены и индексированы.

```
curl -X GET http://localhost:6333/collections/{collection_name}
```

Если вы вставите векторы в коллекцию, **status** поле может измениться на **yellow** во время оптимизации. Оно изменится на **green** после успешной обработки всех точек.

2.10.2 Параметры информации о коллекции

2.10.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

2.10.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

2.10.2.3 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

2.10.3 Статусы коллекции

Возможны следующие цветовые статусы:

- **green**: зеленый, коллекция готова,
- **yellow**: желтый, коллекция оптимизируется,
- **grey**: серый, коллекция ожидает оптимизации,
- **red**: красный, произошла ошибка, от которой двигатель не смог восстановиться.

Коллекция может иметь серый статус **grey** или отображать статус оптимизации «ожидаются оптимизации, ожидается операция обновления». Это означает, что в коллекции есть незавершенные оптимизации, но они приостановлены. Для запуска и возобновления оптимизаций необходимо отправить любую операцию обновления.

Пример.

```
curl -X PATCH http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
    "optimizers_config": {}
  }'
```

3 РАБОТА С ТОЧКАМИ (ВЕКТОРАМИ И МЕТАДАННЫМИ)

3.1 Добавление точек

"Точка" – это комбинация уникального ID, вектора (или нескольких векторов) и опциональной полезной нагрузки (payload, метаданные). Можно добавлять новые точки или обновлять существующие с помощью операции UPSERT.

```
curl -X PUT 'http://localhost:6333/collections/my_collection/points' \
--header 'Content-Type: application/json' \
--data-raw '{
   "points": [
      {
         "id": 1,
         "vector": [0.1, 0.2, 0.3, ...],
         "payload": {"city": "Berlin", "age": 42}
      },
      {
         "id": 2,
         "vector": [0.4, 0.5, 0.6, ...],
         "payload": {"city": "London", "age": 35}
      }
   ]
}'
```

Замените [0.1, 0.2, ...] на ваш вектор, размер которого должен соответствовать размеру коллекции (100 в примере выше).

3.2 Получение точки по ID

Получение конкретной точки и ее метаданных:

```
curl -X GET 'http://localhost:6333/collections/my_collection/points/1'
```

3.3 Удаление точки

3.3.1 Удаление точки

Удаление определенной точки из коллекции:

```
curl -X DELETE 'http://localhost:6333/collections/my_collection/points/1'
```

3.3.2 Параметры удаления точки

3.3.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

3.3.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

3.3.2.3 Параметры запроса

wait *boolean Optional*

Если значение true, то изменения произведены.

ordering *enum Optional*

Определяет порядок выполнения операции.

3.3.2.4 Запрос

Операции по выполнению запроса:

PointIdsList *object Required*

Свойства объекта:

points *list of uint64s or strings Required*

shard_key *string or uint64 or list of strings or uint64s or object or any Optional*

FilterSelector *object Required*

Свойства объекта:

filter *object Required*

shard_key *string or uint64 or list of strings or uint64s or object or any Optional*

3.3.2.5 Ответ

Успешная операция:

usage *object or any or null*

time *double or null*

status *string or null*

result *object or null*

3.4 Извлечение точки

3.4.1 Извлечение точки

Универсальная точка запроса, охватывает все возможности поиска, рекомендаций, обнаружения и фильтрации. А также позволяет выполнять гибридные и многоэтапные запросы.

```
curl -X POST \
  'http://localhost:6333/collections/collection_name/points/query' \
  --header 'api-key: <api-key-value>' \
  --header 'Content-Type: application/json' \
  --data-raw '{
  "query": "43cf51e2-8777-4f52-bc74-c2cbde0c8b04"
}'
```

3.4.2 Параметры извлечения точки

3.4.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

3.4.2.2 Параметры пути

Название коллекции, из которой нужно получить данные:

collection_name *string Required*

3.4.2.3 Параметры запроса

consistency *integer or enum Optional*

Определите гарантии согласованности чтения для данной операции.

timeout *integer Optional >=1*

Если тайм-аут задан, то он отменяет глобальный тайм-аут для этого запроса. Единица измерения – секунды.

3.4.2.4 Запрос

Ищет точки с идентификаторами:

ids *list of uint64s or strings Required*

Укажите, в каких shard искать точки; если не указано иное, поиск по всем **shard**:

shard_key *string or uint64 or list of strings or uint64s or object or any Optional*

Выберите, какие данные следует вернуть в ответе. По умолчанию — **true**.

with_payload *boolean or list of strings or object or any Optional*

Варианты указания того, какой вектор следует включить:

with_vector *boolean or list of strings Optional*

3.4.2.5 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *list or object or null*

4 УПРАВЛЕНИЕ ВЕКТОРАМИ

4.1 Векторные типы данных

Для получения векторного представления объекта необходимо применить к нему алгоритм векторизации. Наиболее распространенными типами векторов являются плотные и разреженные векторы.

Плотные векторы	Обычные векторы, генерируемые большинством моделей встраивания.
Разреженные векторы	Векторы без фиксированной длины, но с небольшим количеством ненулевых элементов. Полезны для точного сопоставления токенов и рекомендаций на основе коллаборативной фильтрации.
Мультивекторы	Матрицы чисел с фиксированной длиной, но переменной высотой.

К одной точке можно прикрепить более одного типа векторов, такие векторы считаем именованными векторами.

4.2 Обновление векторов

4.2.1 Обновление векторов

Чтобы сохранить векторные данные на диск для коллекции, в которой нет именованных векторов, используйте "" в качестве имени:

```
curl -X PATCH http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
    "vectors": {
      "": {
        "on_disk": true
      }
    }
  }'
```

Чтобы записать векторные данные на диск для коллекции, содержащей именованные векторы:

```
curl -X PATCH http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
```

```
  "vectors": {
    "my_vector": {
      "on_disk": true
    }
  }
}'
```

В следующем примере обновляются индекс HNSW и параметры квантования как для всей коллекции, так и для **my vector** конкретной коллекции:

```
curl -X PATCH http://localhost:6333/collections/{collection_name} \
 -H 'Content-Type: application/json' \
 --data-raw '{
  "vectors": {
    "my_vector": {
      "hnsw_config": {
        "m": 32,
        "ef_construct": 123
      },
      "quantization_config": {
        "product": {
          "compression": "x32",
          "always_ram": true
        }
      },
      "on_disk": true
    }
  },
```

```
  "hnsw_config": {
    "ef_construct": 123
  },
  "quantization_config": {
    "scalar": {
      "type": "int8",
      "quantile": 0.8,
      "always_ram": false
    }
  }
}'
```

4.2.2 Параметры обновления векторов

4.2.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

4.2.2.2 Параметры пути

Название коллекции, из которой нужно обновить данные:

collection_name *string Required*

4.2.2.3 Параметры запроса

wait *boolean Optional*

Если значение true, то изменения произведены:

ordering *enum Optional*

Определяет порядок выполнения операции. Допустимые значения weak, medium, strong.

4.2.2.4 Запрос

Обновить именованные векторы в точках:

points *list of objects Required*

shard_key *string or uint64 or list of strings or uint64s or object or any Optional*
update_filter *object or any Optional*

4.2.2.5 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

4.3 Удаление векторов

4.3.1 Удаление векторов

Удаляет указанные векторы из точек. Все остальные неуказанные векторы остаются неизменными.

```
# Delete vectors by ID
curl  -X POST \
  'http://localhost:6333/collections/collection_name/points/vectors/delete' \
  --header 'api-key: <api-key-value>' \
  --header 'Content-Type: application/json' \
  --data-raw '{
  "points": [
    0,
    3,
    10
  ],
  "vectors": [
    "text",
    "image"
  ]
}'
# Delete vectors by filter
```

```
curl  -X POST \
  'http://localhost:6333/collections/collection_name/points/vectors/delete' \
  --header 'api-key: <api-key-value>' \
  --header 'Content-Type: application/json' \
  --data-raw '{
  "filter": {
    "must": [
      {
        "key": "color",
        "match": {
          "value": "red"
        }
      }
    ]
  },
  "vectors": [
    "text",
    "image"
  ]
}'
```

4.3.2 Параметры удаления векторов

4.3.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

4.3.2.2 Параметры пути

Название коллекции, из которой нужно удалить данные:

collection_name *string Required*

4.3.2.3 Параметры запроса

wait *boolean Optional*

Если значение true, то изменения произведены.

ordering *enum Optional*

Определяет порядок выполнения операции. Допустимые значения weak, medium, strong.

4.3.2.4 Запрос

Удалить именованные векторы из точек. Название вектора:

vector *list of strings Required*

Удаляет значения из каждой точки этого списка:

points *list of objects Required*

Удаляет значения из точек, удовлетворяющих данному условию фильтра:

filter *object or any Optional*
shard_key *string or uint64 or list of strings or uint64s or object or any Optional*

4.3.2.5 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

4.4 Плотные векторы

Это наиболее распространенный тип вектора. Это простой список чисел, имеющий фиксированную длину, и каждый элемент списка представляет собой число с плавающей запятой.

```
// A PIECE OF A REAL-WORLD DENSE VECTOR
[
  -0.013052909,
  0.020387933,
  -0.007869,
  -0.11111383,
  -0.030188112,
  -0.0053388323,
```

```
  0.0010654867,
  0.072027855,
  -0.04167721,
  0.014839341,
  -0.032948174,
  -0.062975034,
  -0.024837125,
  ....
]
```

4.5 РАЗРЕЖЕННЫЕ ВЕКТОРЫ

Разреженные векторы не имеют фиксированной длины, поскольку она динамически выделяется во время вставки вектора. Количество ненулевых значений в разреженных векторах в настоящее время ограничено диапазоном типов данных u32 (4294967295).

Для определения разреженного вектора необходимо указать список ненулевых элементов и их индексы.

```
// A sparse vector with 4 non-zero elements
{
   "indexes": [1, 3, 5, 7],
   "values": [0.1, 0.2, 0.3, 0.4]
}
```

Разреженные векторы хранятся в специальном хранилище и индексируются отдельным индексом, поэтому их конфигурация отличается от конфигурации плотных векторов.

Для создания коллекции с разреженными векторами:

```
curl -X PUT http://localhost:6333/collections/{collection_name} \
  -H 'Content-Type: application/json' \
  --data-raw '{
    "sparse_vectors": {
        "text": { }
    }
  }'
```

4.6 Мультивекторы

РЕД Вектор Данных поддерживает хранение переменного количества одинаковых по форме плотных векторов в одной точке. Это означает, что вместо одного плотного вектора вы можете загрузить матрицу плотных векторов.

Длина матрицы фиксирована, но количество векторов в матрице может быть различным для каждой точки. Мультивекторы выглядят следующим образом:

```
// A MULTIVECTOR OF SIZE 4
"VECTOR": [
  [-0.013, 0.020, -0.007, -0.111],
  [-0.030, -0.055, 0.001, 0.072],
  [-0.041, 0.014, -0.032, -0.062],
  ....
]
```

Существует два сценария, в которых мультивекторы оказываются полезными:

1) Множественное представление одного и того же объекта — например, можно хранить несколько векторных представлений для изображений одного и того же объекта, снятых с разных ракурсов. Этот подход предполагает, что полезная нагрузка одинакова для всех векторов.

2) Встраивание текста с поздним взаимодействием — некоторые модели встраивания текста могут выдавать несколько векторов для одного текста. Например, семейство моделей, таких как ColBERT, выдает относительно небольшой вектор для каждого токена в тексте.

4.7 Создание индекса полезной нагрузки

4.7.1 Создание индекса полезной нагрузки

```
CURL -X PUT \
 'HTTP://LOCALHOST:6333/COLLECTIONS/COLLECTION_NAME/INDEX' \
 --HEADER 'API-KEY: <API-KEY-VALUE>' \
 --HEADER 'CONTENT-TYPE: APPLICATION/JSON' \
 --DATA-RAW '{
 "FIELD_NAME": "FIELD_NAME",
 "FIELD_SCHEMA": "KEYWORD"
}'
```

4.7.2 Параметры создания полезной нагрузки

4.7.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

4.7.2.2 Параметры пути

Название коллекции, из которой будет выбран набор:

collection_name *string Required*

4.7.2.3 Параметры запроса

wait *boolean Optional*

Если значение true, то изменения произведены.

ordering *enum Optional*

Определяет порядок выполнения операции. Допустимые значения weak, medium, strong.

4.7.2.4 Запрос

Установить полезную нагрузку на точках:

payload *map from strings to any Required*

Назначает полезную нагрузку каждой точке в этом списке:

points *list of uint64s or strings or null Optional*

Назначает полезную нагрузку каждой точке, удовлетворяющей условию этого фильтра:

filter *object or any Optional*
shard_key *string or uint64 or list of strings or uint64s or object or any Optional*
key *string or null Optional*

4.7.2.5 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*

result *object or null*

4.8 Удаление индекса полезной нагрузки

4.8.1 Удаление индекса полезной нагрузки

```
curl  -X DELETE \
  'http://localhost:6333/collections/collection_name/index/field_name' \
  --header 'api-key: <api-key-value>'
```

Удаляет индекс полезной нагрузки для поля в указанной коллекции.

4.8.2 Параметры удаления полезной нагрузки

4.8.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

4.8.2.2 Параметры пути

Название коллекции, из которой нужно удалить данные:

collection_name *string Required*

4.8.2.3 Параметры запроса

wait *boolean Optional*

Если значение true, то изменения произведены.

ordering *enum Optional*

Определяет порядок выполнения операции. Допустимые значения weak, medium, strong.

4.8.2.4 Запрос

Удалить полезную нагрузку в точках. Список ключей полезной нагрузки, которые необходимо удалить из полезной нагрузки:

keys *list of strings Required*

Удаляет значения из каждой точки этого списка:

points *list of uint64s or strings or null Optional*

Удаляет значения из точек, удовлетворяющих данному условию фильтра:

filter *object or any Optional*

shard_key *string or uint64 or list of strings or uint64s or object or any Optional*

4.8.2.5 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *object or null*

4.9 Поиск сходства

В зависимости от **query** параметров, можно отдавать предпочтение различным стратегиям поиска.

Поиск ближайших соседей	Поиск векторного сходства, также известный как k-NN
Поиск по идентификатору	Поиск по уже сохраненному вектору — пропуск вывода модели встраивания

Поиск ближайших соседей:

```
POST /collections/{collection_name}/points/query
{
    "query": [0.2, 0.1, 0.9, 0.7] // <--- Dense vector
}
```

Поиск по идентификатору:

```
POST /collections/{collection_name}/points/query
{
    "query": "43cf51e2-8777-4f52-bc74-c2cbde0c8b04" // <--- point id
}
```

4.10 Массовая загрузка векторов

4.10.1 Массовая загрузка векторов

При быстрой загрузке больших наборов данных рекомендуем использовать клиентскую библиотеку на Rust. При пакетной загрузке частые обновления индекса могут снизить пропускную способность и увеличить потребление ресурсов.

Для управления этим поведением и оптимизации с учетом ограничений вашей системы, отрегулируйте следующие параметры:

Ваша цель	Что делать	Конфигурация
Максимальная скорость загрузки, устойчивость к высокому потреблению оперативной памяти.	Полностью отключить индексирование	indexing_threshold: 0
Низкий уровень потребления памяти во время загрузки.	Отложить построение графика HNSW (рекомендуется)	m: 0
Более быстрая доступность индекса после загрузки.	Оставьте индексирование включенным (поведение по умолчанию).	m: 16, indexing_threshold: 20000 (по умолчанию)

Для активации быстрого поиска HNSW, если он был отключен во время загрузки, индексирование необходимо повторно включить после загрузки.

4.10.2 Отложить построение графика HNSW

Для плотных векторов m установка параметра HNSW полностью отключает построение индекса. Векторы по-прежнему будут храниться, но не будут индексироваться до тех пор, пока вы не включите индексирование позже.

```
PUT /collections/{collection_name}
{
  "vectors": {
    "size": 768,
    "distance": "Cosine"
  },
  "hnsw_config": {
    "m": ⊙
  }
}
```

По завершению повторно включите HNSW, установив значение m, соответствующее вашей версии программного продукта (обычно 16 или 32).

```
PATCH /collections/{collection_name}
{
```

```
  "vectors": {
    "size": 768,
    "distance": "Cosine"
  },
  "hnsw_config": {
     "m": 16
  }
}
```

4.10.3 Полностью отключить индексирование (indexing_threshold: 0)

Если вы выполняете первоначальную загрузку большого набора данных, стоит отключить индексирование во время загрузки. Это позволит избежать ненужного индексирования векторов, которые будут перезаписаны следующей партией данных.

Установка **indexing_threshold** этого параметра ⊙ полностью отключает индексирование:

```
PUT /collections/{collection_name}
{
  "vectors": {
    "size": 768,
    "distance": "Cosine"
  },
  "optimizers_config": {
     "indexing_threshold": ⊙
  }
}
```

После завершения загрузки вы можете включить индексирование, установив **indexing_threshold** желаемое значение (по умолчанию — 20000):

```
PATCH /collections/{collection_name}
{
  "optimizers_config": {
     "indexing_threshold": 20000
  }
}
```

4.10.4 Загрузка непосредственно на диск

При создании коллекции можно включить отображение данных на диск (**memmaps**) для каждого вектора отдельно, используя соответствующий **on_disk** параметр. Это позволит постоянно хранить векторные данные непосредственно на диске. Такой подход подходит для обработки больших объемов данных, необходимых для измерения масштаба в миллиарды единиц.

Использование метода **memmap_threshold** в этом случае не рекомендуется. Это потребует от оптимизатора постоянного преобразования сегментов в оперативной памяти в сегменты отображения памяти на диске.

4.10.5 Параллельная загрузка в несколько сегментов

Каждая коллекция разбивается на сегменты. Каждый сегмент имеет отдельный журнал предварительной записи (WAL), который отвечает за упорядочивание операций. Создание нескольких сегментов позволяет распараллелить загрузку большого набора данных. Разумным количеством является от 2 до 4 сегментов на одной машине.

```
PUT /collections/{collection_name}
{
   "vectors": {
     "size": 768,
     "distance": "Cosine"
   },
   "shard_number": 2
}
```

4.11 Масштабный поиск

Для точного контроля степени избыточной выборки мы будем использовать следующий поисковый запрос:

```
limit = 50
rescore_limit = 1000 # oversampling factor is 20

query = vectors[query_id] # One of existing vectors

response = client.query_points(
      collection_name=RED_VECTOR_DB_COLLECTION_NAME,
      query=query,
      limit=limit,
```

```
    # Go to disk
    search_params=models.SearchParams(
      quantization=models.QuantizationSearchParams(
        rescore=True,
      ),
    ),
    # Prefetch is performed using only in-RAM data,
    # so querying even large amount of data is fast
    prefetch=models.Prefetch(
      query=query,
      limit=rescore_limit,
      params=models.SearchParams(
        quantization=models.QuantizationSearchParams(
          # Avoid rescoring in prefetch
          # We should do it explicitly on the second stage
          rescore=False,
        ),
      )
    )
  )
```

Запрос состоит из двух этапов:

1) предварительная выборка, которая выполняется с использованием только данных из оперативной памяти. Она очень быстрая и позволяет получить большое количество кандидатов;

2) переоценка, которая выполняется с использованием векторов полного размера, хранящихся на дисках.

Использование двухэтапного поиска позволяет точно контролировать объем данных, загружаемых с диска, и обеспечивать баланс между скоростью и точностью поиска.

5 БЭКАПИРОВАНИЕ ДАННЫХ

5.1 Создание снимка

5.1.1 Создание снимка

```
curl  -X POST \
 'http://localhost:6333/collections/collection_name/snapshots' \
 --header 'api-key: <api-key-value>'
```

Создает новый снимок для указанной коллекции.

5.1.2 Параметры создания снимка

5.1.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

5.1.2.2 Параметры пути

Название коллекции, для которой нужно создать снимок:

collection_name *string Required*

5.1.2.3 Параметры запроса

wait *boolean Optional*

Если true, дождаться фактического внесения изменений. Если false — позволить изменениям происходить в фоновом режиме. По умолчанию — true.

5.1.2.4 Ответ

Успешная операция:

time *double or null*

status *string or null*

result *object or null*

5.2 Удаление снимка

5.2.1 Удаление снимка

```
curl  -X DELETE \
'http://localhost:6333/collections/collection_name/snapshots/snapshot_name' \
 --header 'api-key: <api-key-value>'
```

Удаляет указанный снимок коллекции.

5.2.2 Параметры удаления снимка

5.2.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

5.2.2.2 Параметры пути

Название коллекции, для которой нужно создать снимок:

collection_name *string Required*

Название снимка для удаления:

snapshot_*name string Required*

5.2.2.3 Параметры запроса

wait *boolean Optional*

Если true, дождаться фактического внесения изменений. Если false — позволить изменениям происходить в фоновом режиме. По умолчанию — true.

5.2.2.4 Ответ

Успешная операция:

time *double or null*
status *string or null*
result *boolean or null*

5.3 Список снимков

5.3.1 Список снимков

```
curl  -X GET \
  'http://localhost:6333/collections/collection_name/snapshots' \
  --header 'api-key: <api-key-value>'
```

Получает список всех снимков для указанной коллекции.

5.3.2 Параметры списка снимков

5.3.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

5.3.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

5.3.2.3 Ответ

Успешная операция:

usage *object or any or null*
time *double or null*
status *string or null*
result *list or objects or null*

5.4 Восстановление из снимка

5.4.1 Восстановление из снимка

```
curl  -X PUT \
  'http://localhost:6333/collections/collection_name/snapshots/recover' \
  --header 'api-key: <api-key-value>' \
  --header 'Content-Type: application/json' \
  --data-raw '{
  "location": "http://example.com/path/to/snapshot.shapshot"
}'
```

Восстанавливает локальные данные коллекции из снимка. При этом будут перезаписаны любые данные коллекции, хранящиеся на узле. Если коллекция не существует, она будет создана.

5.4.2 Параметры восстановления из снимка

5.4.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

5.4.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

5.3.2.3 Параметры запроса

wait *boolean Optional*

Если true, дождаться фактического внесения изменений. Если false — позволить изменениям происходить в фоновом режиме. По умолчанию — true.

5.3.2.4 Запрос

Расположение файла:

location *string Required format: "uri"*

Определяет, какие данные следует использовать в качестве источника достоверной информации, если в кластере есть другие реплики. Если установлено значение Snapshot, в качестве источника достоверной информации будет использоваться снимок, а текущее состояние будет перезаписано. Если установлено значение Replica, в качестве источника достоверной информации будет использоваться текущее состояние, и после восстановления оно будет синхронизировано со снимком.

priority *enum or any Optional*

Дополнительная контрольная сумма SHA256 используется для проверки целостности снимка перед восстановлением:

checksum *string or null Optional*

Необязательный ключ API, используемый при получении снимка с удаленного URL-адреса:

api_key *string or null Optional*

5.3.2.5 Ответ

Успешная операция:

time *double or null*

status *string or null*

result *boolean or null*

5.5 Восстановление из загруженного снимка

5.5.1 Восстановление из загруженного снимка

```
curl  -X POST \
 'http://localhost:6333/collections/collection_name/snapshots/upload' \
 --header 'api-key: <api-key-value>' \
 --form 'snapshot=@/path/to/snapshot.shapshot'
```

Восстанавливает локальные данные коллекции из загруженного снимка. При этом будут перезаписаны любые данные коллекции, хранящиеся на узле. Если коллекция не существует, она будет создана.

5.5.2 Параметры восстановления из загруженного снимка

5.5.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

5.5.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

5.5.2.3 Параметры запроса

wait *boolean Optional*

Если true, дождаться фактического внесения изменений. Если false — позволить изменениям происходить в фоновом режиме. По умолчанию — true.

priority *enum Optional*

Определяет источник достоверной информации для восстановления моментальных снимков.

checksum *string Optional*

Дополнительная контрольная сумма SHA256 используется для проверки целостности снимка перед восстановлением.

5.5.2.4 Запрос

Снимок после восстановления:

snapshot *file Optional*

5.5.2.5 Ответ

Успешная операция:

time *double or null*
status *string or null*
result *boolean or null*

5.6 Скачивание снимка

5.6.1 Скачивание снимка

```
curl http://localhost:6333/collections/collection_name/snapshots/snapshot_name \
    -H "api-key: <apiKey>"
```

Загружает указанный файл снимка из коллекции.

5.6.2 Параметры скачивания снимка

5.6.2.1 Аутентификация

Аутентификация по ключу API через заголовок:

api-key *string*

5.6.2.2 Параметры пути

Название коллекции:

collection_name *string Required*

Название снимка для загрузки:

snapshot_name *string Required*

5.6.2.3 Ответ

Файл снимка.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

№ версии	*Дата изменения*	*Автор*	*Состав изменений*
1.0	18.12.2025	Галкина М.Н.	Первоначальная версия документа.