



Зачем мне векторная база данных, если уже есть PostgreSQL?

8 апреля 2024

Владлен Пополитов, Олег Бартунов

Векторные базы данных в 2023 году

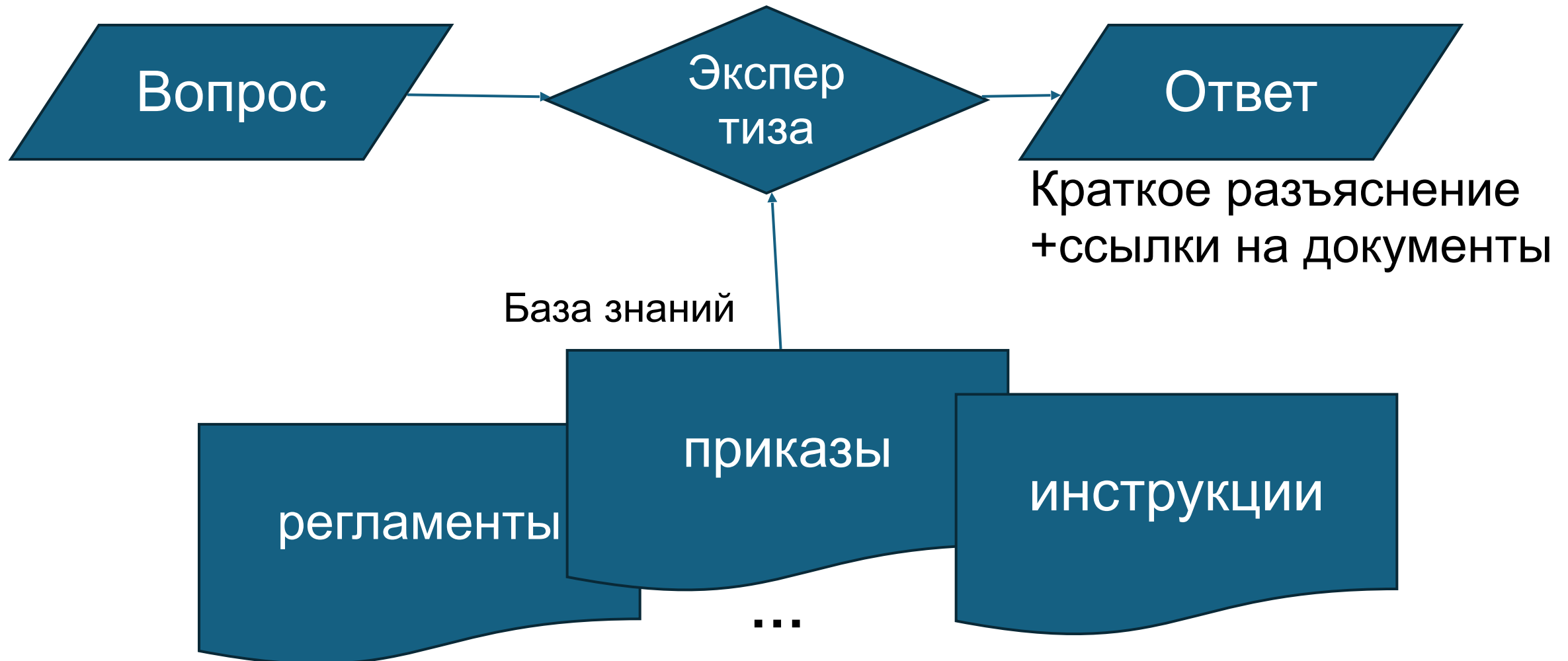
Название	Сайт	Год основания	Финансирование в 2023
Pinecone	Pinecone.io	2018	Series B \$100M
Qdrant	qdrant.tech	2021	Seed \$7.5M
Weaviate	weaviate.io	2022	Series B \$50M
Chroma	www.trychroma.com	2022	Seed \$18M
Marqo	www.marqo.ai	2022	Seed \$5.3M

Seed - первое финансирование стартапа (angel investor, бизнес-инкубатор и т.п.)

Series A - финансирование на развитие продукта, когда есть работающий продукт и несколько сотрудников

Series B - финансирование на развитие бизнеса (но уже не на развитие продукта)

Пример - консультант по корпоративной документации



Демонстрация

PostgreSQL documentation bot

User

Как завести нового пользователя базы данных?

20:48

Assistant

Ответ: Для создания нового пользователя в базе данных вы можете использовать команду 'createuser' в PostgreSQL. Эта команда позволяет определить новую учетную запись пользователя с различными опциями, такими как установка ограничений на подключения, определение ролей, установка паролей и многое другое. Только суперпользователи или пользователи с привилегией CREATEROLE могут создавать новых пользователей с помощью этой команды. Команда 'createuser' является оболочкой для SQL-команды CREATE ROLE, и нет эффективной разницы между созданием пользователей с помощью этой утилиты и другими методами доступа к серверу.

[createuser \(PostgreSQL 9.4\)](#)

Name createuser – define a new PostgreSQL user account Synopsis createuser [connection-option ..] [option ..] [username] Description createuser cre...

[CREATE USER \(Postgres Pro Standard 9.5\)](#)

CREATE USER – define a new database role Synopsis

CREATE USER name [[WITH] option [..]]

where option can be:

[SUPERUSER](#) | [NOSUPERUSER](#) | [CREATEDB](#) | [NOCR](#)...

[CREATE GROUP \(Postgres Pro Standard 9.5\)](#)

Send a message

Send

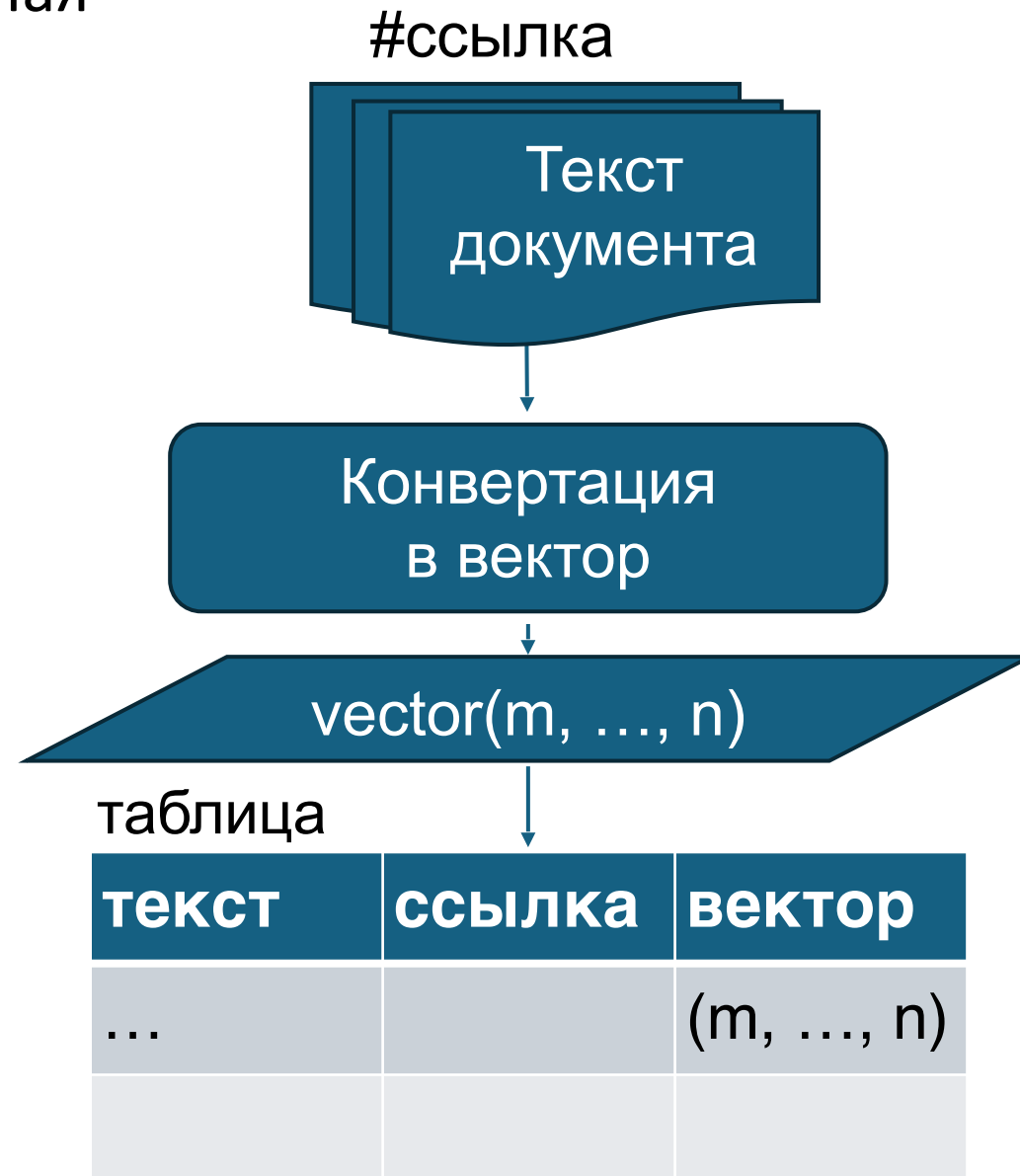
Rerun

Undo

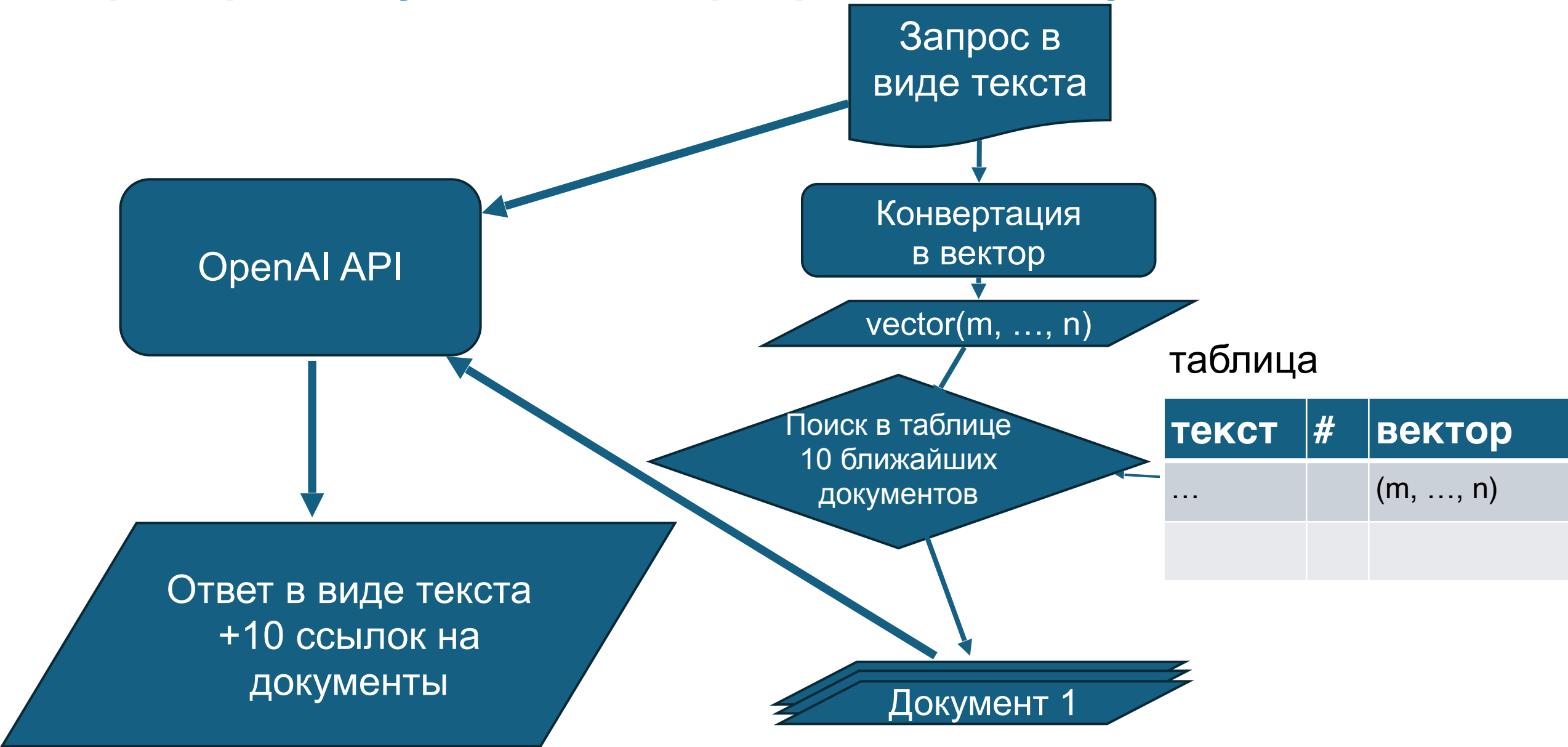
Clear

Пример - консультант по корпоративной документации

Предварительная
подготовка

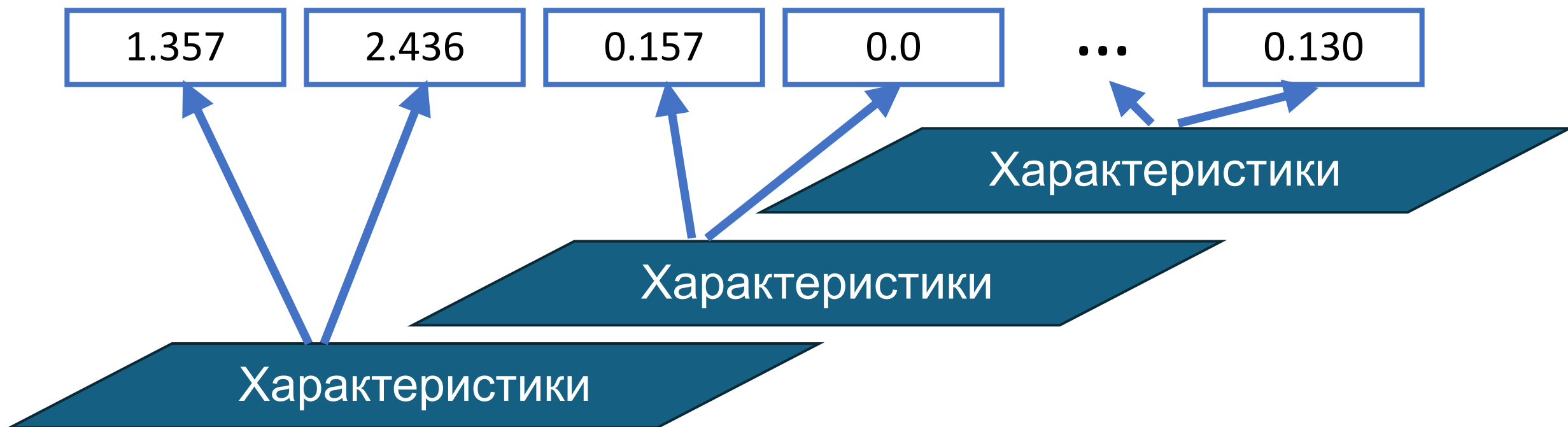


Пример - консультант по корпоративной документации



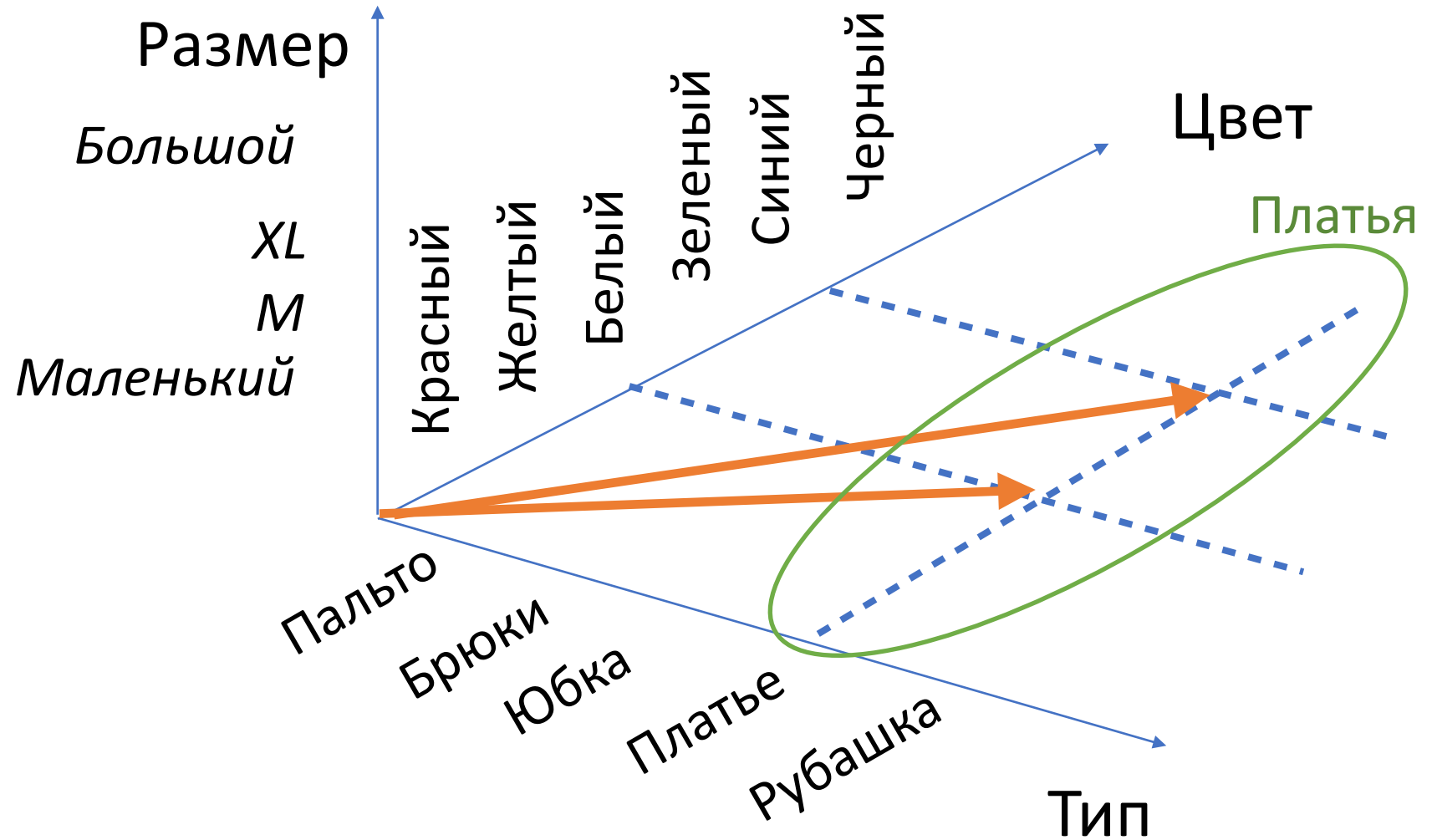
Что такое вектор

- Массив из **N** элементов типа **float4**



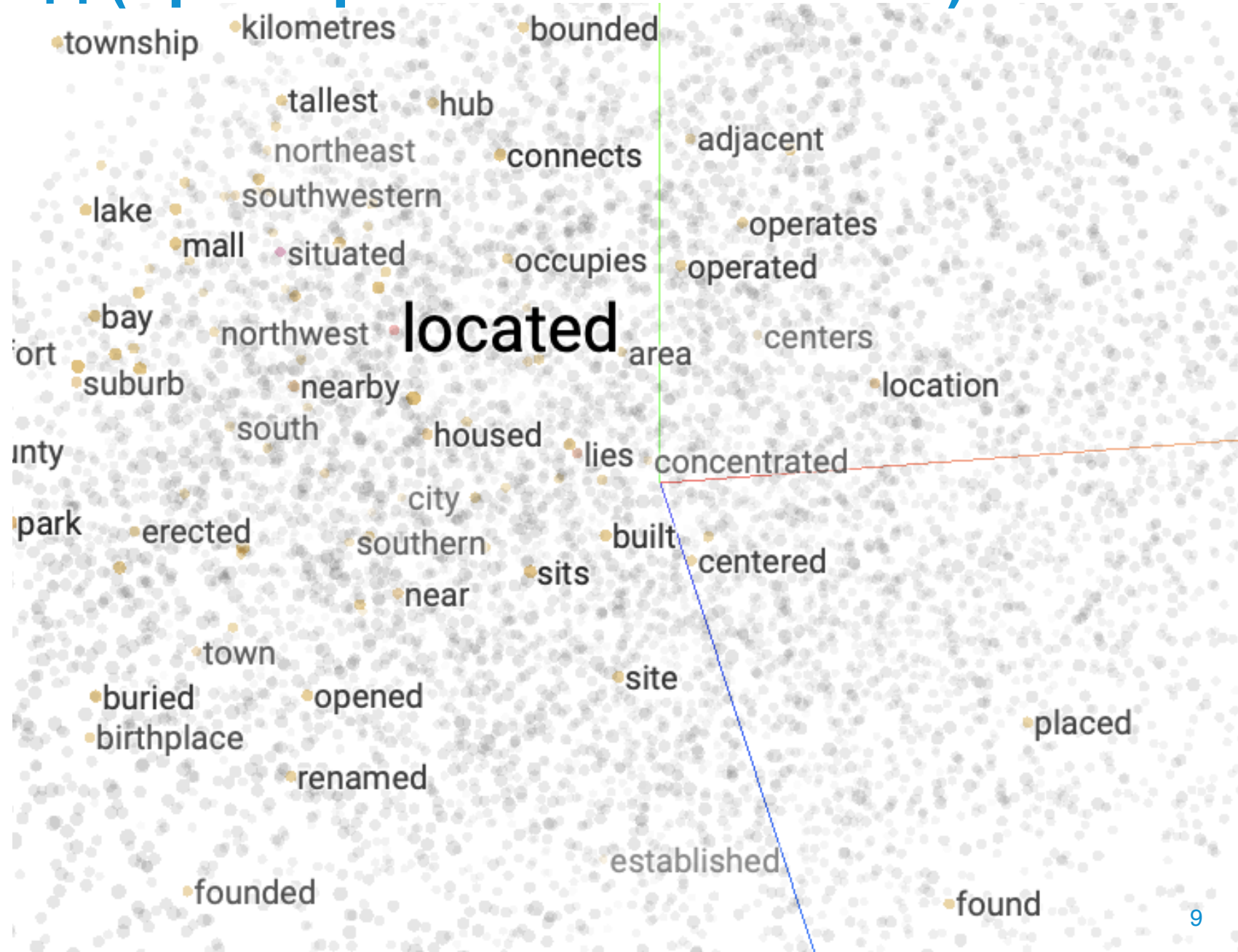
Векторный подход

- В векторе каждое измерение означает характеристику



Векторный подход (пример — значения слов)

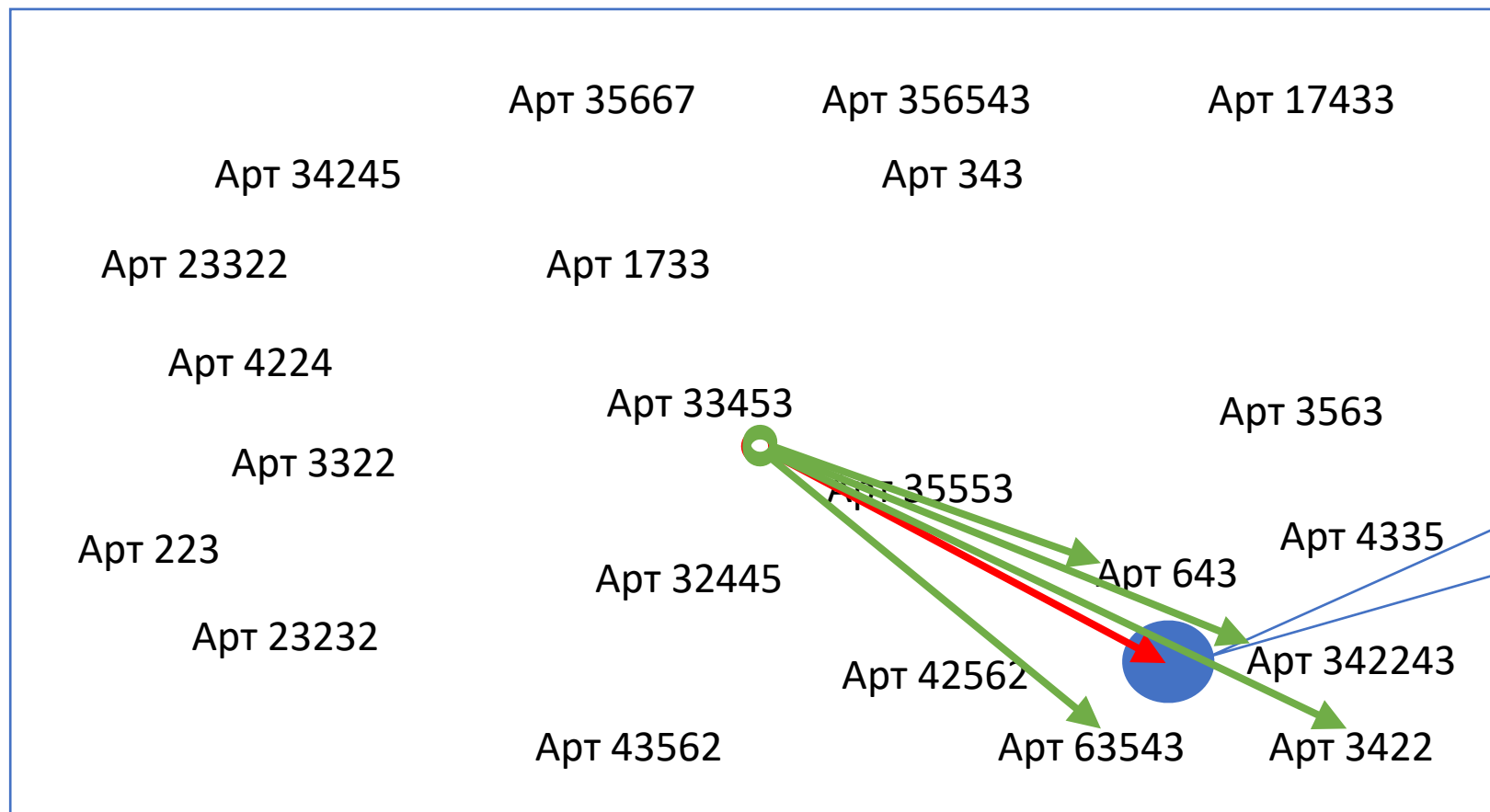
- Расчет делается программой word2vec или GloVe
- Можно делать поиск не по корням слов, а по смыслу



Векторы - поиск ближайших

По расстоянию или по направлению

Красный
Оранжевый
Желтый
Зеленый
Голубой
Синий
Темносиний
Фиолетовый

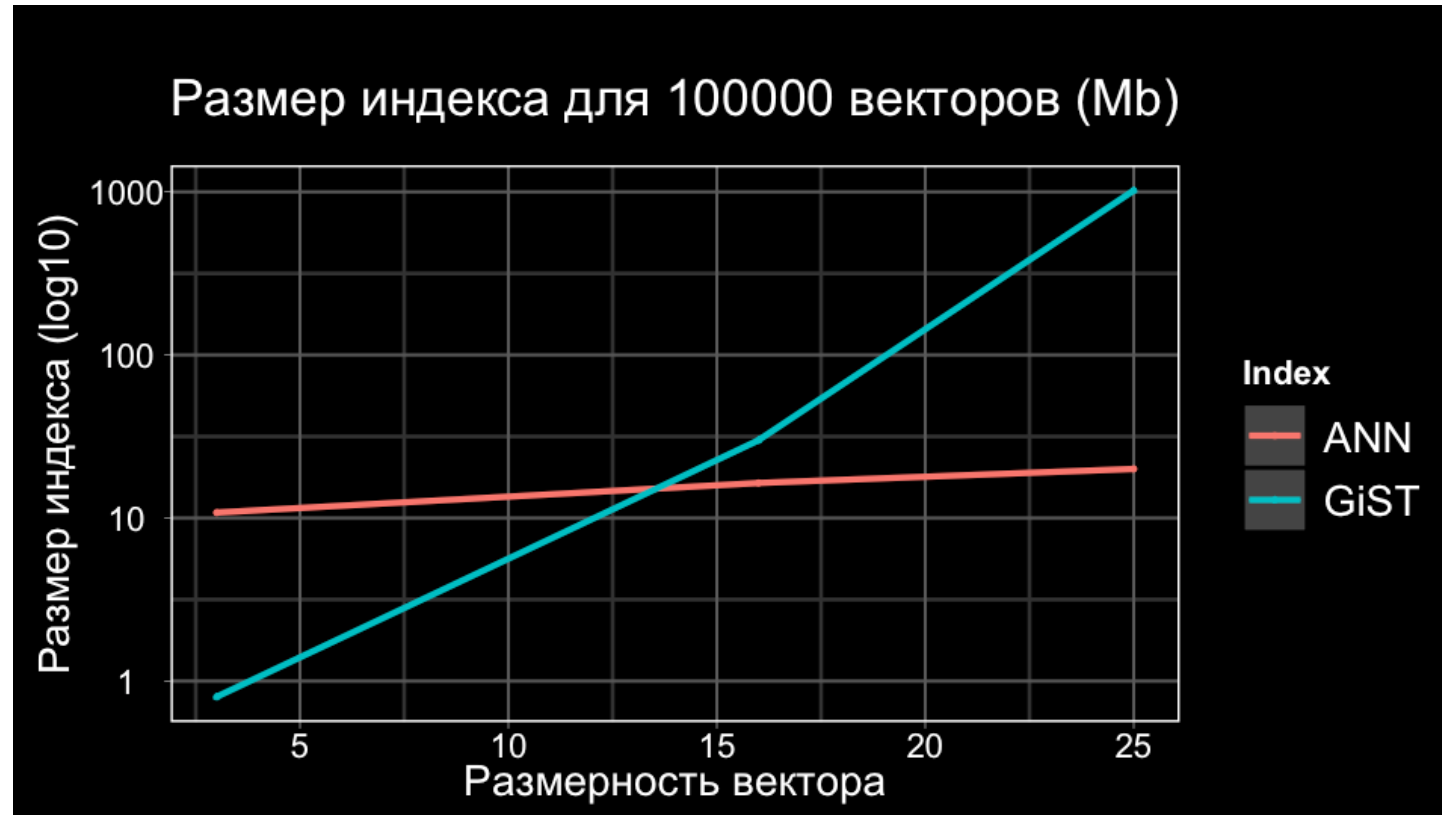


Ваши
заказы или
просмотры

Юбка Платье Блузка Пальто Рубашка Брюки Костюм Жилет

«Проклятие размерности» (1994 год)

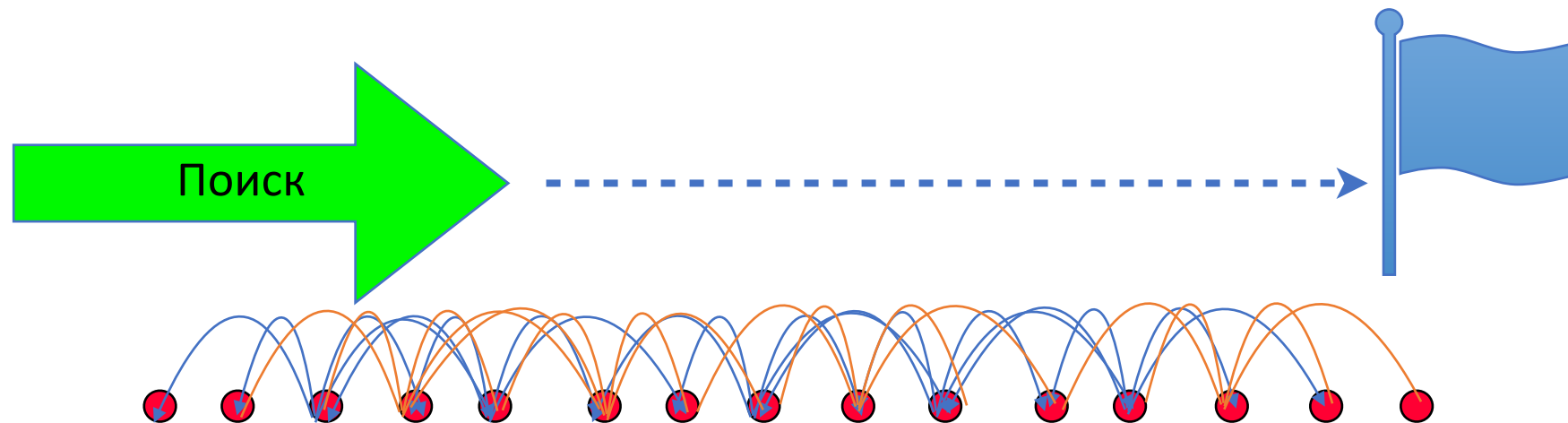
- Время поиска растет, как $N^{(2*k)}$, k размерность вектора



- Применяется поиск приближенных соседей (ANN)

Алгоритмы поиска ANN

- «Приблизительный» означает, что найденные соседи не обязательно ближайšie
- Алгоритм может хорошо работать на одних данных и очень плохо на других



- Пример: для каждого вектора вычислены ближайšie вектора
- Если вектора в базе отсортированы, то поиск потребует полного перебора
- Алгоритм должен давать сопоставимый результат на разных данных

Алгоритмы поиска ANN - примеры

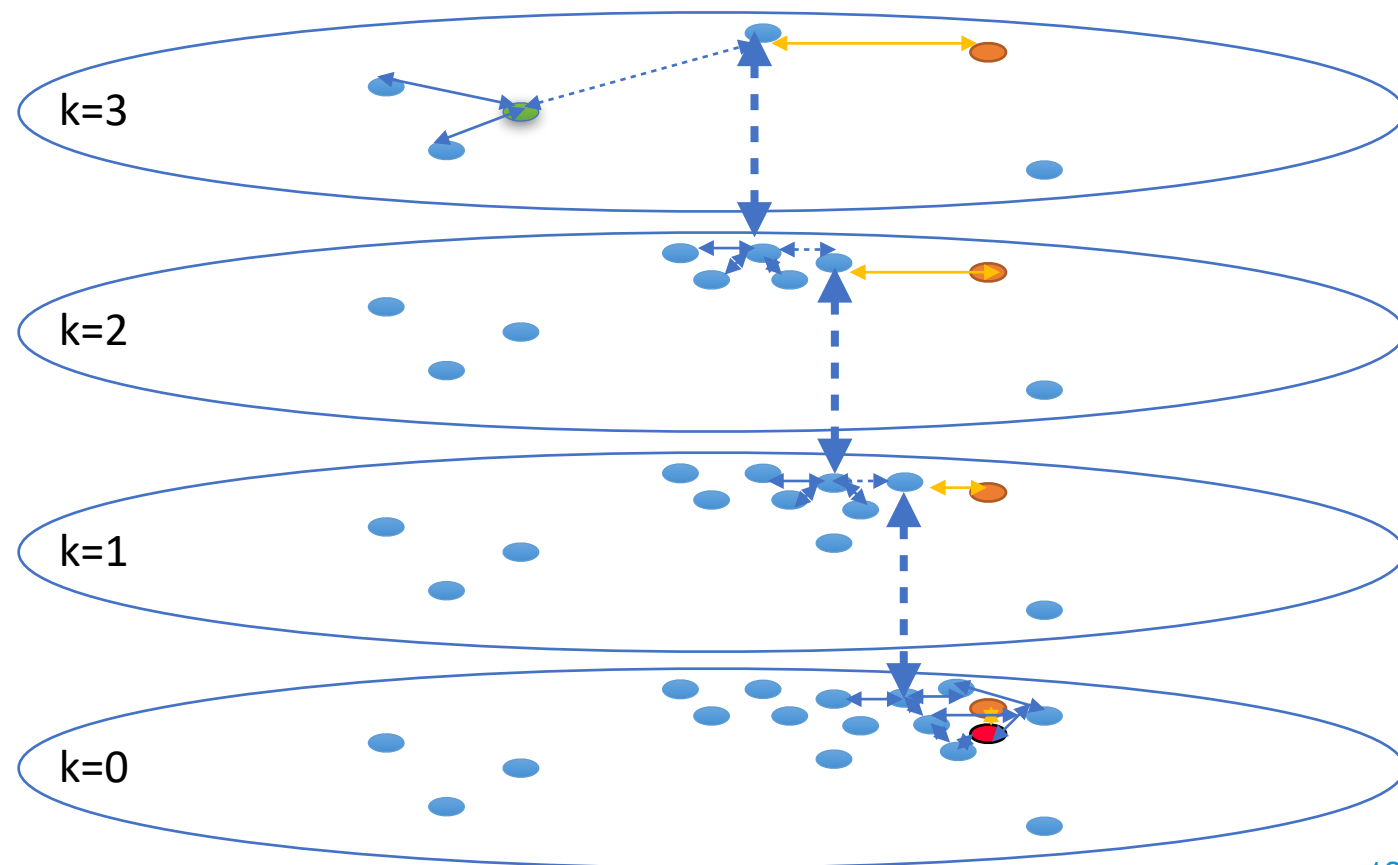
- Много статей с алгоритмами поиска ANN с конца 2000х
- Примеры алгоритмов NSW, HNSW, PQ, LSH, FLANN и много других
- Один из популярнейших HNSW: Hierarchical Navigable Small World - «иерархические миры»
- Векторы распределены по уровням с убыванием количества
- На каждом уровне вектор соединен графом с ближайшими соседями

● Точка входа

● Искомый вектор

● Найденный

● ближайший вектор



Алгоритмы поиска ANN - HNSW

- 2 параметра при построении индекса

M - количество соседей у каждой точки

ef - длина очереди кандидатов в соседи

	Величина	Скорость построения индекса	Скорость поиска	Точность поиска
M	 Больше	 Медленнее		 Больше
	 Меньше	 Быстрее		 Меньше
ef	 Больше	 Медленнее	 Быстрее	 Больше
	 Меньше	 Быстрее	 Медленнее	 Меньше

- 1 параметр при поиске

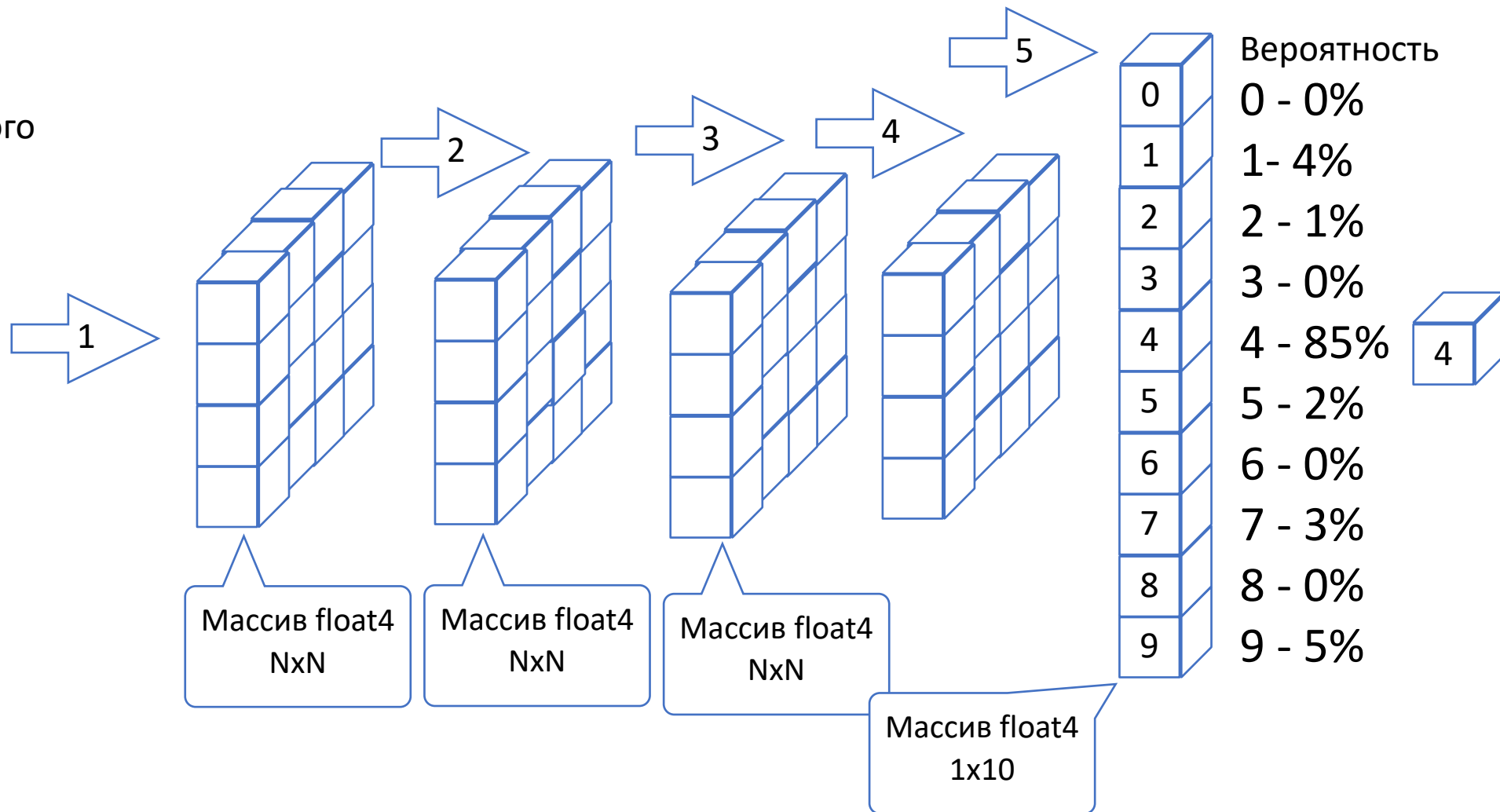
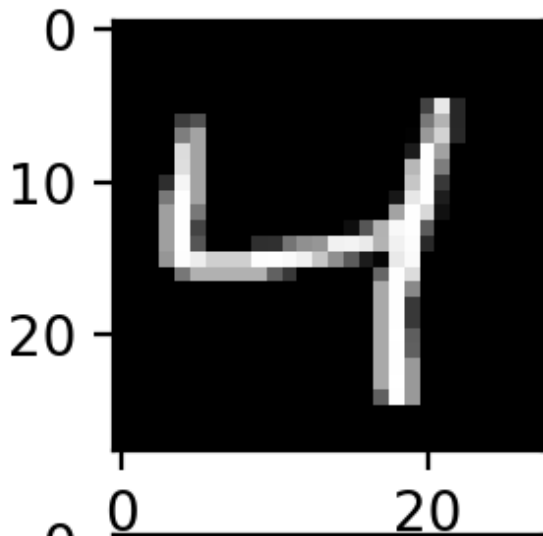
ef - длина очереди ближайших кандидатов

	Величина	Скорость поиска	Точность поиска
ef	 Больше	 Медленнее	 Больше
	 Меньше	 Быстрее	 Меньше

- Текстовый поиск по смыслу слов
- Поиск изображений
- Поиск звукозаписей (по звуковому спектру)
- Системы рекомендаций
- Приложения, отвечающие на вопросы
- Выявление аномальных отклонений в данных

Пример машинного обучения (данные MNIST)

Массив 28x28, значения
0-255 от белого до черного



- 1) Задача программиста - выбрать последовательность операций
- 2) Задача компьютера - подобрать значения коэффициентов в каждой операции, чтобы правильных результатов было больше

Что такое модель?



Расчет модели (Tensorflow, PyTorch)

Сохранить на диск

Сохранить в облако



Hugging Face

Загрузить в память

Скачать из облака



TensorFlow

Входящая информация



PyTorch

Результат (вектор)

- Модели конвертации текста в вектор (python) :

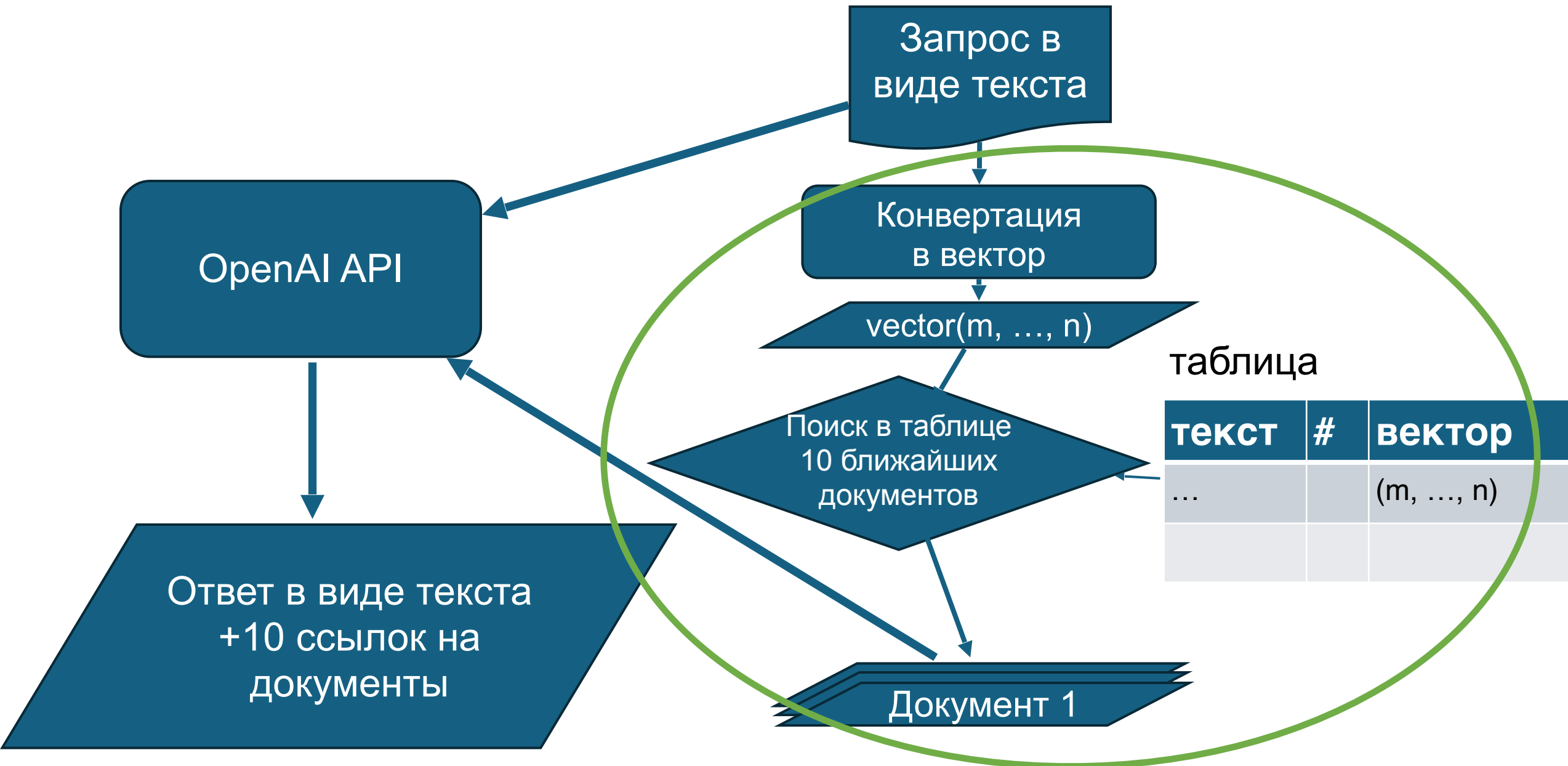
```
model = pull_from_hub(repo_id="Dimitre/universal-sentence-encoder")
```

```
vectors = model([  
    "The quick brown fox jumps over the lazy dog.",  
    "I am a sentence for which I would like to get its embedding"])
```

```
print(vectors)
```

```
# вывод векторов размерностью 512 для каждого предложения  
# вектор для: The quick brown fox jumps over the lazy dog.  
# [-0.03133016 -0.06338634 -0.01607501, ...]  
# вектор для: I am a sentence for which I would like to get its embedding.  
# [0.05080863 -0.0165243 0.01573782, ...]
```

Функции векторных баз данных



- Хранение векторов
- Поиск векторов
- Фильтры при поиске векторов
- Преобразование текста в вектор (опция)
- Бэкап и восстановление
- Собственный API (нестандартный)
 - добавить вектор
 - найти K ближайших соседей к вектору

Поиск ближайших в PostgreSQL

- Индекс GiST для поиска kNN
- Расширения для поиска по географическим координатам
- Полнотекстовый поиск
- Расширение cube (3-мерный вектор)
- Расширение ImgSmlr (16-мерный вектор)

kNN-запрос использует ключевое слово ORDER BY

```
SELECT ... FROM ... WHERE ...  
ORDER BY p <-> ' (0.0, 0.0) '::point  
LIMIT k;
```

<-> - оператор расстояния, должен быть определен для типа данных

pgvector - векторное расширение в PostgreSQL

Название	Описание с сайта	Алгоритмы	Комментарий
pgvector	Определили свой тип данных vector	HNSW, IVFFLAT	Текущая версия 0.6.2. Используется провайдерами, как стандартное векторное расширение

- Создание индекса

```
CREATE INDEX idx_name  
ON tablename  
USING hnsw  
(vec l2_opclass) WITH (m=20, efconstruction=100)
```

- Запрос ближайших 10 векторов

```
SELECT id FROM tablename  
ORDER BY vec <=> '[1,4,2,6,4,2,5,2]':vector  
LIMIT 10
```

pgvector - плюсы и минусы

- + два алгоритма (быстрый с индексом большого размера, медленный с большей точностью и с меньшим индексом)
- + стабильный код
- + приложены большие усилия на оптимизацию реализации алгоритмов
- может не выдать строки, если в запросе есть условие WHERE
- нет планов добавления алгоритмов
- тип vector встроен в расширение (поля с данными удаляются вместе с расширением)
- каждый алгоритм использует свой access method

Timescale - векторное расширение diskann

- Алгоритм с уменьшением размерности
- Индекс меньшего размера
- Увеличение скорости при уменьшении точности

GANN - векторное расширение в PostgreSQL

GANN - Generalised Approximate Nearest Neighbour

Один метод доступа GANN

Детали создания метода доступа - в коде GANN

Детали реализации алгоритмов - в функции оператора класса

Для подключения алгоритма необходимо написать функции Build, Scan, Insert, Vacuum

```
CREATE OPERATOR CLASS gann_vector
  DEFAULT FOR TYPE vector USING gann AS
  OPERATOR 1 <-> (vector, vector) FOR ORDER BY float_ops,
  FUNCTION 1 gann_config(internal),
  FUNCTION 2 gann_build(internal, vector),
  FUNCTION 3 gann_distance(vector, vector),
  FUNCTION 4 gann_norm(vector),
  FUNCTION 5 gann_scan(internal, vector),
  FUNCTION 6 gann_insert(internal, vector),
  FUNCTION 7 gann_vacuum(internal, internal, internal)
;
```

GANN - векторное расширение в PostgreSQL

GANN - Generalised Approximate Nearest Neighbour

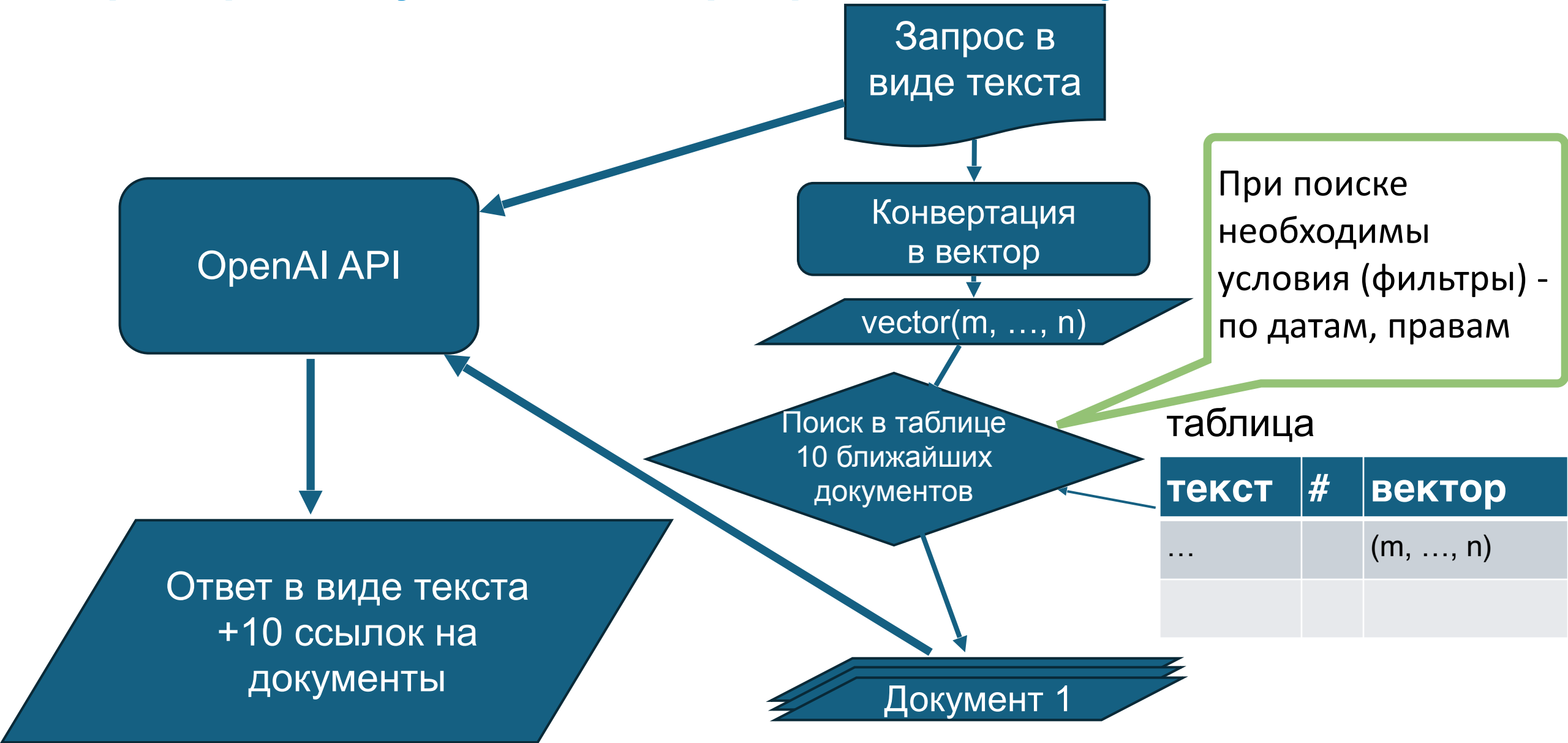
- Создание индекса

```
CREATE INDEX idx_name  
ON tablename  
USING gann  
(vec gann_vector) WITH (dim=8)
```

- Запрос ближайших 10 векторов

```
SELECT id FROM tablename  
ORDER BY vec <-> '[1,4,2,6,4,2,5,2]':vector  
LIMIT 10
```

Пример - консультант по корпоративной документации PosgresPro



GANN и pgvector - условие WHERE

pgvector

```
# CREATE EXTENSION vector;  
CREATE EXTENSION  
# SELECT t,  
Array[t,t+1,t+2]::vector(3) AS val  
INTO test_where  
FROM generate_series(1, 2000) as t;  
SELECT 2000  
# CREATE INDEX ON test_where USING hnsw  
(val vector_l2_ops) WITH (m=10);  
CREATE INDEX  
# SELECT t FROM test_where WHERE t>1500  
ORDER BY val <-> '[1000.2,1000,1000.1]'  
LIMIT 2;
```

t

(0 rows)

GANN (hnsw_vbase)

```
# CREATE EXTENSION hnswbase CASCADE;  
CREATE EXTENSION  
# SELECT t,  
Array[t,t+1,t+2]::ganntype(3) AS val  
INTO test_where  
FROM generate_series(1, 2000) as t;  
SELECT 2000  
# CREATE INDEX ON test_where USING gann  
(val hnsw_vbase) WITH (m=10, dim=3);  
CREATE INDEX  
# SELECT t FROM test_where WHERE t>1500  
ORDER BY val <-> '[1000.2,1000,1000.1]'  
LIMIT 2;
```

t

1501

1502

(2 rows)

- Поддержка в алгоритме условий в WHERE
- Поддержка многоколоночных индексов (для поддержки WHERE)
- Добавление новых алгоритмов

Векторные базы vs реляционные базы

	Векторные	Реляционные
Скорость	Быстрее	Медленнее
Транзакции и целостность данных	Нет	Есть
Стандартный API	Нестандартный	SQL
Бэкап и мониторинг	Да	Да
Доступ пользователей	Да	Да
Использование с другими данными	Нет	Да

1. PostgreSQL уже сейчас поддерживает работу с векторами
2. PostgresPro предлагает GANN для расширения возможностей
3. В GANN планируем :
 - А. добавление новых алгоритмов
 - В. многоколоночные индексы для улучшения фильтрации данных

PosgresPro

