



УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Облачные и туманные вычисления

Лабораторная работа №3

Автоматизированное создание индексов БД на основе существующих запросов

Преподаватель: Перл Ольга Вячеславовна

Выполнил: Кульбако Артемий Юрьевич Р34112

Оглавление

Оглавление	2
Концепт	3
Поставщик облачных услуг	3
Компонентная схема	3
UseCases	4
API	6
BPMN	8
Тестовая база данных	8
UML-приложения	12
Используемые ресурсы	12

Концепт

Приложение, которое автоматически создаёт наилучшие индексы в базе данных на основе используемых запросов к существующей БД.

Идея подобного приложения выросла из настоящей задачи с работы, когда появилась необходимость ускорить очень сложный (более 50 строк кода) и очень долгий (более минуты выполнения) запрос к **MongoDB**. Надёжнейшим и быстреешим вариантом оказалось создание **Python**-скрипта, который сам переберёт все возможные комбинации индексов для полей и таблиц, фигурируемых в запросе, и выберет наиболее оптимальный по скорости работы и/или памяти.

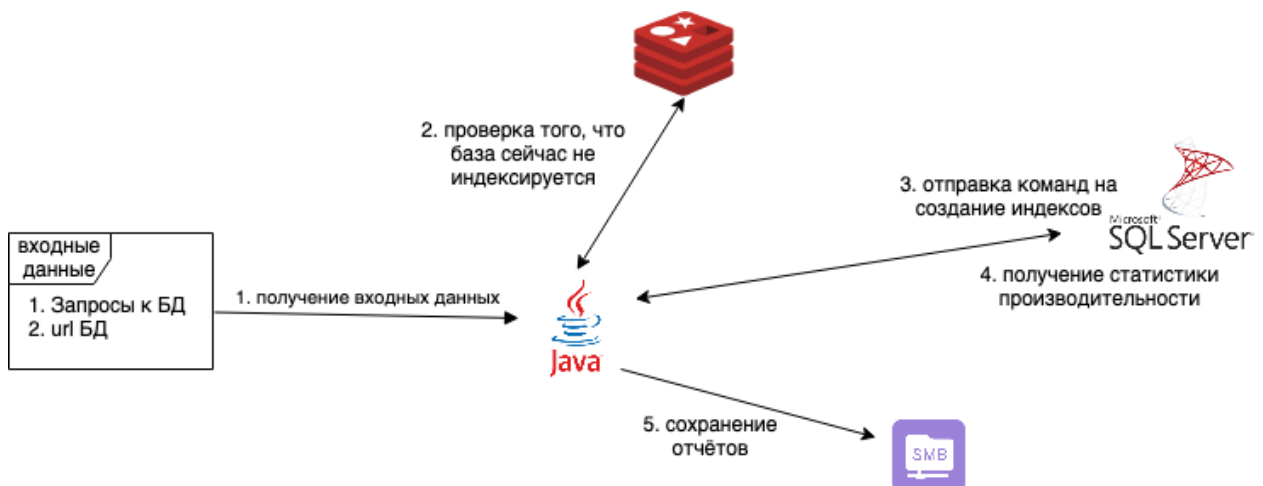
Стоит учитывать, что сам сервис ничего не хранит - он получает на вход данные от клиента, модифицирует предоставленную БД и отдаёт отчёт о проделанной работе. Т.е. БД и файловое хранилище здесь используются для симуляции работы с сервисом, а не являются его частями.

Поставщик облачных услуг

В качестве провайдера "as service" было решено выбрать **Microsoft Azure** по ряду причин:

1. Подробная документация на [Microsoft Docs](#).
2. Студенческий аккаунт не имеет ограничений: вам представляется 15000руб на любые существующие в **Azure** услуги и сервисы.
3. Есть консольный клиент, позволяющий быстро и удобно управлять сервисами, автоматизировать их.
4. Интерфейс на русском языке (так намного приятнее работать).

Компонентная схема



API и бизнес-логика будут написаны на **Kotlin**, соответственно, работать в **JVM**. Для этого использует **Azure**-ресурс Веб-приложение.



Веб-приложение
[Создать](#) | [Документы](#) | [MS Learn](#)

Базу данных можно получить двумя способами - создать "Сервер базы данных" - т.е. приложение базы данных, где можно будет вручную создать логическую БД, либо совсем абстрактно - сразу получить логическую БД для хранения. Этим мы и воспользуемся (опытным путём можно узнать, что сервером БД является **Microsoft SQL Server**).



SQL Database
[Создать](#) | [Документы](#) | [MS Learn](#)

Для хранения сгенерированных отчётов (т.е. для файлового хранилища) воспользуюсь "[Файлами Azure](#)" - по сути, виртуальным жёстким диском с доступом по протоколу SMB.



Учетная запись хранения
[Создать](#) | [Документы](#) | [MS Learn](#)

Для избежания ситуаций, когда несколько запросов захотят индексировать одну и ту же БД нам надо хранить информацию о происходящих в данный момент операциях, в идеале, это делать надо где-то во вне, чтобы при разворачивании нескольких инстансов сервиса шарить это информацию. Для этого нам подойдёт ресурс "[Кэш Redis](#)".



Кэш Redis
[Создать](#) | [Документы](#) | [MS Learn](#)

UseCases



Имя прецедента:	Ввод кредов БД
ID:	1
Описание:	Ввод данных (url, username, password) для подключения к удалённой БД
Акторы:	Пользователь
Основной поток:	- (Пользователь запустил приложение)
Альтернативный поток:	Вывести сообщение о ошибке подключения

Имя прецедента:	Ввод через файл
ID:	2а

Описание:	Передача программе текстового файла с SQL-запросами, которые необходимо оптимизировать
Акторы:	Пользователь
Основной поток:	из 1
Альтернативный поток:	Вывести сообщение о ошибке (несуществующий путь или структура файла не соответствует требуемой)

Имя прецедента:	Ввод с клавиатуры
ID:	2b
Описание:	Ввод серии SQL-запросов, которые необходимо оптимизировать с клавиатуры
Акторы:	Пользователь
Основной поток:	из 1
Альтернативный поток:	Вывод ошибки о неверном запросе

Имя прецедента:	Получить результаты на почту
ID:	3a
Описание:	Получить результаты на введённую пользователем почту
Акторы:	Пользователь
Основной поток:	из 2a или 2b
Альтернативный поток:	Указать на некорректный почтовый адрес

Имя прецедента:	Получить результаты HTTP-ответом
ID:	3b
Описание:	Ожидать выполнения операции и получить результат по HTTP
Акторы:	Пользователь
Основной поток:	из 2a или 2b
Альтернативный поток:	Ошибка интернет-соединения

Имя прецедента:	Сохранить на SMB-диск
ID:	3c
Описание:	Сохранить отчёт на SMB-диск пользователя
Акторы:	Пользователь
Основной поток:	из 2a или 2b

Альтернативный поток:	Выдать ошибку
-----------------------	---------------

API

Описание API сервиса согласно стандарту OpenAPI 3.0:

openapi: 3.0.3

info:

title: SQL-OPTIMIZER

version: 1.0.0

servers:

– **url:** https://sql-optimizer.azurewebsites.net/

components:

schemas:

TestParams:

type: object

properties:

dbUrl:

type: string

required: true

saveBetter:

type: boolean

required: false

queries:

type: array

required: true

items:

type: string

outputMode:

type: string

nullable: true

enum:

– EMAIL

– HTTP

– SMB

TestResult:

type: object

properties:

bestIndexes:

```

    type: string
    required: true
  timeBefore:
    type: integer
    required: true
  timeAfter:
    type: integer
    required: true
  diff:
    type: number
    format: double
    required: true
paths:
  /actions/:
    post:
      summary: Tests SQL-queries for speed with all possible indexes and returns a
report.
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/TestParams'
      responses:
        200:
          description: Returns result of autoindexing.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TestResult'
            text/csv:
              schema:
                $ref: '#/components/schemas/TestResult'
        400:
          description: Bad request.
          content:
            text/plain:

```

```

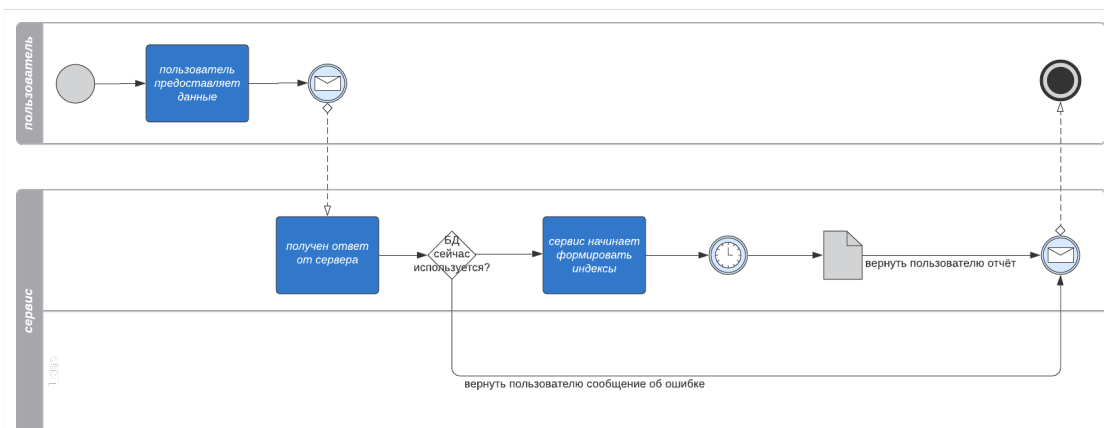
    schema:
      type: string

409:
  description: Database is busy.
  content:
    text/plain:
      schema:
        type: string

5XX:
  description: Unexpected error.

```

BPMN



Тестовая база данных

Для тестирования функциональности сервиса воспользуемся моделью БД, которая была создана мною в рамках курсовой работе по "Информационным системам и базам данных" на 3-ем курсе - модель менеджмента частной военной компании. Эта БД содержит немало таблиц, а также уже готов генератор случайных записей для неё.

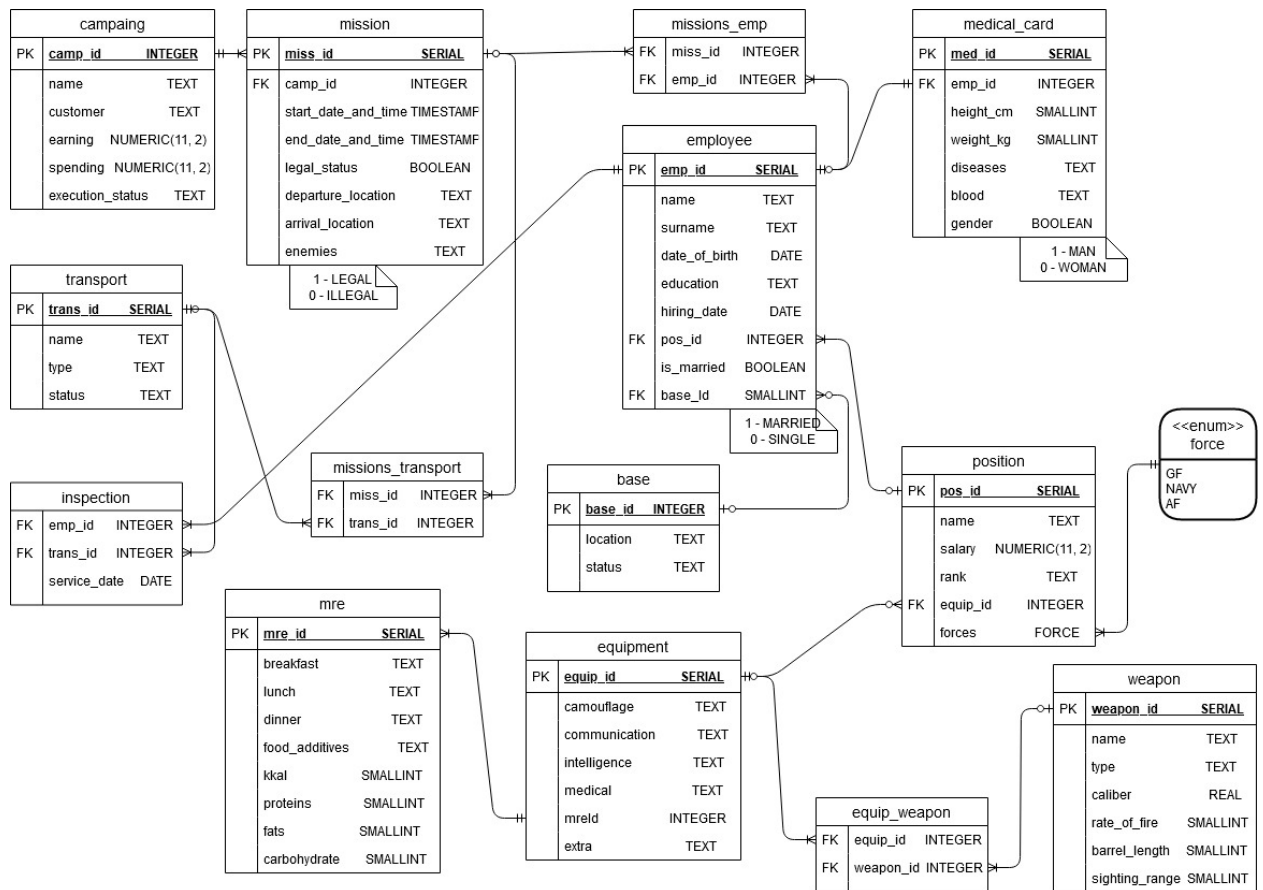
Помимо самих таблиц, будет существовать ряд функций и триггеров, основанных на бизнес-процессах информационной системы.

--1: Тех, кто не имеет воинских званий, нельзя отправлять на боевые миссии.

```

CREATE FUNCTION check_is_military_on_mission() RETURNS trigger AS $$
    DECLARE enemy TEXT;
    DECLARE emp_rank TEXT;
    BEGIN
        enemy = (SELECT enemies FROM mission WHERE miss_id = new.miss_id);
        emp_rank = (SELECT rank FROM position JOIN employee USING (pos_id) WHERE emp_id =
new.emp_id);
        IF enemy IS NOT NULL AND emp_rank IS NULL THEN

```

```

RAISE EXCEPTION 'Cannot set not military employee to a combat mission';

ELSE RETURN new;

END IF;

END;

$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER check_is_military_on_mission BEFORE INSERT ON missions_emp
FOR EACH ROW EXECUTE PROCEDURE check_is_military_on_mission();

```

/*

2: Информационная система должна учитывать какие сотрудники отправились на миссии (один и тот же сотрудник не

может находиться на двух миссиях одновременно).

*/

```

CREATE FUNCTION check_periods_of_emp_missions() RETURNS trigger AS $$

```

```

DECLARE start_time TIMESTAMPT;

```

```

DECLARE end_time TIMESTAMPT;

```

```

BEGIN

```

```

    SELECT start_time, end_time INTO start_time, end_time FROM mission WHERE miss_id =
new.miss_id;

```

```

    IF (TRUE) IN (

```

```

        SELECT (start_time, end_time) OVERLAPS
            (start_date_and_time, end_date_and_time) FROM mission
            WHERE miss_id IN (SELECT miss_id FROM missions_emp WHERE emp_id = new.emp_id))
    THEN
        RAISE EXCEPTION 'This worker cannot be assigned to a mission as he was on another
mission at the time';
    ELSE RETURN new;
    END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_emp_mission_period BEFORE INSERT ON missions_emp
    FOR EACH ROW EXECUTE PROCEDURE check_periods_of_emp_missions();

CREATE INDEX mission_period ON mission USING btree(start_date_and_time, end_date_and_time);

--3: Работников неподходящих по физическим данным запрещено устраивать как военных сотрудников
(рост < 150 см или вес < 45 кг).

CREATE FUNCTION check_physical_condition() RETURNS trigger AS $$
    DECLARE h SMALLINT;
    DECLARE w SMALLINT;
    BEGIN
        SELECT height_cm, weight_kg INTO h, w FROM medical_card JOIN employee USING (emp_id)
        WHERE emp_id = new.emp_id;
        IF h < 150 OR w < 45 THEN
            RAISE EXCEPTION 'Cannot hire this employee to military position because his physical
data does not require the minimum';
        ELSE RETURN new;
        END IF;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_physical_condition BEFORE INSERT ON employee
    FOR EACH ROW EXECUTE PROCEDURE check_physical_condition();

/*
4: Необходимо хранить историю инспекций транспорта (реализована отдельной таблицей),
а транспорт со статусами «сломан» или «в ремонте» нельзя использовать в операциях.
*/

CREATE FUNCTION check_transport_condition() RETURNS trigger AS $$
    BEGIN

```

```

        IF (SELECT status FROM transport WHERE trans_id = new.trans_id AND status = 'available')
IS NULL THEN

        RAISE EXCEPTION 'Cannot set not available transport to mission';

        ELSE RETURN new;

        END IF;

    END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER check_transport_condition BEFORE INSERT ON missions_transport
    FOR EACH ROW EXECUTE PROCEDURE check_transport_condition();

-- 5: Если за базой не закреплён ни один сотрудник, стоит закрыть её.
CREATE FUNCTION close_empty_bases() RETURNS SETOF void AS $$
    BEGIN
        DELETE FROM base WHERE base_id IN (SELECT * FROM base_count_emp);
    END;
$$ LANGUAGE plpgsql;

CREATE MATERIALIZED VIEW base_count_emp AS
    (SELECT base_id FROM base JOIN employee USING (base_id) GROUP BY base_id HAVING COUNT(emp_id)
= 0);

CREATE FUNCTION update_base_count_emp() RETURNS trigger AS $$
    BEGIN
        REFRESH MATERIALIZED VIEW base_count_emp;
        RETURN new;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER update_base_count_emp AFTER INSERT OR UPDATE OR DELETE ON employee
    FOR EACH ROW EXECUTE PROCEDURE update_base_count_emp();

/*
6: Стараться отправлять на боевые операции при прочих равных в первую очередь неженатых военных,
давно не
    участвовавших в миссиях, имеющих большой опыт работы.
*/

CREATE FUNCTION get_combat_candidates(n int DEFAULT 1) RETURNS employee AS $$
    BEGIN
        SELECT emp_id FROM employee

```

```

JOIN position USING (pos_id)

JOIN missions_emp USING (emp_id)

JOIN mission USING (miss_id)

WHERE rank IS NOT NULL OR !~~ ''

ORDER BY is_married DESC, end_date_and_time DESC, hiring_date DESC

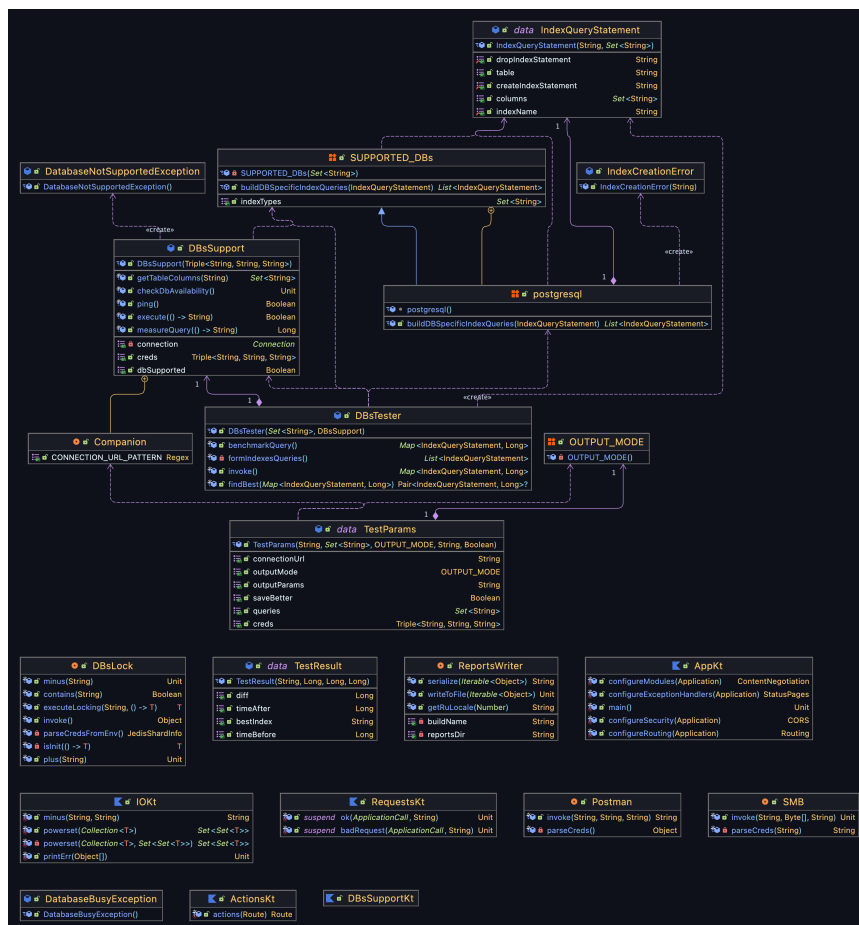
LIMIT n;

END;

$$ LANGUAGE plpgsql;

```

UML-приложения



Используемые ресурсы

Имя	Тип	Последний просмотр
foggy	Учетная запись хранения	5 мин назад
foggy	Служба приложений	8 мин назад
foggy	Кэш Azure для Redis	18 мин назад
cloud_labs	Группа ресурсов	47 мин назад
foggy	Гибкий сервер Базы данных Azure для PostgreSQL	6 ч назад
Azure for Students	Подписка	день назад