Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Отчёт по лабораторным работам №1-4

Дисциплина: Базы данных

Выполнил студент гр. 3530901/70203 _____ А.А. Ворошилов
(подпись)

Преподаватель _____ А.В. Мяснов
(подпись)

“___”_____ 2021 г.

Санкт-Петербург
2021

1. Лабораторная работа №1. Проектирование БД

Цели работы

Познакомиться с основами проектирования схемы БД, способами организации данных в SQL-БД.

Программа работы

- Создание проекта для работы в GitLab.
- Выбор задания (предметной области), описание набора данных и требований к хранимым данным в свободном формате в wiki своего проекта в GitLab.
- Формирование в свободном формате (предпочтительно в виде графической схемы) схемы БД, соответствующей заданию. Должно получиться не менее 7 таблиц.
- Согласование с преподавателем схемы БД. Обоснование принятых решений и соответствия требованиям выбранного задания.
- Выкладывание схемы БД в свой проект в GitLab.
- Демонстрация результатов преподавателю.

- Самостоятельное изучение SQL-DDL.
- Создание скрипта БД в соответствии с согласованной схемой. Должны присутствовать первичные и внешние ключи, ограничения на диапазоны значений. Демонстрация скрипта преподавателю.
- Создание скрипта, заполняющего все таблицы БД данными.
- Выполнение SQL-запросов, изменяющих схему созданной БД по заданию преподавателя. Демонстрация их работы преподавателю.

Выполнение работы

В качестве предмета проекта была выбрана база данных шпионского агенства:

1) person

- Информация личного характера о всех работниках агенства

```
CREATE TABLE IF NOT EXISTS person
(
    person_id serial      not null
        constraint person_file_pk
            primary key,
    bio       text,
    name      varchar(50) not null
);
```

2) unit_profile

- Кратко говоря, профессиональное досье/портфолио.
- Информация и легенда текущего и прошедших рангов, которые были у работника (какие звания в какой период носил, какие навыки способствовали ношению такого звания).

```
CREATE TABLE IF NOT EXISTS unit_profile
(
    unit_profile_id serial  not null
        constraint unit_profile_pk
            primary key,
    info            text,
    rank            integer not null,
    person_id       integer not null
        constraint unit_profile_person_person_id_fk
            references person
            on update cascade on delete cascade,
    date_from       date    not null,
    date_to         date,
    agent_id        integer
        constraint unit_profile_agent_agent_id_fk_2
            references agent
            on update cascade on delete cascade,
    operator_id     integer
        constraint unit_profile_operator_operator_id_fk
            references operator
            on update cascade on delete cascade
);
```

3) med_record

- Состояние здоровья работника в определенный период.

```
CREATE TABLE IF NOT EXISTS med_record
(
    med_record_id serial      not null
        constraint status_pk
            primary key,
    title         varchar(30) not null,
```

```
    info          text,
    person_id     integer     not null
        constraint med_record_person_person_id_fk
            references public.person
            on update cascade on delete cascade,
    date_from     date        not null,
    date_to       date
);
```

4) operator

- Работники такого вида, которые руководят миссией или координируют её.

```
CREATE TABLE IF NOT EXISTS operator
(
    operator_id serial               not null
        constraint person_pk
            primary key,
    info        text,
    available   boolean default true not null
);
```

5) mission

- Миссия, её состояние, назначенный оператор, и описание.
- mission_status является enum со следующими значениями: 'PLANNING', 'STARTING', 'PERFORMING', 'CANCELLING', 'FINISHED').

```
CREATE TABLE IF NOT EXISTS mission
(
    mission_id     serial
not null
        constraint mission_type_pk
            primary key,
    name           varchar(50)
not null,
    rank           integer
not null,
    info           text,
    operator_id    integer
        constraint mission_operator_operator_id_fk
            references public.operator
            on update cascade on delete set null,
    mission_status public.mission_status default
'PLANNING'::public.mission_status not null
);
```

6) pack

- Такой набор вещей (множество item), который берёт с собой на задание один агент.

```
CREATE TABLE IF NOT EXISTS public.pack
(
    pack_id serial      not null
        constraint pack_pk
            primary key,
```

```
    name    varchar(50) not null,
    info    text
);
```

7) item

- Любая вещь, костюм, гаджет, оружие. Может и человек, не работающий в текущем агенстве(заложник, переводчик).

```
CREATE TABLE IF NOT EXISTS item
(
    item_id serial      not null
        constraint item_pk
            primary key,
    name    varchar(50) not null,
    info    text,
    pack_id integer     not null
        constraint item_pack_pack_id_fk
            references public.pack
            on update cascade on delete set null
);
```

8) agent

- Агент, у которого имеется звание, ссылка на текущее портфолио, а также отображается доступность найма на текущий момент. (Например, если агент болеет, то available должно быть присвоено false)

```
CREATE TABLE IF NOT EXISTS agent
(
    agent_id  serial               not null
        constraint agend_bio_pk
            primary key,
    name      varchar(50)          not null,
    available boolean default true not null,
    pack_id   integer
        constraint agent_pack_id_fkey
            references pack
);
```

9) agent_mission

- Отношение агента и миссии, описывающее на какой период был нанят агент на задание, а также дополнительная информация про выполненное содействие. По факту, история операций у агента.

```
CREATE TABLE IF NOT EXISTS agent_mission
(
    agent_mission_id serial  not null
        constraint agent_mission_pk
            primary key,
    agent_id         integer not null
        constraint agent_mission_agents_agent_id_fk
            references public.agent
            on update cascade,
    mission_id       integer not null
        constraint agent_mission_mission_mission_id_fk
            references public.mission
            on update cascade on delete cascade,
```

```
    info             text,
    date_from        date    not null,
    date_to          date
);
```

10) mission_result

- Результат выполнения миссии. Результатов может быть много, например, в поэтапных миссиях.

```
CREATE TABLE IF NOT EXISTS mission_result
(
    mission_result_id serial                    not null
        constraint mission_result_pk
            primary key,
    mission_id        integer                   not null
        constraint mission_result_mission_mission_id_fk
            references public.mission
            on update cascade on delete cascade,
    info              text,
    time              varchar(30) default now() not null
);
```

11) loot

- Информация о вещах или информации, добытой на опредённой стадии миссии (mission_result). loot (англ. добыча), также может указывать на человека: преступник, спасённая жертва, привлечённый двойной агент

```
CREATE TABLE IF NOT EXISTS loot
(
    loot_id           serial      not null
        constraint loot_pk
            primary key,
    mission_result_id integer     not null
        constraint intel_mission_result_mission_result_id_fk
            references public.mission_result
            on update cascade on delete cascade,
    rank              integer     not null,
    info              text,
    name              varchar(50) not null
);
```

Скрипт заполнения БД данными:

```
INSERT INTO public.person (bio, name)
	VALUES
	('1985, Portsmooth, England, fem.', 'Lucy Landa'),
	('1987, Kong Hong, China, mal.', 'Wei Shen'),
	('1983, Ryazan, USSR', 'Dmitry Tachankin'),
	('1990, Poland', 'Dober Lodvinof');

INSERT INTO public.unit_profile (info, person_id, date_from, date_to)
	VALUES
	('Good coordinator, knows 7 languages:..., the most young in
agency', 1, '2006-05-15','2008-05-15'),
	('Excellent coordinator, knows 9 languages:...', 1, '2008-05-15',
null),
	('Excellent integorator, knows 3 languages:...', 2, '2009-05-15',
null),
	('Heavy weapons and explosives expert, 2 languages:...', 3, '2010-
05-15', null),
	('Stealth expert, agent Colossus eliminated during mission "Sunset
2020-07-23"', 4, '2009-05-15', '2020-07-23');

INSERT INTO public.med_record (title, info, person_id, date_from, date_to)
	VALUES
	('Pneumonia', 'hard case', 1, '2010-05-15','2010-06-25'),
	('KIA', 'Bullet shot, blood loss', 4,'2020-03-20','2020-03-20');

INSERT INTO public.operator (info, unit_profile_id, available)
	VALUES
	('operations manager', 1, false);

INSERT INTO public.mission (name, rank, info, operator_id, mission_status)
	VALUES
	('Electric Vision', 3, 'Eliminate the general', 1, 'FINISHED'),
	('Rising Storm', 3, 'Rescue the diplomat', 1, 'PERFORMING'),
	('Sunset', 3, 'Lenetti mafia undercover', 1, 'FINISHED');
INSERT INTO public.pack (name, info)
	VALUES
	('Civilian disguise', 'civil concealment + hidden pistol'),
	('Killer loadout', 'handheld rifle + high tech jacket');

INSERT INTO public.item (name, info, pack_id)
	VALUES
	('classic suit', 'jacket+pants+shoes with camera and integrated
telephone device', 1),
	('civilian suit', 'jacket+jeans+trainers with camera and integrated
telephone device', 2),
	('Lady pistol', '2 shots, 45 cal., silenced', 1),
	('M21', 'Sniper rifle, silenced', 2),
	('Train ticket', 'from Stambul to Bagdat on 2020-07-08', 2);

INSERT INTO public.agent (name, rank, pack_id, unit_profile_id, available)
	VALUES
	('Blackbeard', 3, 1, 4, true),
	('Tiger', 3, 2, 3, false);
INSERT INTO public.agent_mission (agent_id, mission_id, info, date_from,
date_to)
	VALUES
	(1, 2 , 'excellent chances, agent prepared and dislocated', '2020-
07-08', '2020-08-15'),
	(2, 1 , 'high risk, dangerous mission', '2020-07-15', '2020-03-20');

INSERT INTO public.mission_result (mission_id, info, time)
```

```
        VALUES
        (1, 'Blackbeard aka Dmitry Tachankin [Blackbeard] has done his
mission with success. General eliminated', DEFAULT),
        (2, 'Diplomat now being protected by Wei Chen [Tiger]', DEFAULT),
        (3, 'Agent Colossus [Dober Lodvinof] revealed by mafia and killed',
DEFAULT),
        (3, 'Body of Agent Colossus [Dober Lodvinof] delivered to morgue',
DEFAULT);

INSERT INTO public.loot (mission_result_id, rank, info, name)
        VALUES
        (1, 3, 'photos of mission process and result', 'evidences'),
        (2, 2, 'We confirmed the message from Tiger: diplomat being
secured', 'message'),
                                        (4, 4, 'Corpse of the agent Colossus. Important
information in usb-storage found inside his belly.', 'information');
```

Индивидуальное задание:

1. Во все таблицы, где есть поля, которые могут быть обновлены у записей (например, статус), добавить поля `creating_date` и `updating_date`, то есть дату создания записи и дату последнего обновления. Рекомендуемый тип для этих полей: `datetime` или `timestamp`. Для всех текущих записей проставить их как текущее время.
2. Ссылку на `pack` переместить из таблицы `agent` в таблицу `agent_mission`. Тогда каждая запись в таблице `agent_mission` будет нам говорить о том, какой агент был назначен на какую миссию и какой набор вещей ему был выдан именно на эту миссию. Для текущих данных перенести значения из одной таблицы в другую.

Скрипт изменения БД в связи с индивидуальным заданием:

```
ALTER TABLE mission
ADD creating_date TIMESTAMP NULL,
ADD updating_date TIMESTAMP NULL;

UPDATE mission
SET creating_date = now()::timestamp,
updating_date = now()::timestamp;


-- присоединение pack к agent
ALTER TABLE agent
ADD pack_id INT NULL;

ALTER TABLE agent
ADD FOREIGN KEY(pack_id) REFERENCES pack(pack_id);

INSERT INTO agent(pack_id)
SELECT pack_id FROM agent_mission;

ALTER TABLE agent_mission
```

```
DROP COLUMN pack_id;

-- присоединение pack к agent_mission
ALTER TABLE agent_mission
ADD pack_id INT NULL;

ALTER TABLE agent_mission
ADD FOREIGN KEY(pack_id) REFERENCES pack(pack_id);

INSERT INTO agent_mission(pack_id)
SELECT pack_id FROM agent;

ALTER TABLE agent
DROP COLUMN pack_id;
```

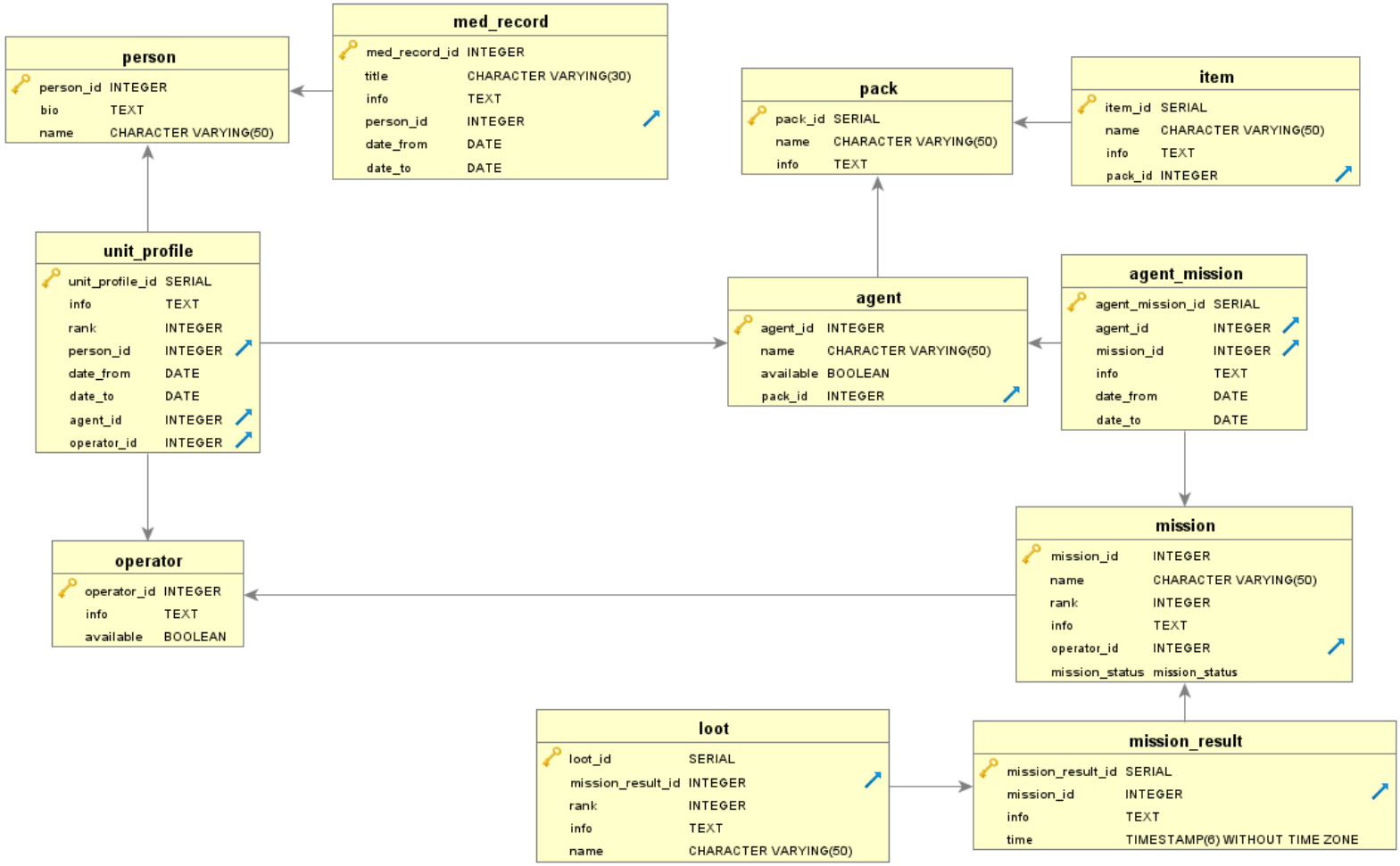


Рис 1. Итоговая схема БД.

https://gitlab.icc.spbstu.ru/mermaider/db-spring-2020-spybase

2. Лабораторная работа №2. Генерация тестовых данных

Цели работы

Сформировать набор данных, позволяющий производить операции на реальных объемах данных.

Программа работы

1. Реализация в виде программы параметризируемого генератора, который позволит сформировать набор связанных данных в каждой таблице.
2. Частные требования к генератору, набору данных и результирующему набору данных:
    - количество записей в справочных таблицах должно соответствовать ограничениям предметной области
    - количество записей в таблицах, хранящих информацию об объектах или субъектах должно быть параметром генерации
    - значения для внешних ключей необходимо брать из связанных таблиц
    - сохранение уже имеющихся данных в базе данных

Выполнение работы

Был выбран язык python и драйвер psycopg2, тип генерации – полностью случайный. Было написано консольное приложение, где в качестве аргументов можно задать количество записей, вставляемых в какую-либо таблицу. Значения по умолчанию – 0. Приложение может работать как с пустой базой данных, так и с уже заполненной, используя имеющиеся данные. Заполнять можно как по одной таблице, так и несколько или все сразу.

Кроме того, существует отдельный параметр ***--truncate*** .

Когда он равен 1, ко всем таблицам в базе данных применяется truncate table и restart identity перед заполнением.

Все ограничения базы данных соблюдены при генерации. Полный код приложения расположен по ссылке https://gitlab.icc.spbstu.ru/mermaider/db-spring-2020-spybase/-/blob/master/2/main.py

Для чтения данных о соединении используется файл config.ini.

3. Лабораторная работа №3. Язык SQL DML

Цели работы

Познакомиться с языком создания запросов управления данными SQL-DML.

Программа работы

- Изучение SQL-DML.
- Выполнение всех запросов из списка стандартных запросов. Демонстрация результатов преподавателю.
- Получение у преподавателя и реализация SQL-запросов в соответствии с индивидуальным заданием. Демонстрация результатов преподавателю.
- Сохранение в БД выполненных запросов SELECT в виде представлений, запросов INSERT, UPDATE или DELETE -- в виде ХП. Выкладывание скрипта в GitLab.

Выполнение работы

Стандартные запросы

Сделать выборку всех данных из каждой таблицы.

Запросы:

```
SELECT * FROM person;
SELECT * FROM med_record;
SELECT * FROM unit_profile;
SELECT * FROM operator;
SELECT * FROM agent;
SELECT * FROM pack;
SELECT * FROM item;
SELECT * FROM agent_mission;
SELECT * FROM mission;
SELECT * FROM mission_result;
SELECT * FROM loot;
```

Сделать выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, LIKE, BETWEEN, IN (не менее 3-х разных примеров).

Запросы:

```
-- Вывод личных дел работников бюро, ранг доступа которых в диапазоне от 2 до 3

SELECT * FROM unit_profile WHERE rank BETWEEN 2 AND 3;
```

Результат:

	unit_profile_i integer	info text	rank integer	person_id integer	date_from date	date_to date	agent_id integer	operator_id integer
1	1	эюцвзхезлъяо...	3	1739	1960-08-06	1943-09-...	[null]	110
2	2	къечщьусацлэ...	3	522	1962-08-23	1995-09-...	98	135
3	4	гчйшхцюувмф...	2	1521	1999-03-25	1943-05-...	63	105
4	5	жэйпчъгмячю...	3	4	1937-12-06	1979-10-...	139	[null]
5	9	эфрфвнсрщмсх	2	1182	1950-08-04	2005-06-...	154	[null]
6	11	ъссбвпкллрю...	2	1546	1973-11-16	1923-03-...	[null]	23
7	13	кйрбтшмоыььт...	3	1551	1955-04-17	1977-03-...	230	[null]
8	18	п	2	1050	1969-12-11	2000-04-...	[null]	41
9	23	ауссщтафдвою...	3	1913	1984-12-25	1937-09-...	[null]	192
10	24	тбюыёьзсткрн...	2	5	1946-02-27	2006-04-...	43	95

Рис 2. Результат запроса

```
--Вывод агентов, в имени которых есть часть “уаз”

SELECT * FROM agent WHERE agent.name LIKE '%уаз%';
```

Результат:

	agent_id integer	name character varying (50)	available boolean	pack_id integer
1	177	уазцзёяжёезпзммвёоъечлшлияёиъф...	true	2458

Рис 3. Результат запроса

```
--Вывод медицинских карт, даты date_from которых '1984-12-22' и '1952-07-09'.

SELECT * FROM med_record WHERE date_from IN ('1984-12-22', '1952-07-09');
```

Результат:

	med_record_id integer	title character varying (30)	info text	person_id integer	date_from date	date_to date
1	4	чууёлеыкчб	ейь...	4	1984-12-22	1967-05-...
2	8	пецйжнайцщьъётшпёзхьн	жбо...	8	1952-07-09	1980-03-...

Рис 4. Результат запроса

Создать в запросе вычисляемое поле.

```
--Столбец days – это количество суток, прошедших между date_to и date_from в личных делах (unit_profile) сотрудников

 SELECT date_from, date_to, unit_profile_id, abs(extract(epoch FROM date_to::timestamp - date_from::timestamp)/3600/24) AS days FROM unit_profile;
```

Результат:

	date_from date	date_to date	unit_profile_id integer	days double precision
1	1960-08-06	1943-09-...	1	6161
2	1962-08-23	1995-09-...	2	12085
3	1953-08-14	1947-05-...	3	2274
4	1999-03-25	1943-05-...	4	20401
5	1937-12-06	1979-10-...	5	15297
6	1988-04-03	1994-07-...	6	2284
7	1926-06-12	1933-10-...	7	2690
8	1985-11-11	1965-08-...	8	7396
9	1950-08-04	2005-06-...	9	20029
10	1970-09-25	1950-07-...	10	7363
11	1973-11-16	1923-03-...	11	18514

Рис 5. Результат запроса

Сделать выборку всех данных с сортировкой по нескольким полям.

```
--Сортировка данных в таблице mission по рекомендуемому рангу выполнения и по текущему статусу миссии:

 SELECT * FROM mission ORDER BY rank, mission_status;
```

Результат:

	mission_id integer	name character varying (50)	rank integer	info text	operator_id integer	mission_status mission_status
234	2613	юэлсишкаёырбзхвгивяг...	1	нпщ...	22	PLANNING
235	5504	ыкыёгзараублциноайан...	1	пгы...	84	PLANNING
236	2617	жзлрапаулумъгдынмбёъ...	1	мкз...	126	PLANNING
237	2632	ёрзшсллрилчмпщбышпз...	1	еенз...	71	PLANNING
238	2703	ёюцлёюьйюмбавцжрэч...	1	итлв...	195	PLANNING
239	651	ртжгюьдфучшывмлувэс...	1	кебс...	71	PLANNING
240	4486	рмцыхяолйкюйвыфхдф...	1	еыа...	116	PLANNING
241	4485	ыьщдьфуифйфзрэсчрж...	1	гфю...	12	PLANNING
242	4481	йссъхунждъэрюхдаэшч...	1	щвш...	161	PLANNING
243	4469	ысцыхлщвюнжьтъёыхе...	1	бюс...	131	PLANNING
244	986	ечмсжьёефлюпъглмыйт...	1	озво...	156	PLANNING
245	2733	зюмсьвижпхзрйщзьютк...	1	ртэа...	90	PLANNING
246	461	ьоуттшааюнзэмтщыыхи...	1	ёизх...	136	PLANNING
247	2215	нбьсцифнтпэягеуозвсэш...	1	ърк...	198	STARTING
248	732	июзтвчгёщыуцуопдаъьн...	1	хэчё...	144	STARTING
249	907	амосьфеыаауйщещиён...	1	ггтч...	102	STARTING
250	885	жсъфеабмбгигаеуюувах...	1	ътш...	109	STARTING
251	862	чбгуктфцнеаъфеттхязщя...	1	уеут...	132	STARTING
252	861	схлыркожёфыжркддаег...	1	озм...	6	STARTING
253	752	шрзгяйлзаивтафмфячъа...	1	ъжт...	172	STARTING
254	755	ссейъахлюыйиюшозъф...	1	ьме...	137	STARTING

Рис 6. Результат запроса

Создать запрос, вычисляющий несколько совокупных характеристик таблиц

```
--Вывод среднего значения ранга в делах сотрудников и самой поздней даты начала действия личного дела.

 SELECT AVG(up.rank), MAX(up.date_from) FROM unit_profile up;
```

Результат:

	avg numeric	max date
1	3.0070000000000000	2007-05-12

Рис 7. Результат запроса

Сделать выборку данных из связанных таблиц

```
--Вывод агентов и названия миссий, в которых они участвовали

SELECT a.agent_id, a.name, m.mission_id, m.name

FROM agent a

JOIN agent_mission am ON a.agent_id = am.agent_id

JOIN mission m ON am.mission_id = m.mission_id

ORDER BY agent_id

LIMIT 100;
```

Результат:

	agent_id integer	name character varying (50)	mission_id integer	name character varying (50)
6	1	эёжжяэфхсалчмянефрч...	783	кужкгнйфшъэвммрдежу...
7	1	эёжжяэфхсалчмянефрч...	1477	плфёаэцъвкеншнемцзц...
8	1	эёжжяэфхсалчмянефрч...	2120	фьэаетлешяяхщжчбщян...
9	1	эёжжяэфхсалчмянефрч...	3620	ёйпюлгкиьйюбхёщфокэ...
10	1	эёжжяэфхсалчмянефрч...	4789	цфпкибвнжбчюрсюафх...
11	1	эёжжяэфхсалчмянефрч...	3009	юемдэцучрздытэйаттвт...
12	1	эёжжяэфхсалчмянефрч...	2952	чщцоеоцюгблщрвмжшр...
13	1	эёжжяэфхсалчмянефрч...	1937	еукщиаюпирцэфылйшъг...
14	1	эёжжяэфхсалчмянефрч...	383	вёёмэйьюцефыихздфыс...
15	1	эёжжяэфхсалчмянефрч...	795	цчлущьхжюжиюцмбфзо...
16	1	эёжжяэфхсалчмянефрч...	3052	фчфнмзбйълцлитыгнща...
17	1	эёжжяэфхсалчмянефрч...	850	вчркгчтёринцныъопжац...
18	1	эёжжяэфхсалчмянефрч...	2527	фабйщьсмимяъвжйсбм...
19	2	фбдемттюмбтрсздслтоп...	901	хмэзиыаёытаёымдшёлэ...
20	2	фбдемттюмбтрсздслтоп...	1558	фсфуыоиыеёисдбярщщ...
21	2	фбдемттюмбтрсздслтоп...	340	срскляаёэкеккуфлщнкц...
22	2	фбдемттюмбтрсздслтоп...	950	бафпхърдвлчпярнпчрют...
23	2	фбдемттюмбтрсздслтоп...	1640	ихмбфявслеммюцпкпть...
24	2	фбдемттюмбтрсздслтоп...	4229	ъыркыиюаэцекяхёуъзхл...
25	2	фбдемттюмбтрсздслтоп...	4115	хсбшууезяэзёгдргюуечл...
26	2	фбдемттюмбтрсздслтоп...	490	ншгьхбаощйзгэюьъщцш...
27	2	фбдемттюмбтрсздслтоп...	746	якёмгауябыъэёриэйбну...

Рис 8. Результат запроса

```
--Вывод всех миссий FINISHED с агентами

     SELECT agent.agent_id AS aid, agent.name AS aname,
mission.mission_id AS mid, mission.name AS mname,
     mission.mission_status AS mstatus
     FROM agent
     JOIN agent_mission ON agent.agent_id =
agent_mission.agent_id
     JOIN mission ON agent_mission.mission_id =
mission.mission_id
     WHERE mission.mission_status = 'FINISHED'
```

Результат:

	aid integer	aname character varying (50)	mid integer	mname character varying (50)	mstatus mission_status
1	51	Искариот	1758	лээрщнооояуэгтйднкюгк...	FINISHED
2	51	Искариот	1728	оахмешцпозобщжбэчун...	FINISHED
3	56	ЖезинеРадастый	1033	учашщрэёшадкхпбёужи...	FINISHED
4	57	Вуфей	1410	итгёшлышысццздузужкк...	FINISHED
5	69	Виталь	1428	дьпюшгйлценифаууцрэх...	FINISHED
6	91	ЛопайПопой	1948	сеоёаёофуиаэвймшсёяё...	FINISHED
7	100	КрутойПАЦАН	907	щекгфщмшбуеьлфцббтж...	FINISHED
8	159	ЛопайПопой	1977	мгъэиилчёбрлфмцйешэ...	FINISHED
9	166	Кинглиаолианг	1200	тщнлъдущнрэипгктюкм...	FINISHED
10	184	ШалДре	401	фяъмалшщчшйцсёъашл...	FINISHED
11	210	палантир	341	холцчпмфннйссукгюмр...	FINISHED

Рис 9. Результат запроса

Создать запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки

Группировка миссий по их статусу, расчёт количества миссий с определённым статусом и вывод таких статусов, к которым относится более 400 миссий.

```
SELECT COUNT(mis.mission_id), mission_status FROM mission mis
GROUP BY mission_status HAVING COUNT(mis.mission_id) > 400;
```

Результат:

	count bigint	mission_status mission_status
1	410	PERFORMING
2	402	FINISHED
3	402	STARTING

Рис 10. Результат запроса

Придумать и реализовать пример использования вложенного запроса

```
--Вложенный запрос, который выводит строку из таблицы медкарточек, где есть ID персоны, у которой в информации к болезни есть часть “ушту” и при этом дата начала этой болезни лежит в пределах 20-ых и 80-ых годов

 SELECT * FROM person per WHERE per.person_id = (SELECT mr.person_id FROM med_record mr WHERE mr.info LIKE '%ушту%' AND (mr.date_from BETWEEN '1920.01.01' AND '1980.01.01' ));
```

Результат:

	person_id integer	bio text	name character varying (50)
1	1	нъътзгербжыеюойёйатчтыиепгднщ...	Карим

Рис 11. Результат запроса

С помощью команды INSERT добавить в каждую таблицу по одной записи.

Запросы:

```
INSERT INTO public.person (bio, name)
	VALUES
	('1985, Portsmooth, England, fem.', 'Lucy Landa'),
	('1987, Kong Hong, China, mal.', 'Wei Shen'),
	('1983, Ryazan, USSR', 'Dmitry Tachankin'),
	('1990, Poland', 'Dober Lodvinof');

INSERT INTO public.unit_profile (info, person_id, date_from, date_to)
	VALUES
	('Good coordinator, knows 7 languages:..., the most young in agency', 1, '2006-05-15','2008-05-15'),
	('Excellent coordinator, knows 9 languages:...', 1, '2008-05-15', null),
	('Excellent integorator, knows 3 languages:...', 2, '2009-05-15', null),
	('Heavy weapons and explosives expert, 2 languages:...', 3, '2010-05-15', null),
	('Stealth expert, agent Colossus eliminated during mission "Sunset 2020-07-23"', 4, '2009-05-15', '2020-07-23');

INSERT INTO public.med_record (title, info, person_id, date_from, date_to)
	VALUES
	('Pneumonia', 'hard case', 1, '2010-05-15','2010-06-25'),
	('KIA', 'Bullet shot, blood loss', 4,'2020-03-20','2020-03-20');

INSERT INTO public.operator (info, unit_profile_id, available)
	VALUES
```

```
        ('operations manager', 1, false);

INSERT INTO public.mission (name, rank, info, operator_id, mission_status)
        VALUES
        ('Electric Vision', 3, 'Eliminate the general', 1, 'FINISHED'),
        ('Rising Storm', 3, 'Rescue the diplomat', 1, 'PERFORMING'),
        ('Sunset', 3, 'Lenetti mafia undercover', 1, 'FINISHED');
INSERT INTO public.pack (name, info)
        VALUES
        ('Civilian disguise', 'civil concealment + hidden pistol'),
        ('Killer loadout', 'handheld rifle + high tech jacket');

INSERT INTO public.item (name, info, pack_id)
        VALUES
        ('classic suit', 'jacket+pants+shoes with camera and integrated telephone device', 1),
        ('civilian suit', 'jacket+jeans+trainers with camera and integrated telephone device', 2),
        ('Lady pistol', '2 shots, 45 cal., silenced', 1),
        ('M21', 'Sniper rifle, silenced', 2),
        ('Train ticket', 'from Stambul to Bagdat on 2020-07-08', 2);

INSERT INTO public.agent (name, rank, pack_id, unit_profile_id, available)
        VALUES
        ('Blackbeard', 3, 1, 4, true),
        ('Tiger', 3, 2, 3, false);
INSERT INTO public.agent_mission (agent_id, mission_id, info, date_from, date_to)
        VALUES
        (1, 2 , 'excellent chances, agent prepared and dislocated', '2020-07-08', '2020-08-15'),
        (2, 1 , 'high risk, dangerous mission', '2020-07-15', '2020-03-20');

INSERT INTO public.mission_result (mission_id, info, time)
        VALUES
        (1, 'Blackbeard aka Dmitry Tachankin [Blackbeard] has done his mission with success.
General eliminated', DEFAULT),
        (2, 'Diplomat now being protected by Wei Chen [Tiger]', DEFAULT),
        (3, 'Agent Colossus [Dober Lodvinof] revealed by mafia and killed', DEFAULT),
        (3, 'Body of Agent Colossus [Dober Lodvinof] delivered to morgue', DEFAULT);

INSERT INTO public.loot (mission_result_id, rank, info, name)
        VALUES
        (1, 3, 'photos of mission process and result', 'evidences'),
        (2, 2, 'We confirmed the message from Tiger: diplomat being secured', 'message'),
        (4, 4, 'Corpse of the agent Colossus. Important information in usb-storage found inside
his belly.', 'information');
```

С помощью оператора UPDATE измените значения полей у всех записей, отвечающих заданному условию

Установить статус доступности available у агентов в состояние FALSE, если они используют pack_id от 1 до 4

```
UPDATE agent SET available = FALSE WHERE pack_id BETWEEN 1 AND 4;
```

До выполнения запроса:

	agent_id integer	name character varying (50)	available boolean	pack_id integer
1	605	Тениарфас	false	3
2	1767	Птичка	true	3
3	1828	Птичка	true	4
4	2306	Залдарад	true	1
5	2738	МедведЗаснул	false	1
6	3259	Компик	true	4
7	3372	Вий	true	2

Рис 12. Результат запроса

После выполнения запроса:

	agent_id integer	name character varying (50)	available boolean	pack_id integer
1	605	Тениарфас	false	3
2	1767	Птичка	false	3
3	1828	Птичка	false	4
4	2306	Залдарад	false	1
5	2738	МедведЗаснул	false	1
6	3259	Компик	false	4
7	3372	Вий	false	2

Рис 13. Результат запроса

С помощью оператора DELETE удалить запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики

```
--Изначально добавляем назначение на миссию (agent_mission) с очень поздней датой date_to, а затем удаляем её

 INSERT INTO agent_mission (agent_id, mission_id, info, date_from, date_to)
VALUES (3622, 204, 'удали меня', '2008-05-15', '2021-05-20');

DELETE FROM agent_mission WHERE date_to = (select max(date_to) from agent_mission);
```

До выполнения запроса:

	agent_mission_id integer	info text	date_to date
1	1487	удали меня	2021-05-20
2	1485	чзсёргъливлртбниьоапсашрцтбтвып...	1937-06-04
3	1484	асшлщмршъозщдзьфдвзтфбжбнбучк...	1999-05-05
4	1483	йфупылсяднказбиаыивчбуиабхойвр...	1956-11-07
5	1482	вънычиоёшншёфхщгртзрннхоылфък...	1950-07-10
6	1481	цйхъцхвапумощкнлквнпёффшгбтзён...	1929-03-27
7	1480	кфржсйзбэчыфнхаыэньммбььшдюё...	1954-12-18
8	1479	чгцагензиющнлыобщднемулнншопд...	1965-12-17

Рис 14. Результат запроса

После выполнения запроса:

	agent_mission_id integer	info text	date_to date
1	1485	чзсёргъливлртбниьоапсашрцтбтвып...	1937-06-04
2	1484	асшлщмршъозщдзьфдвзтфбжбнбучк...	1999-05-05
3	1483	йфупылсяднказбиаыивчбуиабхойвр...	1956-11-07
4	1482	вънычиоёшншёфхщгртзрннхоылфък...	1950-07-10
5	1481	цйхъцхвапумощкнлквнпёффшгбтзён...	1929-03-27

Рис 15. Результат запроса

С помощью оператора DELETE удалить записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос)

```
--Удаляет строку agent которая нигде не используется в таблице unit_profile.

 INSERT INTO agent VALUES (100000,'удали меня', TRUE, NULL);

DELETE agent_id FROM agent WHERE agent_id NOT IN (SELECT up.agent_id FROM
unit_profile up);
```

До выполнения запроса:

	agent_id integer	name character varying (50)
1	100000	удали меня
2	4000	Ыхтик
3	3999	Ферлис
4	3998	Тромминор
5	3997	ОрЗу
6	3996	ВИнТ
7	3995	Волшебник

Рис 16. Результат запроса

После выполнения запроса:

	agent_id integer	name character varying (50)
1	4000	Ыхтик
2	3999	Ферлис
3	3998	Тромминор
4	3997	ОрЗу
5	3996	ВИнТ
6	3995	Волшебник

Рис 17. Результат запроса

Создание запросов по индивидуальному заданию:

1) Вывести все миссии агентов, у которых средний результат выполнения миссий не хуже заданного. Предварительно добавить в результат миссии типизированное значение.

```
-- Вывести все миссии агентов, у которых средний результат выполнения миссий
не хуже заданного.
-- Все миссии FINISHED с агентами
WITH finished(aid, aname, mid, mstatus) AS (
        SELECT agent.agent_id AS aid, agent.name AS aname,
mission.mission_id AS mid, mission.name AS mname,
        mission.mission_status AS mstatus
        FROM agent
        JOIN agent_mission ON agent.agent_id = agent_mission.agent_id
        JOIN mission ON agent_mission.mission_id = mission.mission_id
        WHERE mission.mission_status = 'FINISHED'
        GROUP BY aid, mid
),


-- количество финишд миссий по каждому агенту
finishedct(aid, fins) AS (
        SELECT finished.aid AS aid, COUNT(mid) AS fins FROM finished
        GROUP BY finished.aid
),

-- Миссии каждого агента
allmissions(aid, aname, mid, mmstatus) AS (
        SELECT agent.agent_id AS aid, agent.name AS aname,
mission.mission_id AS mid, mission.name AS mname,
        mission.mission_status AS mmstatus
        FROM agent
        JOIN agent_mission ON agent.agent_id = agent_mission.agent_id
        JOIN mission ON agent_mission.mission_id = mission.mission_id
        GROUP BY aid, mid, mmstatus
),

-- count of all missions 4each agent
mct(aid, acnt) AS (
        SELECT am.aid as aid, COUNT(am.mid) AS acnt
        FROM allmissions am
        GROUP BY am.aid
)


SELECT am.aid, am.aname, am.mid, ms.mission_status, fc.fins AS fins,
mct.acnt AS cnt, (fins/(mct.acnt+0.0)) as coef
FROM allmissions am
JOIN finishedct fc ON fc.aid = am.aid
JOIN mct ON mct.aid = am.aid
JOIN mission ms ON ms.mission_id = am.mid

WHERE am.aid IN (
                        SELECT aid from finished
)
GROUP BY am.aid, am.aname, am.mid, ms.mission_status, fins, mct.acnt
        HAVING (fins/(mct.acnt+0.0)) > 0.35
ORDER BY aid, fins DESC
```

2) Удалить из миссий агентов, которые являются операторами в этих миссиях.

```
-- Удалить из миссий агентов, которые являются операторами в этих миссиях.
-- select all unit_profiles where agent and operator is not null
WITH up(aid, oid) AS (
        SELECT agent_id AS aid, operator_id AS oid
        FROM unit_profile
        WHERE agent_id IS NOT NULL AND operator_id IS NOT NULL
        GROUP BY agent_id, operator_id
),


-- select all missions AND their agents,
-- where operator is an agent + agent_id as op_agent for those operators

allmissions(mid, oid, op_agent, aid) AS (
        SELECT mission.mission_id AS mid, mission.operator_id AS oid,
        up.aid as op_agent, am.agent_id AS aid
        FROM mission
        JOIN up ON up.oid = mission.operator_id
        JOIN agent_mission am ON am.mission_id = mission.mission_id
        WHERE mission.operator_id IN (
                SELECT oid FROM up
        )

        GROUP BY mission.mission_id, mission.operator_id,  am.agent_id,
up.aid
),

--select missions where operators are in those missions agent lists
deletefun(mid, oid, opaid, aid) AS (
        SELECT allmissions.mid AS mid, allmissions.oid AS oid,
allmissions.op_agent AS opaid, allmissions.aid AS aid
        FROM allmissions
        WHERE allmissions.op_agent = allmissions.aid
        GROUP BY allmissions.mid, allmissions.oid, allmissions.op_agent,
allmissions.aid
        ORDER BY allmissions.mid
)

DELETE FROM agent_mission
WHERE agent_mission.agent_id
IN (
        SELECT aid FROM deletefun
)


-- SELECT missions, agents, operators, ops-agents,
-- next -> SELECT ops-agents WHERE ops-agent = agent
```

3) Вывести агентов, которые никогда не участвовали в миссиях, где не были добыты луты.

```
--Задание: Вывести агентов, которые никогда не участвовали в миссиях, где не
были добыты луты.

--все миссии с mission_result, лутами, агентами
WITH mss(mid, mrid, lid, aid) AS (
```

```
    SELECT mission.mission_id, mr.mission_result_id, loot.loot_id,
agent.agent_id
    FROM mission
    JOIN mission_result mr ON mr.mission_id = mission.mission_id
    JOIN loot on loot.mission_result_id = mr.mission_result_id
    JOIN agent_mission am ON am.mission_id = mission.mission_id
    JOIN agent ON agent.agent_id = am.agent_id
    GROUP BY mission.mission_id, mr.mission_result_id, loot.loot_id,
agent.agent_id
),

--агенты и миссии, где хотя бы однажды лут не был получен
nolootags(aid, mid) AS (
    SELECT agent.agent_id, mission.mission_id
    FROM mission
    JOIN agent_mission ON agent_mission.mission_id = mission.mission_id
    JOIN agent ON agent.agent_id = agent_mission.agent_id
    WHERE mission.mission_id NOT IN
    (SELECT mid FROM mss)
    GROUP BY  mission.mission_id, agent.agent_id
)

SELECT agent.agent_id, agent.name, mission.mission_id
FROM agent
JOIN agent_mission ON agent_mission.agent_id = agent.agent_id
JOIN mission on mission.mission_id = agent_mission.mission_id
WHERE agent.agent_id NOT IN (SELECT aid FROM nolootags)
ORDER BY agent.agent_id
```

Лабораторная работа №4. Нагрузка базы данных и оптимизация запросов

Цели работы

Знакомство с проблемами, возникающими при высокой нагрузке на базу данных, и методами их решения, путем оптимизации запросов.

Программа работы

1. *Написание параметризированных типовых запросов пользователей.* Необходимо проанализировать БД и сформировать не менее 5 запросов, результаты которых требовались бы потенциальным пользователям данных. Запросы должны содержать соединения таблиц.
2. *Моделирование нагрузки базы данных.* Параметризируемое (количество запросов, количество потоков) приложение, где можно было моделировать нагрузку на БД и получать результаты выполнения в виде графиков и таблиц.
3. *Снятие показателей работы сервиса и построение соответствующих графиков.* Необходимо оценить зависимость времени ответа от

количества потоков и от количества запросов в единицу времени. Также полезно иметь возможность оценить нарастание нагрузки с разной скоростью и зависимость показателей от скорости. При оценке показателей необходимо пытаться добиться ситуации, когда нагрузка "кладет" БД.

4. *Применение возможных оптимизаций запросов и повторное снятие показателей.* Анализ планов запросов, формирование предложений по снижению времени выполнения запросов. Последовательное применение шагов по оптимизации и снятие показателей после каждого шага.
5. *Сравнительный анализ результатов.*
6. *Демонстрация результатов преподавателю.*

Выполнение работы

Для реализации нагрузки базы данных использовалась программа, разработанная в работе No2. С ее помощью было задано относительно большое количество записей в таблицах (около 10 тысяч в каждой).

В качестве типовых были созданы следующие запросы:

```
1: '''SELECT agent.agent_id, agent.name, person.name, pack.name,
unit_profile.rank
      FROM agent
      JOIN pack ON pack.pack_id = agent.pack_id
      JOIN unit_profile ON unit_profile.agent_id = agent.agent_id
      JOIN person ON person.person id = unit profile.person id
      WHERE person.name = (%s);'''
```

```
2: '''SELECT "operator".operator_id, person.name, med_record.title,
med_record.date_from, med_record.date_to
      FROM "operator"
      JOIN unit_profile ON "operator".operator_id = unit_profile.operator_id
      JOIN person ON person.person_id = unit_profile.person_id
      JOIN med_record ON person.person_id = med_record.person_id
      WHERE med_record.date_to < (%s);'''
```

```
3: '''SELECT agent.name, pack.name as pack, item.name as item
      FROM agent
      JOIN pack ON agent.pack_id = pack.pack_id
```

```
      JOIN item ON pack.pack_id = item.pack_id
      WHERE agent.name = (%s);'''
```

```
4: '''SELECT mission.mission_id, mission.name, mission.info, "operator".info
as operator_info, mission_result.info
        as mission_res
      FROM mission
      JOIN "operator" ON mission.operator_id = "operator".operator_id
      JOIN mission_result ON mission.mission_id = mission_result.mission_id
      WHERE mission_result."time" between %(time_from)s and %(time_to)s;'''
```

```
5: '''SELECT agent_mission.agent_mission_id, agent.name, mission.name,
agent_mission.date_from,
      agent_mission.date_to
      FROM agent_mission
      JOIN agent ON agent.agent_id = agent_mission.agent_id
      JOIN mission ON mission.mission_id = agent_mission.mission_id
      JOIN "operator" ON "operator".operator_id = mission.operator_id
      WHERE agent_mission.date_to > (%s);'''
```

Эти запросы являются достаточно сложными для успешной последующей оптимизации, так как содержат условия, группировку, соединения и вложенные запросы. Также в них уже заданы поля для параметров, которые в программе нагрузки будут генерироваться случайным образом.

Полученные запросы будем использовать для нагрузки БД. Они будут задаваться двумя способами: в одном потоке некоторое количество раз подряд и в нескольких потоках одновременно.

Для получения статистики по выполнению запросов и ответу БД на эти действия используем оператор EXPLAIN ANALYZE. Из результатов будет доступно время на планирование и выполнения определенного запроса. Эти данные будут сохраняться, а затем использоваться для построения графиков зависимости усредненного времени выполнения одного запроса от количества заданных запросов и зависимости времени от количества потоков.

Для оптимизации были использован два способа: добавления индексов и сохранение плана запроса.

Создание индексов

Индексы — это традиционное средство увеличения производительности БД. Используя индекс, сервер баз данных может находить и извлекать нужные строки гораздо быстрее, чем без него.

PostgreSQL поддерживает несколько типов индексов: B-дерево, хеш, GiST, SP-GiST, GIN и BRIN. Для разных типов индексов применяются разные алгоритмы, ориентированные на определённые типы запросов. По умолчанию команда CREATE INDEX создаёт индексы типа B-дерево, эффективные в большинстве случаев.

B-деревья могут работать в условиях на равенство и в проверках диапазонов с данными, которые можно отсортировать в некотором порядке. Точнее, планировщик запросов PostgreSQL может задействовать индекс-B-дерево, когда индексируемый столбец участвует в сравнении с одним из следующих операторов:

- <
- <=
- =
- >=
- >

Добавление индексов:

```
def optimize_create_indices():
    connect = psycopg2.connect(**params)
    cursor = connect.cursor()
    cursor.execute("CREATE INDEX IF NOT EXISTS i1 ON person(name);")
    cursor.execute("CREATE INDEX IF NOT EXISTS i2 ON med_record(date_to);")
    cursor.execute("CREATE INDEX IF NOT EXISTS i3 ON agent(name);")
    cursor.execute("CREATE INDEX IF NOT EXISTS i4 ON mission_result(time);")
    cursor.execute("CREATE INDEX IF NOT EXISTS i5 ON
agent_mission(date_to);")
    # cursor.execute("CREATE INDEX IF NOT EXISTS i6 ON
agent_mission(date_from);")
    # cursor.execute("CREATE INDEX IF NOT EXISTS i7 ON
agent_mission(date_to);")
    connect.commit()
    connect.close()
```

Индексы были добавлены для тех атрибутов, которые встречаются при операторах WHERE, JOIN ON, GROUP BY.

Подготовленный оператор представляет собой объект на стороне сервера, позволяющий оптимизировать производительность приложений. Когда выполняется PREPARE, указанный оператор разбирается, анализируется и переписывается. При последующем выполнении команды EXECUTE подготовленный оператор планируется и исполняется. Такое разделение труда исключает повторный разбор запроса, при этом позволяет выбрать наилучший план выполнения в зависимости от определённых значений параметров. Подготовленные операторы существуют только в рамках текущего сеанса работы с БД.

Подготовленные операторы дают наибольший выигрыш в производительности, когда в одном сеансе выполняется большое число однотипных операторов. Отличие в производительности особенно значительно, если операторы достаточно сложны для планирования или перезаписи, например, когда в запросе объединяется множество таблиц или необходимо применить несколько правил.

До использования индексов:

План запроса 1:

```
Nested Loop  (cost=9.93..721.44 rows=4 width=138) (actual time=2.014..3.390 rows=2 loops=1)
  ->  Nested Loop  (cost=9.79..720.79 rows=4 width=41) (actual time=2.002..3.376 rows=2 loops=1)
        ->  Hash Join  (cost=9.50..705.08 rows=4 width=20) (actual time=1.296..3.347 rows=6 loops=1)
              Hash Cond: (person.person_id = unit_profile.person_id)
              ->  Seq Scan on person  (cost=0.00..694.00 rows=412 width=16) (actual time=0.029..3.127 rows=412 loops=1)
                    Filter: ((name)::text = 'Павел'::text)
                    Rows Removed by Filter: 19588
              ->  Hash  (cost=7.00..7.00 rows=200 width=12) (actual time=0.167..0.167 rows=200 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 17kB
                    ->  Seq Scan on unit_profile  (cost=0.00..7.00 rows=200 width=12) (actual time=0.076..0.121 rows=200 loops=1)
        ->  Index Scan using agend_bio_pk on agent  (cost=0.29..3.92 rows=1 width=25) (actual time=0.004..0.004 rows=0 loops=6)
              Index Cond: (agent_id = unit_profile.agent_id)
  ->  Index Scan using pack_pk on pack  (cost=0.14..0.16 rows=1 width=105) (actual time=0.006..0.006 rows=1 loops=2)
        Index Cond: (pack_id = agent.pack_id)
Planning Time: 0.523 ms
Execution Time: 3.440 ms
```

Рис 18. План запроса

План запроса 2:

```
Nested Loop  (cost=10.06..784.50 rows=6 width=55) (actual time=38.310..363.259 rows=4 loops=1)
  Join Filter: (unit_profile.person_id = person.person_id)
  ->  Nested Loop  (cost=9.77..766.34 rows=6 width=51) (actual time=38.251..331.857 rows=4 loops=1)
        ->  Hash Join  (cost=9.50..762.92 rows=6 width=51) (actual time=32.407..303.403 rows=5 loops=1)
              Hash Cond: (med_record.person_id = unit_profile.person_id)
              ->  Seq Scan on med_record  (cost=0.00..751.00 rows=629 width=43) (actual time=10.534..303.048 rows=643 loops=1)
                    Filter: (date_to < '1925-01-01'::date)
                    Rows Removed by Filter: 19357
              ->  Hash  (cost=7.00..7.00 rows=200 width=8) (actual time=0.097..0.097 rows=200 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 16kB
                    ->  Seq Scan on unit_profile  (cost=0.00..7.00 rows=200 width=8) (actual time=0.016..0.058 rows=200 loops=1)
        ->  Index Only Scan using person_pk on operator  (cost=0.27..0.57 rows=1 width=4) (actual time=5.684..5.684 rows=1 loops=5)
              Index Cond: (operator_id = unit_profile.operator_id)
              Heap Fetches: 4
  ->  Index Scan using person_file_pk on person  (cost=0.29..3.01 rows=1 width=16) (actual time=7.844..7.844 rows=1 loops=4)
        Index Cond: (person_id = med_record.person_id)
Planning Time: 157.623 ms
Execution Time: 363.321 ms
```

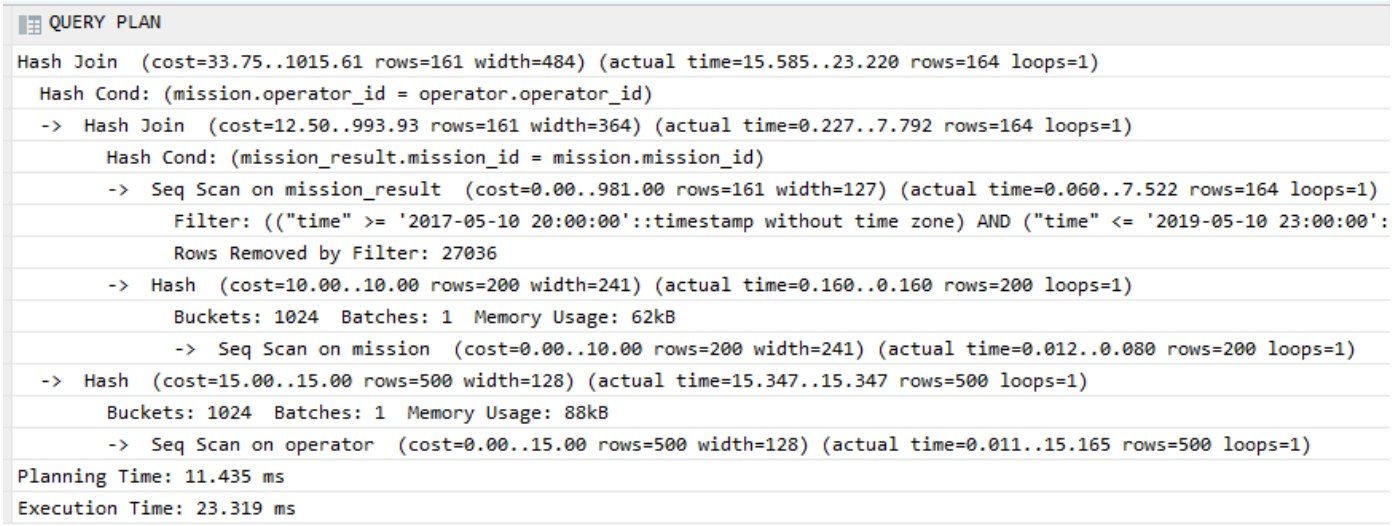
Рис 19. План запроса

План запроса 3:

```
Hash Join  (cost=23.00..439.08 rows=144 width=219) (actual time=138.902..138.903 rows=0 loops=1)
  Hash Cond: (agent.pack_id = pack.pack_id)
  ->  Hash Join  (cost=11.50..427.19 rows=143 width=126) (actual time=119.669..119.669 rows=0 loops=1)
        Hash Cond: (agent.pack_id = item.pack_id)
        ->  Seq Scan on agent  (cost=0.00..413.00 rows=144 width=21) (actual time=119.667..119.667 rows=0 loops=1)
              Filter: ((name)::text = 'Стаф'::text)
              Rows Removed by Filter: 20004
        ->  Hash  (cost=9.00..9.00 rows=200 width=105) (never executed)
              ->  Seq Scan on item  (cost=0.00..9.00 rows=200 width=105) (never executed)
  ->  Hash  (cost=9.00..9.00 rows=200 width=105) (actual time=19.224..19.225 rows=200 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 36kB
        ->  Seq Scan on pack  (cost=0.00..9.00 rows=200 width=105) (actual time=0.023..19.120 rows=200 loops=1)
Planning Time: 79.191 ms
Execution Time: 138.942 ms
```

Рис 20. План запроса

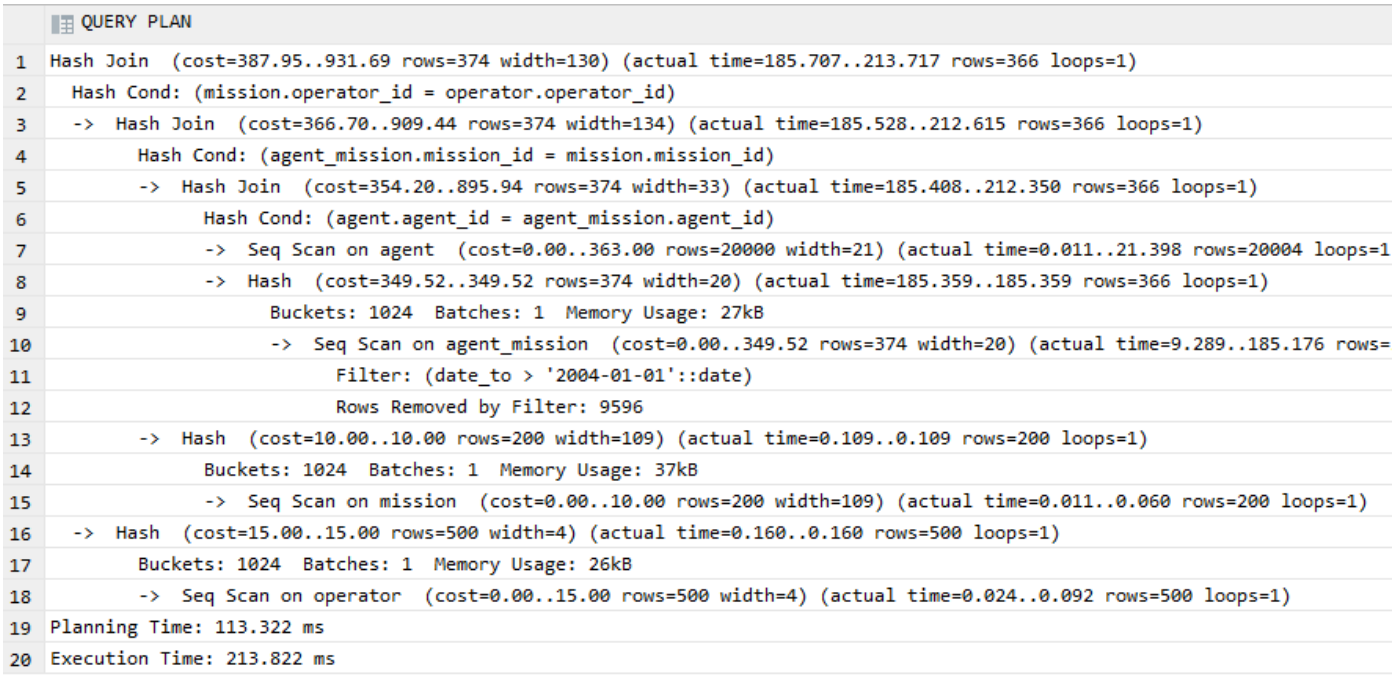
План запроса 4:



```
QUERY PLAN
Hash Join  (cost=33.75..1015.61 rows=161 width=484) (actual time=15.585..23.220 rows=164 loops=1)
  Hash Cond: (mission.operator_id = operator.operator_id)
  ->  Hash Join  (cost=12.50..993.93 rows=161 width=364) (actual time=0.227..7.792 rows=164 loops=1)
        Hash Cond: (mission_result.mission_id = mission.mission_id)
        ->  Seq Scan on mission_result  (cost=0.00..981.00 rows=161 width=127) (actual time=0.060..7.522 rows=164 loops=1)
              Filter: (("time" >= '2017-05-10 20:00:00'::timestamp without time zone) AND ("time" <= '2019-05-10 23:00:00':
              Rows Removed by Filter: 27036
        ->  Hash  (cost=10.00..10.00 rows=200 width=241) (actual time=0.160..0.160 rows=200 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 62kB
              ->  Seq Scan on mission  (cost=0.00..10.00 rows=200 width=241) (actual time=0.012..0.080 rows=200 loops=1)
  ->  Hash  (cost=15.00..15.00 rows=500 width=128) (actual time=15.347..15.347 rows=500 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 88kB
        ->  Seq Scan on operator  (cost=0.00..15.00 rows=500 width=128) (actual time=0.011..15.165 rows=500 loops=1)
Planning Time: 11.435 ms
Execution Time: 23.319 ms
```

Рис 21. План запроса

План запроса 5:



```
QUERY PLAN
Hash Join  (cost=387.95..931.69 rows=374 width=130) (actual time=185.707..213.717 rows=366 loops=1)
  Hash Cond: (mission.operator_id = operator.operator_id)
  ->  Hash Join  (cost=366.70..909.44 rows=374 width=134) (actual time=185.528..212.615 rows=366 loops=1)
        Hash Cond: (agent_mission.mission_id = mission.mission_id)
        ->  Hash Join  (cost=354.20..895.94 rows=374 width=33) (actual time=185.408..212.350 rows=366 loops=1)
              Hash Cond: (agent.agent_id = agent_mission.agent_id)
              ->  Seq Scan on agent  (cost=0.00..363.00 rows=20000 width=21) (actual time=0.011..21.398 rows=20004 loops=1
              ->  Hash  (cost=349.52..349.52 rows=374 width=20) (actual time=185.359..185.359 rows=366 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 27kB
                    ->  Seq Scan on agent_mission  (cost=0.00..349.52 rows=374 width=20) (actual time=9.289..185.176 rows=
                          Filter: (date_to > '2004-01-01'::date)
                          Rows Removed by Filter: 9596
        ->  Hash  (cost=10.00..10.00 rows=200 width=109) (actual time=0.109..0.109 rows=200 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 37kB
              ->  Seq Scan on mission  (cost=0.00..10.00 rows=200 width=109) (actual time=0.011..0.060 rows=200 loops=1)
  ->  Hash  (cost=15.00..15.00 rows=500 width=4) (actual time=0.160..0.160 rows=500 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 26kB
        ->  Seq Scan on operator  (cost=0.00..15.00 rows=500 width=4) (actual time=0.024..0.092 rows=500 loops=1)
Planning Time: 113.322 ms
Execution Time: 213.822 ms
```

Рис 22. План запроса

После использования индексов:

План запроса 1:

	QUERY PLAN
1	Nested Loop (cost=21.41..497.85 rows=4 width=138) (actual time=0.451..0.580 rows=2 loops=1)
2	-> Nested Loop (cost=21.27..497.19 rows=4 width=41) (actual time=0.448..0.574 rows=2 loops=1)
3	-> Hash Join (cost=20.98..481.49 rows=4 width=20) (actual time=0.371..0.556 rows=6 loops=1)
4	Hash Cond: (person.person_id = unit_profile.person_id)
5	-> Bitmap Heap Scan on person (cost=11.48..470.41 rows=412 width=16) (actual time=0.170..0.420 rows=412 loops=1)
6	Recheck Cond: ((name)::text = 'Павел'::text)
7	Heap Blocks: exact=267
8	-> Bitmap Index Scan on i1 (cost=0.00..11.38 rows=412 width=0) (actual time=0.140..0.140 rows=412 loops=1)
9	Index Cond: ((name)::text = 'Павел'::text)
10	-> Hash (cost=7.00..7.00 rows=200 width=12) (actual time=0.095..0.095 rows=200 loops=1)
11	Buckets: 1024 Batches: 1 Memory Usage: 17kB
12	-> Seq Scan on unit_profile (cost=0.00..7.00 rows=200 width=12) (actual time=0.012..0.058 rows=200 loops=1
13	-> Index Scan using agend_bio_pk on agent (cost=0.29..3.92 rows=1 width=25) (actual time=0.002..0.002 rows=0 loops=6)
14	Index Cond: (agent_id = unit_profile.agent_id)
15	-> Index Scan using pack_pk on pack (cost=0.14..0.16 rows=1 width=105) (actual time=0.002..0.002 rows=1 loops=2)
16	Index Cond: (pack_id = agent.pack_id)
17	Planning Time: 1.625 ms
18	Execution Time: 0.644 ms

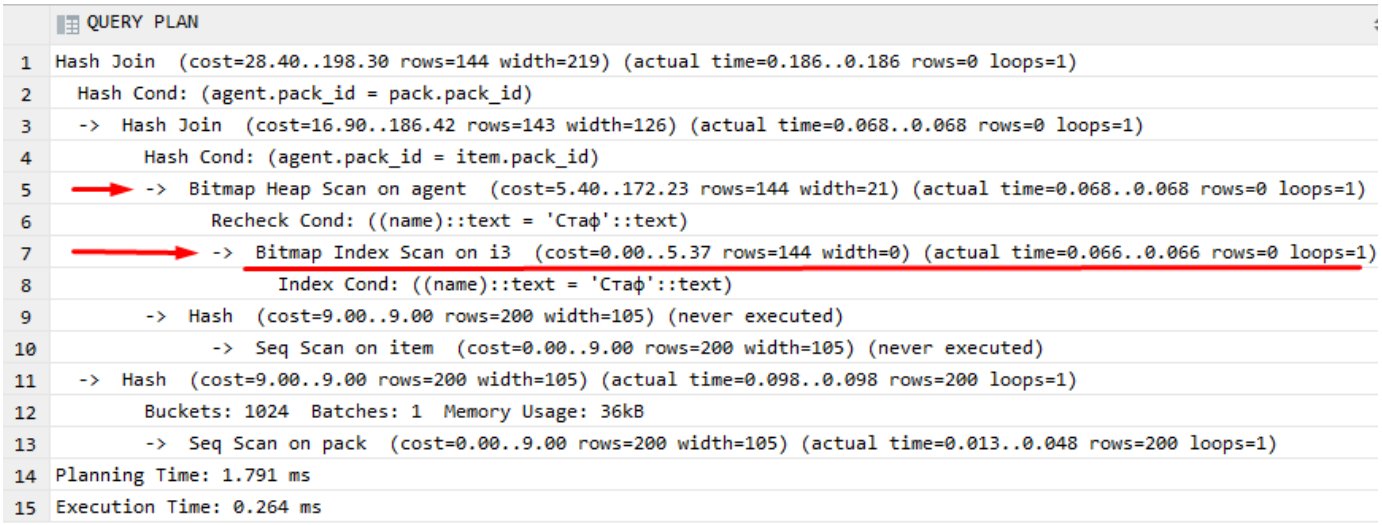
Рис 23. План запроса

План запроса 2:

	QUERY PLAN
1	Nested Loop (cost=23.22..582.13 rows=6 width=55) (actual time=0.294..0.755 rows=4 loops=1)
2	Join Filter: (unit_profile.person_id = person.person_id)
3	-> Nested Loop (cost=22.93..563.97 rows=6 width=51) (actual time=0.288..0.737 rows=4 loops=1)
4	-> Hash Join (cost=22.66..560.55 rows=6 width=51) (actual time=0.272..0.712 rows=5 loops=1)
5	Hash Cond: (med_record.person_id = unit_profile.person_id)
6	-> Bitmap Heap Scan on med_record (cost=13.16..548.63 rows=629 width=43) (actual time=0.155..0.558 rows=643
7	Recheck Cond: (date_to < '1925-01-01'::date)
8	Heap Blocks: exact=368
9	-> Bitmap Index Scan on i2 (cost=0.00..13.01 rows=629 width=0) (actual time=0.115..0.115 rows=643 loop
10	Index Cond: (date_to < '1925-01-01'::date)
11	-> Hash (cost=7.00..7.00 rows=200 width=8) (actual time=0.097..0.097 rows=200 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 16kB
13	-> Seq Scan on unit_profile (cost=0.00..7.00 rows=200 width=8) (actual time=0.013..0.056 rows=200 loop
14	-> Index Only Scan using person_pk on operator (cost=0.27..0.57 rows=1 width=4) (actual time=0.004..0.004 rows=1 l
15	Index Cond: (operator_id = unit_profile.operator_id)
16	Heap Fetches: 4
17	-> Index Scan using person_file_pk on person (cost=0.29..3.01 rows=1 width=16) (actual time=0.004..0.004 rows=1 loops=4)
18	Index Cond: (person_id = med_record.person_id)
19	Planning Time: 1.619 ms
20	Execution Time: 0.820 ms

Рис 24. План запроса

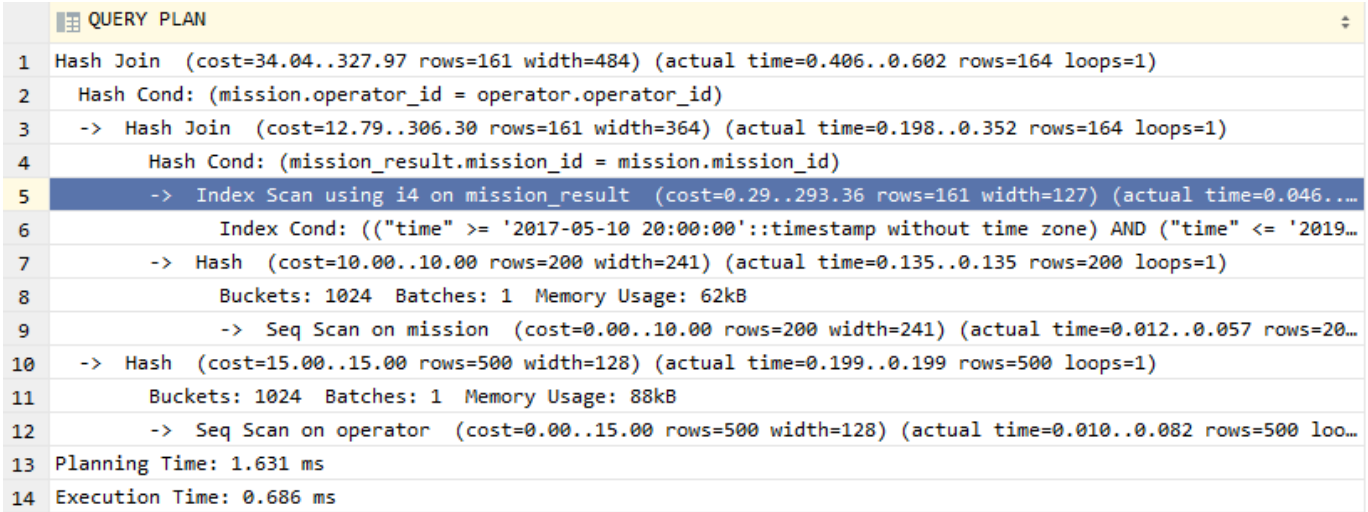
План запроса 3:



	QUERY PLAN
1	Hash Join (cost=28.40..198.30 rows=144 width=219) (actual time=0.186..0.186 rows=0 loops=1)
2	Hash Cond: (agent.pack_id = pack.pack_id)
3	-> Hash Join (cost=16.90..186.42 rows=143 width=126) (actual time=0.068..0.068 rows=0 loops=1)
4	Hash Cond: (agent.pack_id = item.pack_id)
5	-> Bitmap Heap Scan on agent (cost=5.40..172.23 rows=144 width=21) (actual time=0.068..0.068 rows=0 loops=1)
6	Recheck Cond: ((name)::text = 'Стаф'::text)
7	-> Bitmap Index Scan on i3 (cost=0.00..5.37 rows=144 width=0) (actual time=0.066..0.066 rows=0 loops=1)
8	Index Cond: ((name)::text = 'Стаф'::text)
9	-> Hash (cost=9.00..9.00 rows=200 width=105) (never executed)
10	-> Seq Scan on item (cost=0.00..9.00 rows=200 width=105) (never executed)
11	-> Hash (cost=9.00..9.00 rows=200 width=105) (actual time=0.098..0.098 rows=200 loops=1)
12	Buckets: 1024 Batches: 1 Memory Usage: 36kB
13	-> Seq Scan on pack (cost=0.00..9.00 rows=200 width=105) (actual time=0.013..0.048 rows=200 loops=1)
14	Planning Time: 1.791 ms
15	Execution Time: 0.264 ms

Рис 25. План запроса

План запроса 4:



	QUERY PLAN
1	Hash Join (cost=34.04..327.97 rows=161 width=484) (actual time=0.406..0.602 rows=164 loops=1)
2	Hash Cond: (mission.operator_id = operator.operator_id)
3	-> Hash Join (cost=12.79..306.30 rows=161 width=364) (actual time=0.198..0.352 rows=164 loops=1)
4	Hash Cond: (mission_result.mission_id = mission.mission_id)
5	-> Index Scan using i4 on mission_result (cost=0.29..293.36 rows=161 width=127) (actual time=0.046.…
6	Index Cond: (("time" >= '2017-05-10 20:00:00'::timestamp without time zone) AND ("time" <= '2019…
7	-> Hash (cost=10.00..10.00 rows=200 width=241) (actual time=0.135..0.135 rows=200 loops=1)
8	Buckets: 1024 Batches: 1 Memory Usage: 62kB
9	-> Seq Scan on mission (cost=0.00..10.00 rows=200 width=241) (actual time=0.012..0.057 rows=20…
10	-> Hash (cost=15.00..15.00 rows=500 width=128) (actual time=0.199..0.199 rows=500 loops=1)
11	Buckets: 1024 Batches: 1 Memory Usage: 88kB
12	-> Seq Scan on operator (cost=0.00..15.00 rows=500 width=128) (actual time=0.010..0.082 rows=500 loo…
13	Planning Time: 1.631 ms
14	Execution Time: 0.686 ms

Рис 26. План запроса

План запроса 5:

```
Hash Join  (cost=287.25..831.06 rows=374 width=130) (actual time=0.692..5.050 rows=366 loops=1)
  Hash Cond: (mission.operator_id = operator.operator_id)
  ->  Hash Join  (cost=266.00..808.82 rows=374 width=134) (actual time=0.533..4.776 rows=366 loops=1)
        Hash Cond: (agent_mission.mission_id = mission.mission_id)
        ->  Hash Join  (cost=253.50..795.32 rows=374 width=33) (actual time=0.422..4.551 rows=366 loops=1)
              Hash Cond: (agent.agent_id = agent_mission.agent_id)
              ->  Seq Scan on agent  (cost=0.00..363.04 rows=20004 width=21) (actual time=0.010..2.550 rows=20004 loops=1)
              ->  Hash  (cost=248.83..248.83 rows=374 width=20) (actual time=0.388..0.388 rows=366 loops=1)
                    Buckets: 1024  Batches: 1  Memory Usage: 27kB
                    ->  Bitmap Heap Scan on agent_mission  (cost=11.18..248.83 rows=374 width=20) (actual time=0.107..0.320 rows=366 loops=1)
                          Recheck Cond: (date_to > '2004-01-01'::date)
                          Heap Blocks: exact=182
                          ->  Bitmap Index Scan on i5  (cost=0.00..11.09 rows=374 width=0) (actual time=0.086..0.086 rows=366 loops=1)
                                Index Cond: (date_to > '2004-01-01'::date)
        ->  Hash  (cost=10.00..10.00 rows=200 width=109) (actual time=0.103..0.103 rows=200 loops=1)
              Buckets: 1024  Batches: 1  Memory Usage: 37kB
              ->  Seq Scan on mission  (cost=0.00..10.00 rows=200 width=109) (actual time=0.009..0.055 rows=200 loops=1)
  ->  Hash  (cost=15.00..15.00 rows=500 width=4) (actual time=0.150..0.150 rows=500 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 26kB
        ->  Seq Scan on operator  (cost=0.00..15.00 rows=500 width=4) (actual time=0.013..0.076 rows=500 loops=1)
Planning Time: 2.377 ms
Execution Time: 5.186 ms
```

Рис 27. План запроса

Графики (40000 записей):

Для константного количества потоков (1):

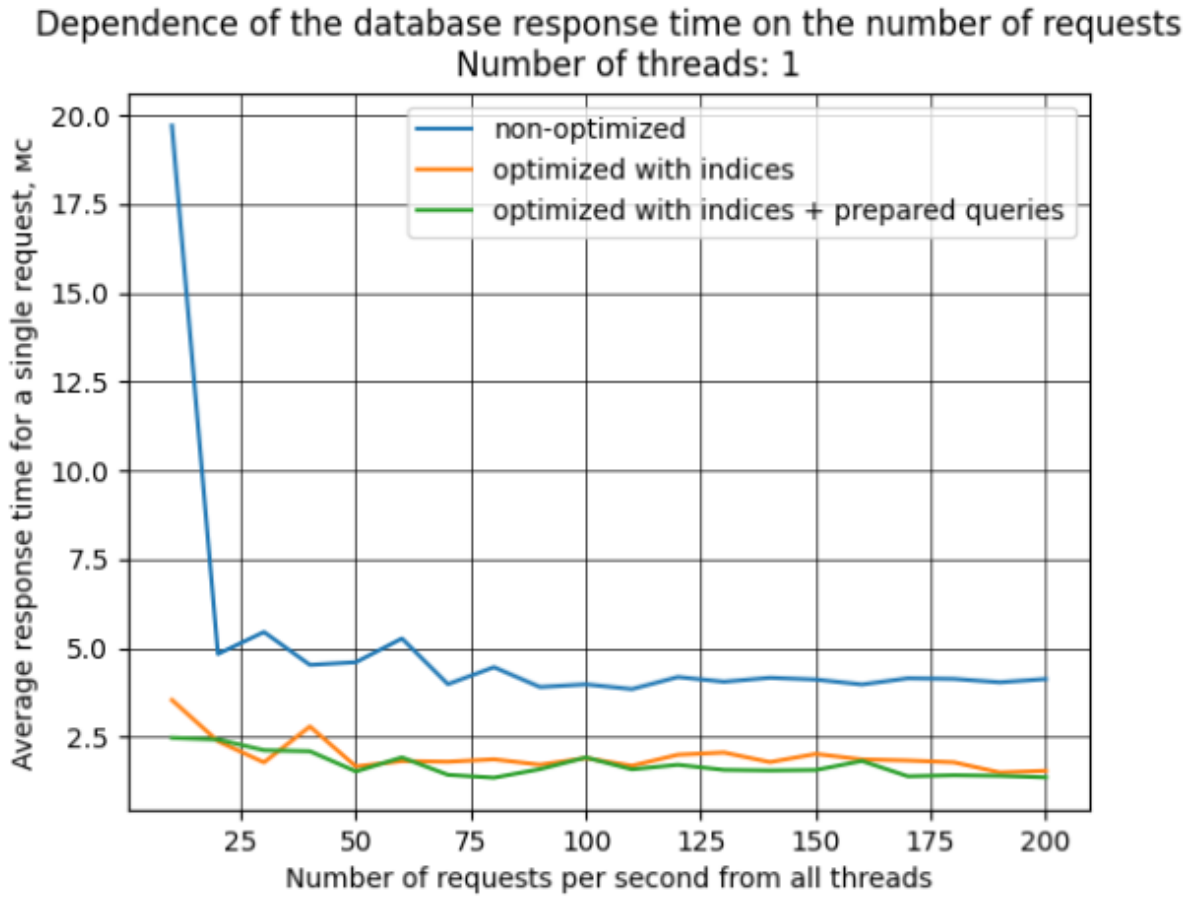


Рис 28. Эксперимент 1.

Для константного количества потоков (3):

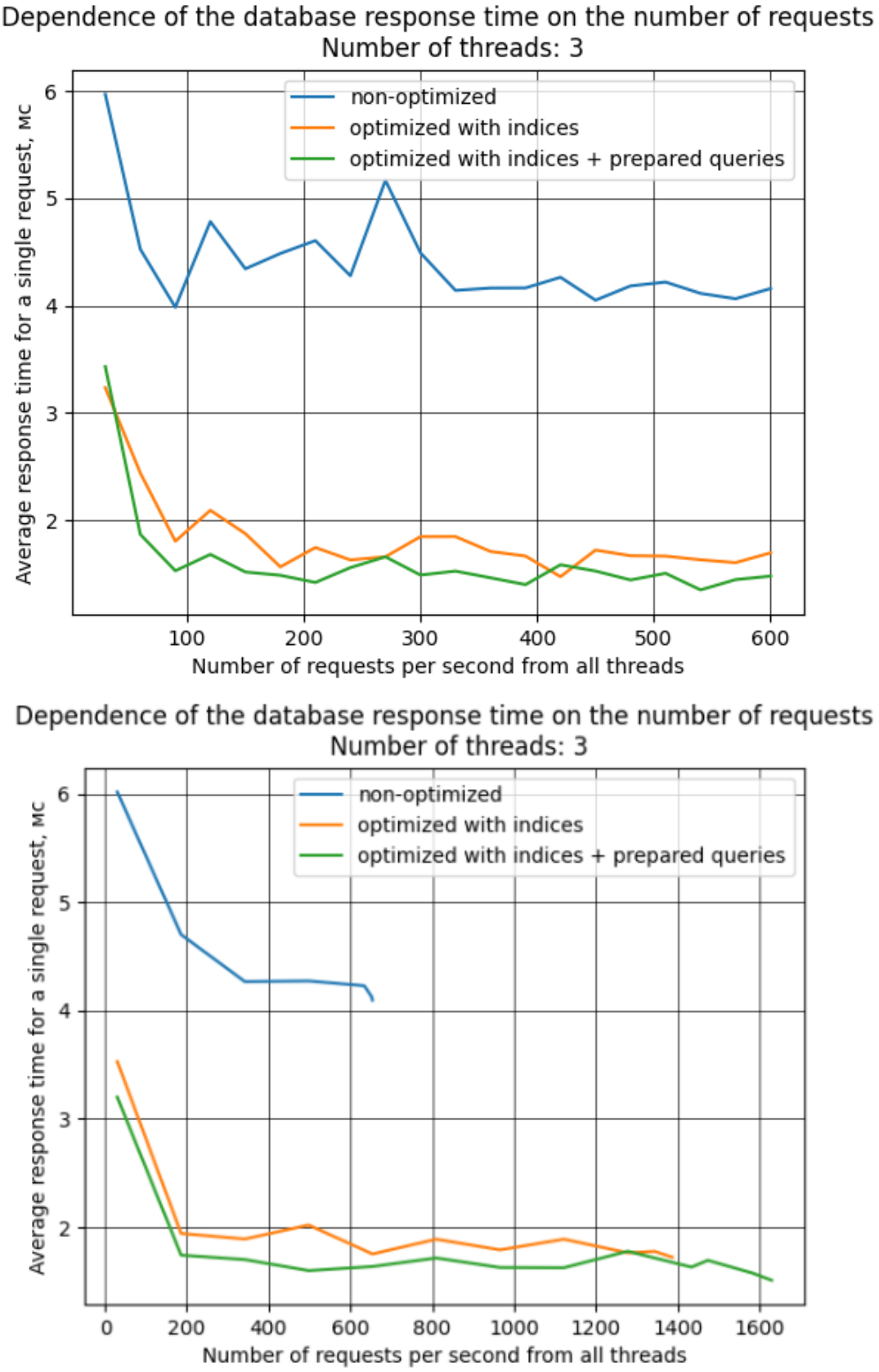
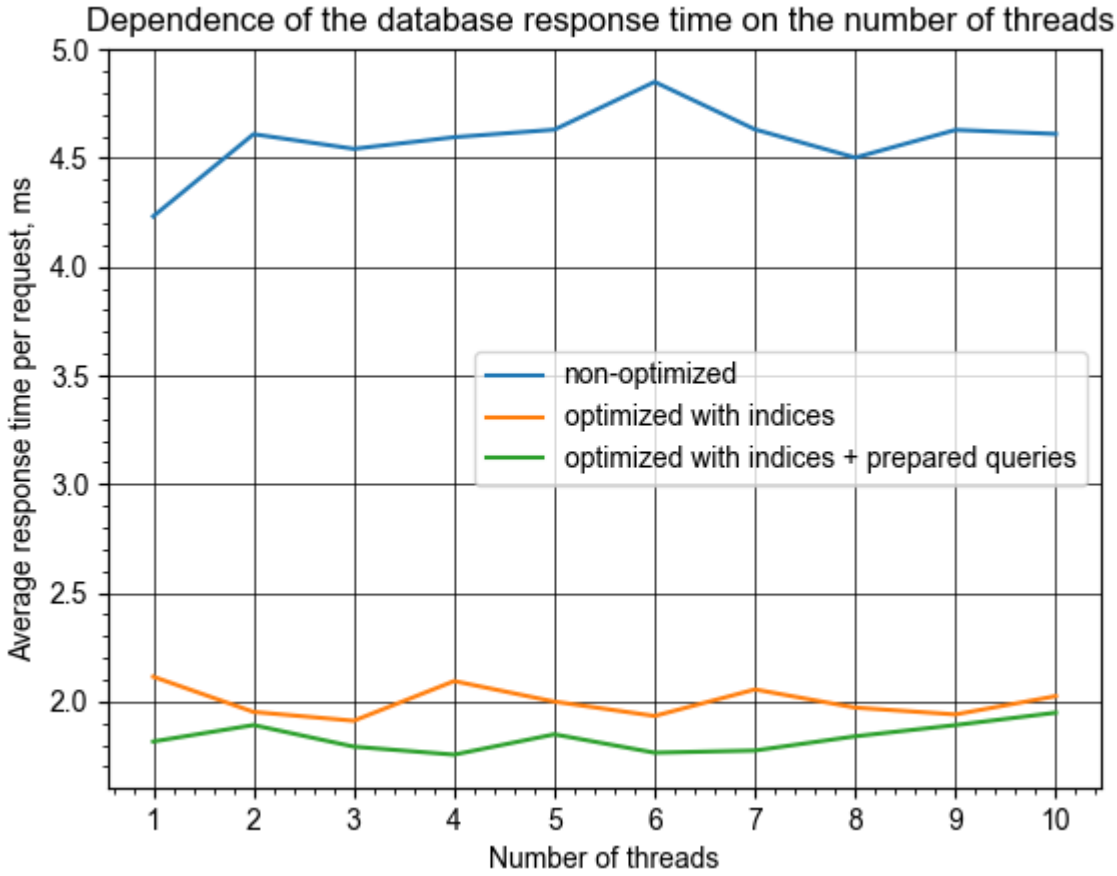
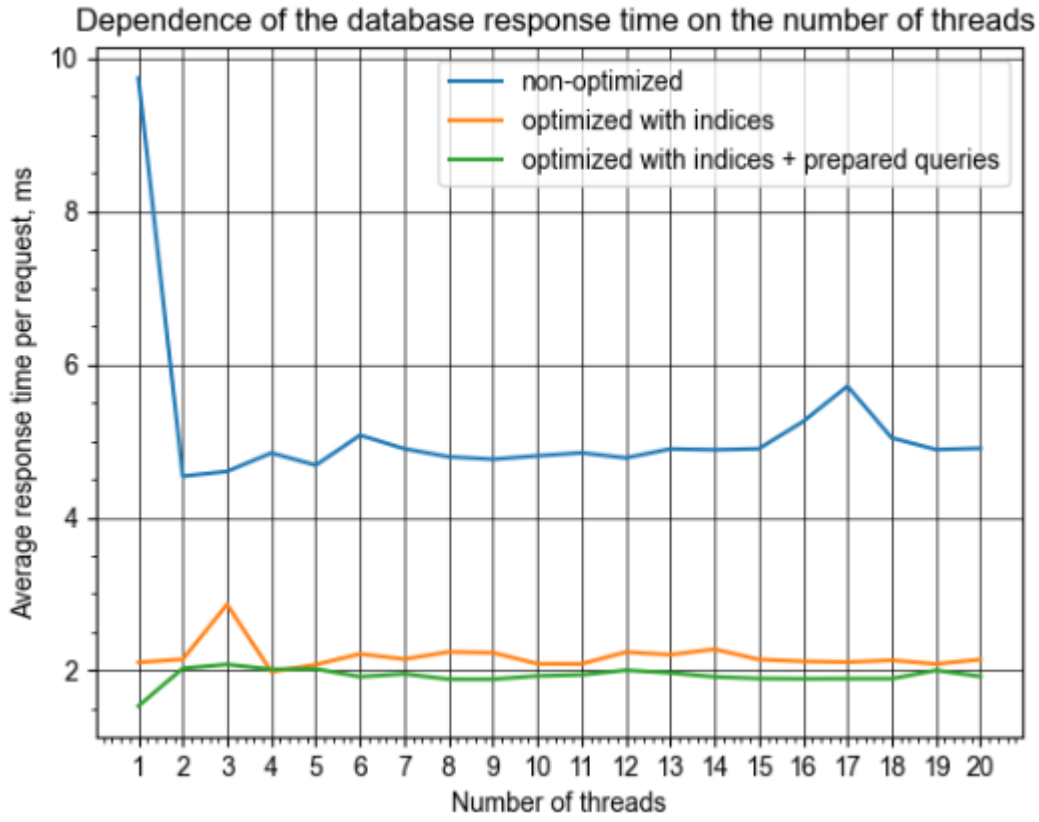


Рис 29. Эксперимент 2.

Для динамического количества потоков:





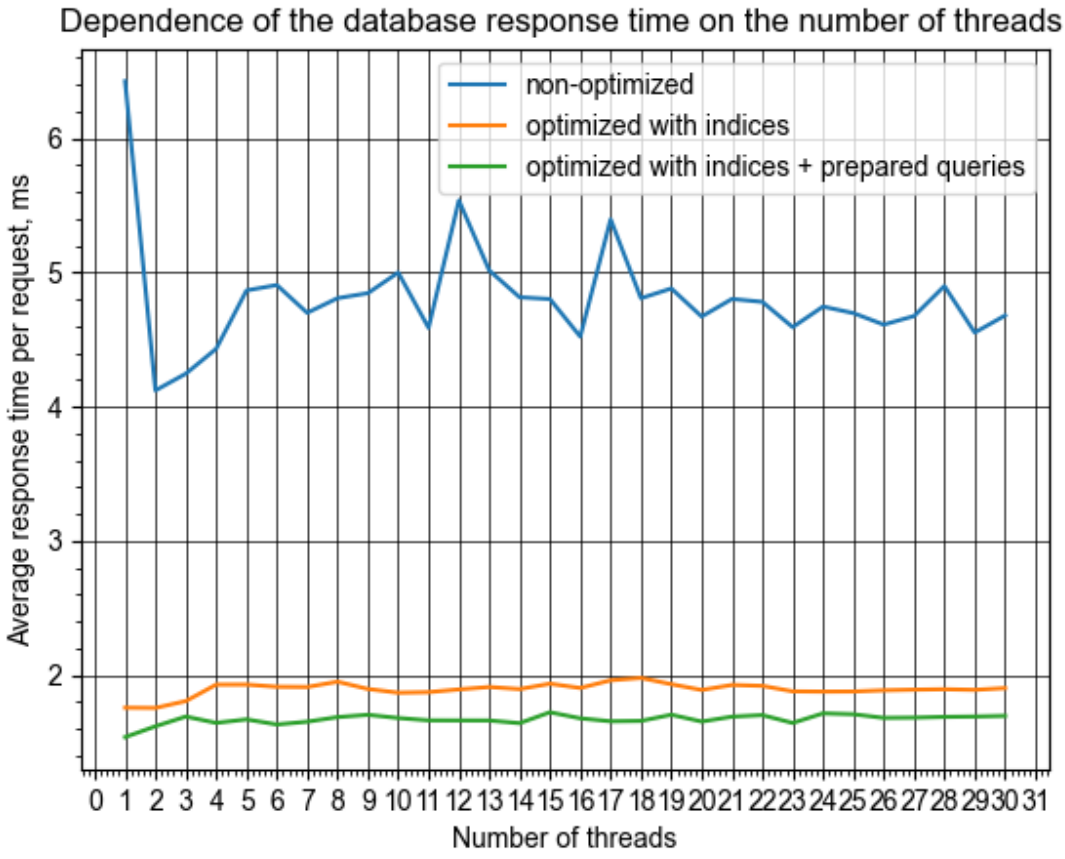

Dependence of the database response time on the number of threads
non-optimized
optimized with indices
optimized with indices + prepared queries
Average response time per request, ms
2
3
4
5
6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Number of threads

Рис 30. Эксперимент 3.

Графики (160000 записей)

(--truncate 1 --person 30000 --operator 1000 --pack 2000 --mission 2000 --mission_result 10 --agent_mission 30000 --agent 30000 --unit_profile 30000 --item 1000):

Для константного количества потоков (1):

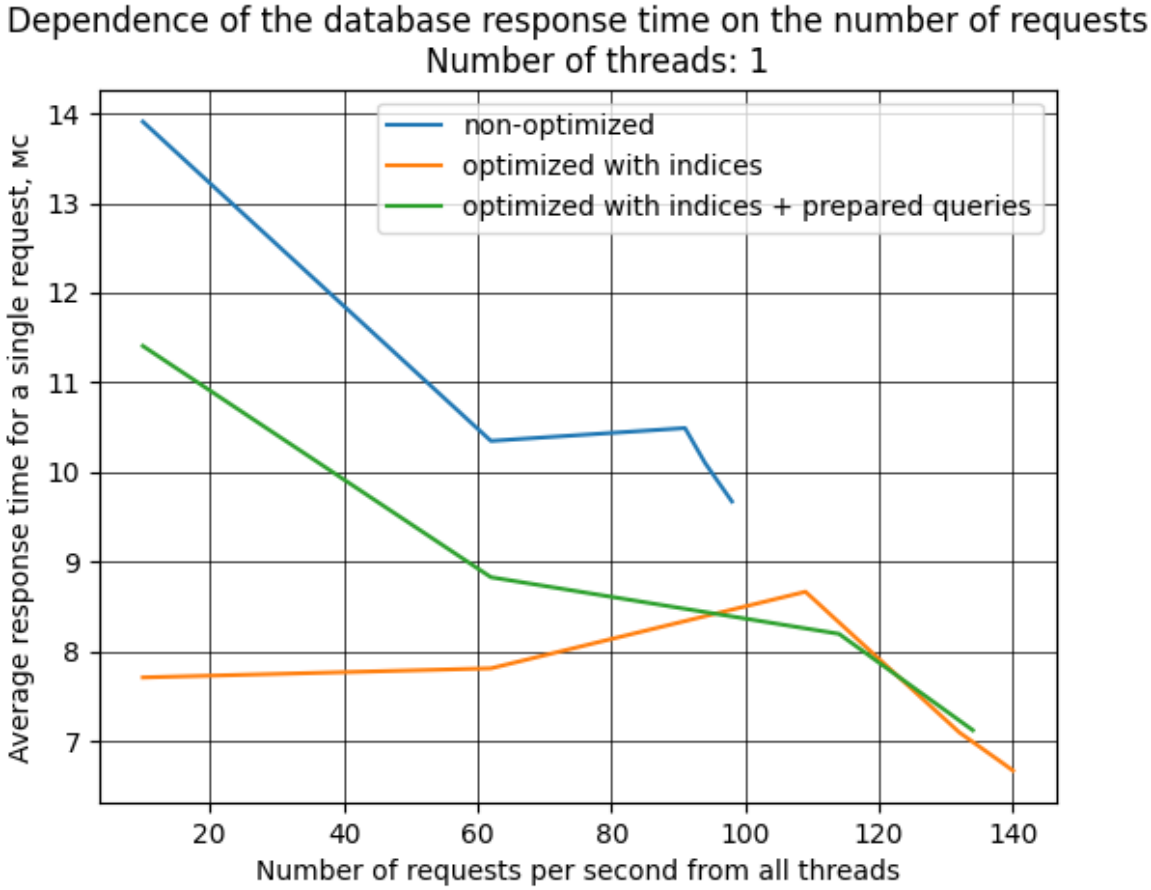


Рис 31. Эксперимент 1.

Для константного количества потоков (3):

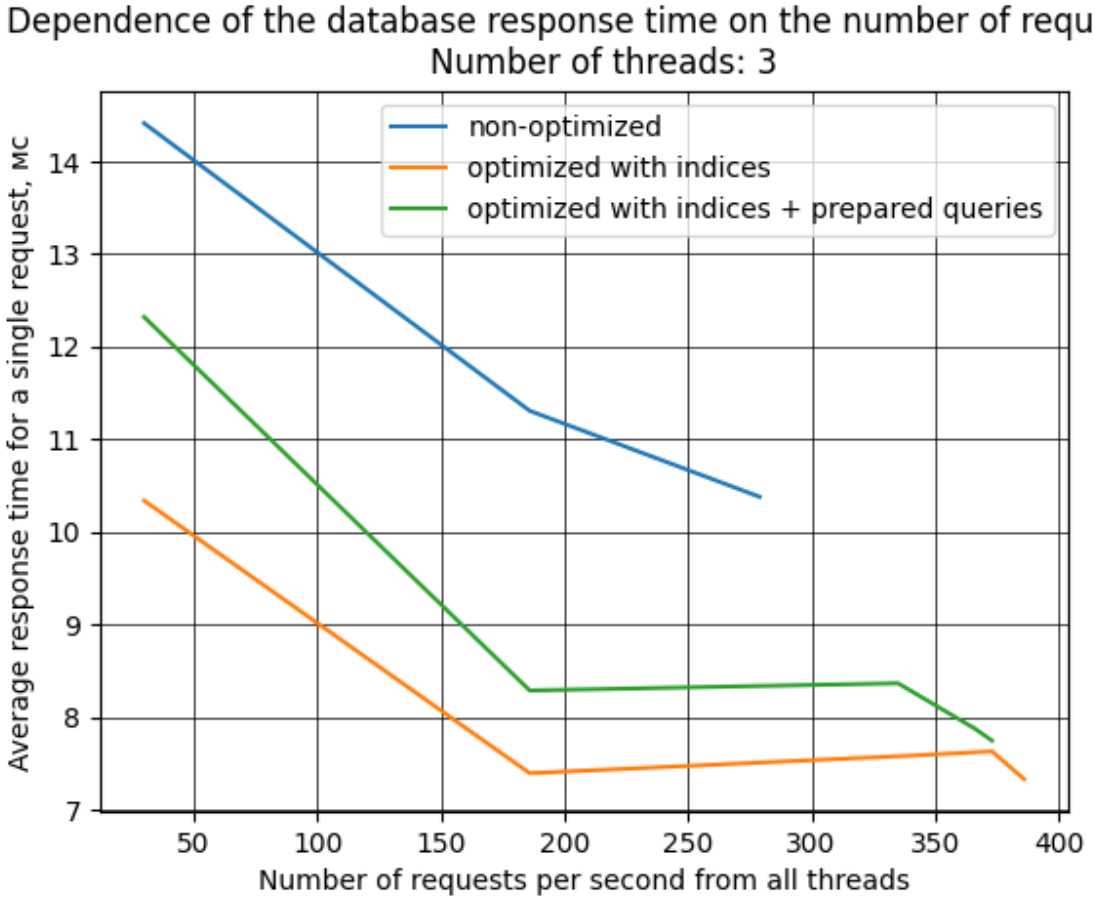


Рис 32. Эксперимент 2.

Для константного количества потоков (5):

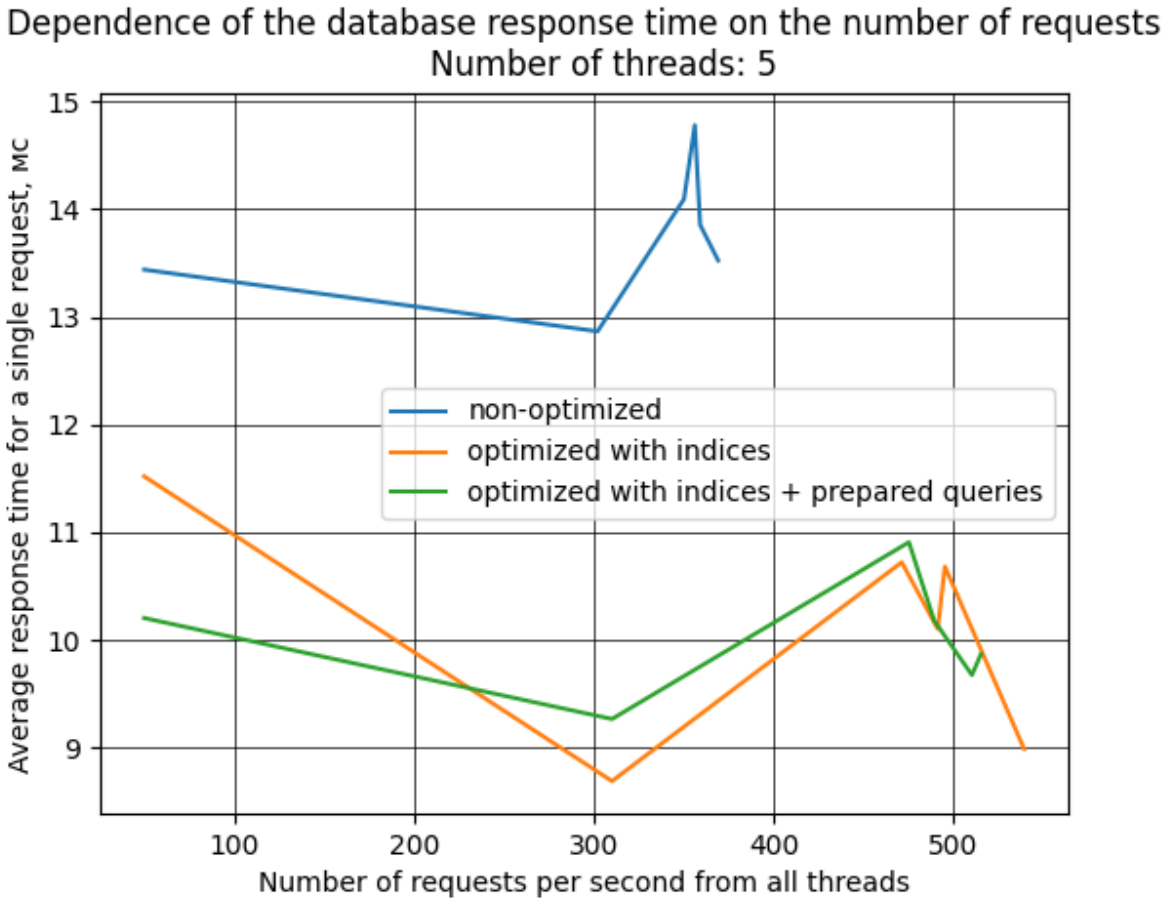


Рис 33. Эксперимент 3.

Для динамического количества потоков:

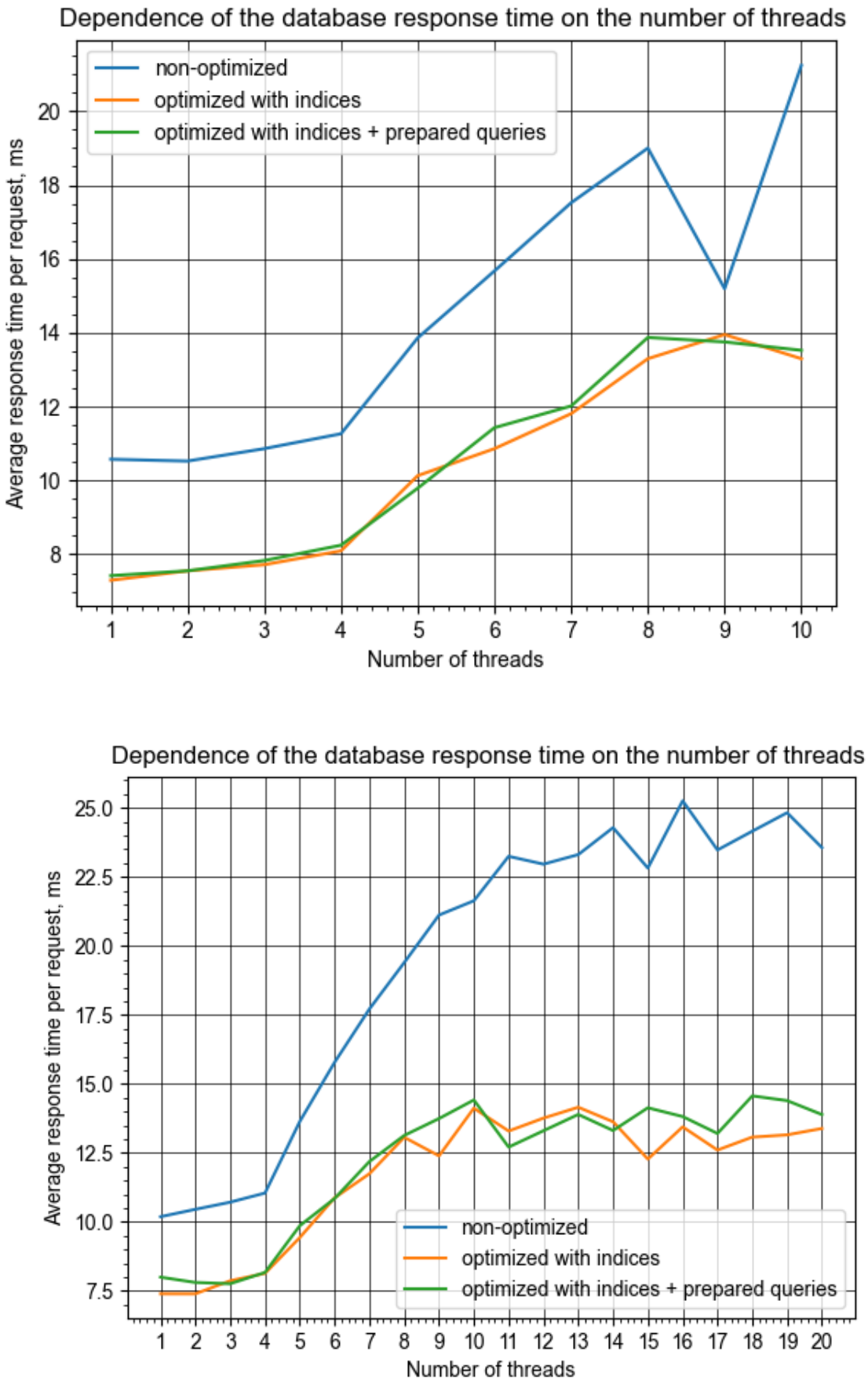


Рис 33. Эксперимент 4.

В ходе проведения экспериментов было обнаружено, что при очень больших количествах записей в базе данных использование подготовленных операторов перестает давать прибавки в производительности. Это происходит потому, что на подготовку запроса тратится очень большое количество времени, большее чем требуется на исполнение запроса. Отсюда мы и не получаем прибавку в производительности.

С другой стороны, при относительно малом количестве записей в БД – меньше 1000 на таблицу – практически теряет смысл использование индексации, поскольку сканирование индексов только увеличивает время работы запроса, не помогая, в отличие от других случаев, ускорить фильтрацию большого объема данных.

Также, стоит отметить, что на результаты экспериментов влияет не только количество данных в базе и параметры запуска, но и разброс при случайном выборе выполняемого запроса, а также фоновые процессы на машине, на которой проводились эксперименты.

Итог.

Было разработано приложение, код находится по ссылке: https://gitlab.icc.spbstu.ru/mermaider/db-spring-2020-spybase/-/tree/master/4

Приложение умеет работать в двух режимах: построение зависимости времени ответа БД от количества запросов в секунду (далее первый) и построение зависимости времени ответа БД от количества соединений, выполняющих запросы (далее второй). В каждом режиме сначала зависимость снимается на неоптимизированных запросах, затем создаются индексы, затем подготавливаются операторы. Все три зависимости выводятся на одном графике.

В первом режиме пользователь задаёт количество потоков, осуществляющие запросы к базе данных, минимальное и максимальное количество запросов в секунду на один поток, а также время моделирования в секундах. Промежуток от минимального до максимального кол-ва запросов в секунду делится на количество секунд, далее потоки пытаются выполнить переданное им количество запросов, со временем наращивая нагрузку вплоть до максимального количества запросов. Выполнение заданного количества запросов не гарантируется и ограничено вычислительными мощностями машины. На каждом шаге производится подсчёт среднего времени ответа БД и общего количества запросов, осуществлённых со всех потоков.

Во втором режиме пользователь задаёт минимальное и максимальное кол-во потоков, а также кол-во запросов на каждый поток. На каждой итерации работы приложения количество соединений увеличивается на 1 и проходит все значения от минимального до максимального количества потоков. Каждый поток выполняется заданное количество запросов (без ограничений по времени). На каждом шаге производится подсчёт среднего времени ответа БД.