РЕФЕРАТ

Расчетно-пояснительная записка 37 с., 7 рис., 16 табл., 16 ист.

БАЗА ДАННЫХ, АВТОПАРКОВКА, POSTGRESQL, ИНДЕКС БАЗЫ ДАННЫХ, СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ.

Цель работы – разработка базы данных и приложения для сети автопарковок.

В рамках курсовой работы был проведен анализ предметной области автопарковок, формализованы бизнес-правила, спроектирована и разработана база данных автопарковок и приложение для доступа к ней. Кроме того, было проведено исследование зависимости времени выполнения запроса от наличия индекса, его типа и количества записей в базе данных. Также, исследовалась зависимость объема требуемой памяти для хранения индекса от его типа и количества записей.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность курсовой работы определяется постоянным ростом числа транспортных средств и острой нехваткой парковочных мест [1]. Поиск свободных парковок не только доставляет неудобство автовладельцам, но и приводит к загруженности дорог и аварийным ситуациям. Возможность получить информацию о наличии свободных мест, а также возможность забронировать заранее парковочное место значительно облегчает планирование передвижений жителей городов и мегаполисов.

Целью курсовой работы является разработка базы данных и приложения для сети автопарковок.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) провести анализ предметной области;
2) формализовать требования к создаваемой системе;
3) спроектировать базу данных и приложение для доступа к ней;
4) реализовать спроектированное программное обеспечение;
5) исследовать зависимость времени выполнения запроса и занимаемой индексом памяти от типа индекса и объема записей базы данных.

1 Аналитический раздел

В данном разделе приводится анализ предметной области и формализуются данные и задачи программного обеспечения. Также приводится анализ моделей баз данных.

1.1 Анализ предметной области

Создание парковочных мест является неотъемлемой частью организации дорожно-транспортной системы. Один из самых удобных способов решения этой задачи – автоматические автопарковки [2]. Использование такого типа парковок позволяет не только снизить затраты на организацию инфраструктуры парковок, но и обеспечить круглосуточный сервис для автовладельцев.

Въезд и выезд на такие парковки осуществляется при помощи талонов или RFID-карт. Оплачивается же время парковки на специальных устройствах – паркоматах, поддерживающих оплату как наличными, так и банковской картой.

Как правило, на въезде на парковку стоит табло с указанием количества свободных мест. На самой парковки ведется видеонаблюдение, что повышает безопасность припаркованного транспорта.

1.2 Анализ существующих решений

Разработка приложения для сети автопарковок позволяет сделать процесс парковки более удобным. В частности, с помощью приложения возможно организовать бронь парковочного места, получить информацию о количестве свободных мест, оформить абонемент, оплатить время парковки с помощью QR-кода. Используя перечисленные возможности как критерии, был проведен анализ существующих решений, таких как:

— Came Vector [3];

— Квазар [4];

— Московский паркинг [5].

Результаты анализа приведены в таблице 1.1.

Таблица 1.1 – Сравнение известных решений

	Московский паркинг	Came Vector	Квазар
Возможность брони	нет	нет	нет
Информация о количестве свободных мест	да	да	нет
Оформление абонемента	да	нет	да
Оплата с помощью QR-кода	нет	нет	нет

Как видно из результатов анализа, ни одно из существующих решений не удовлетворяет всем требованиям.

1.3 Формализация задача

В рамках курсовой работы необходимо разработать базу данных для хранения информации о парковках и приложение для получения и обработки информации, хранящейся в ней. В приложении необходимо реализовать следующих акторов: пользователь, гость (незарегистрированный пользователь), администратор парковки, паркомат (устройство для фиксации начала и окончания парковки). Диаграмма использования приложения приведена на рисунке 1.1.

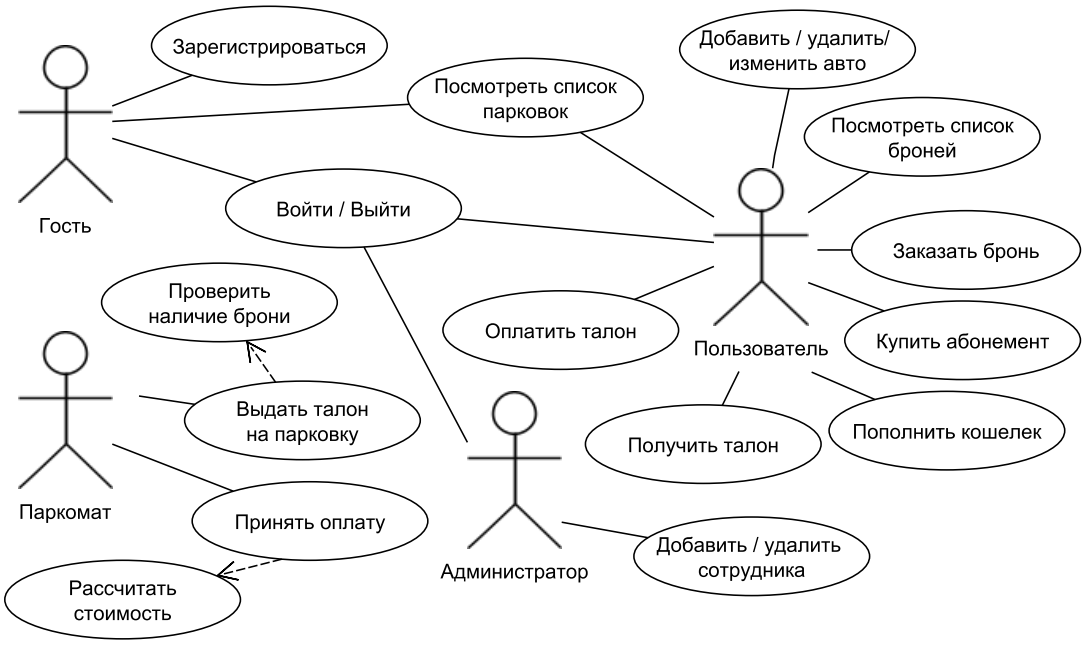

Рисунок 1.1 — Диаграмма использования приложения

1.4 Формализация данных

В результате анализа предметной области и формализации задачи был сформирован набор сущностей. Информация о них представлена в таблице 1.2.

Таблица 1.2 – Сущности базы данных

Сущность	Атрибуты
Парковка	Адрес, количество мест (всего и свободных), действующий тариф
Автовладелец	ФИО, текущий абонемент, состояние счета
Бронь	Статус, даты начала и окончания, автовладелец, парковка
Парковочный талон	Статус, даты начала и окончания, автовладелец, парковка
Транспорт	Номер, марка
Абонемент	Название, стоимость, срок действия, бонусы
Тариф	Название, цена
Паркомат	Парковка, номер въезда / выезда
Сотрудник	ФИО, должность
Должность	Наименование, оклад

На рисунке 1.2 приведена ER-диаграмма сущностей в нотации Чена.

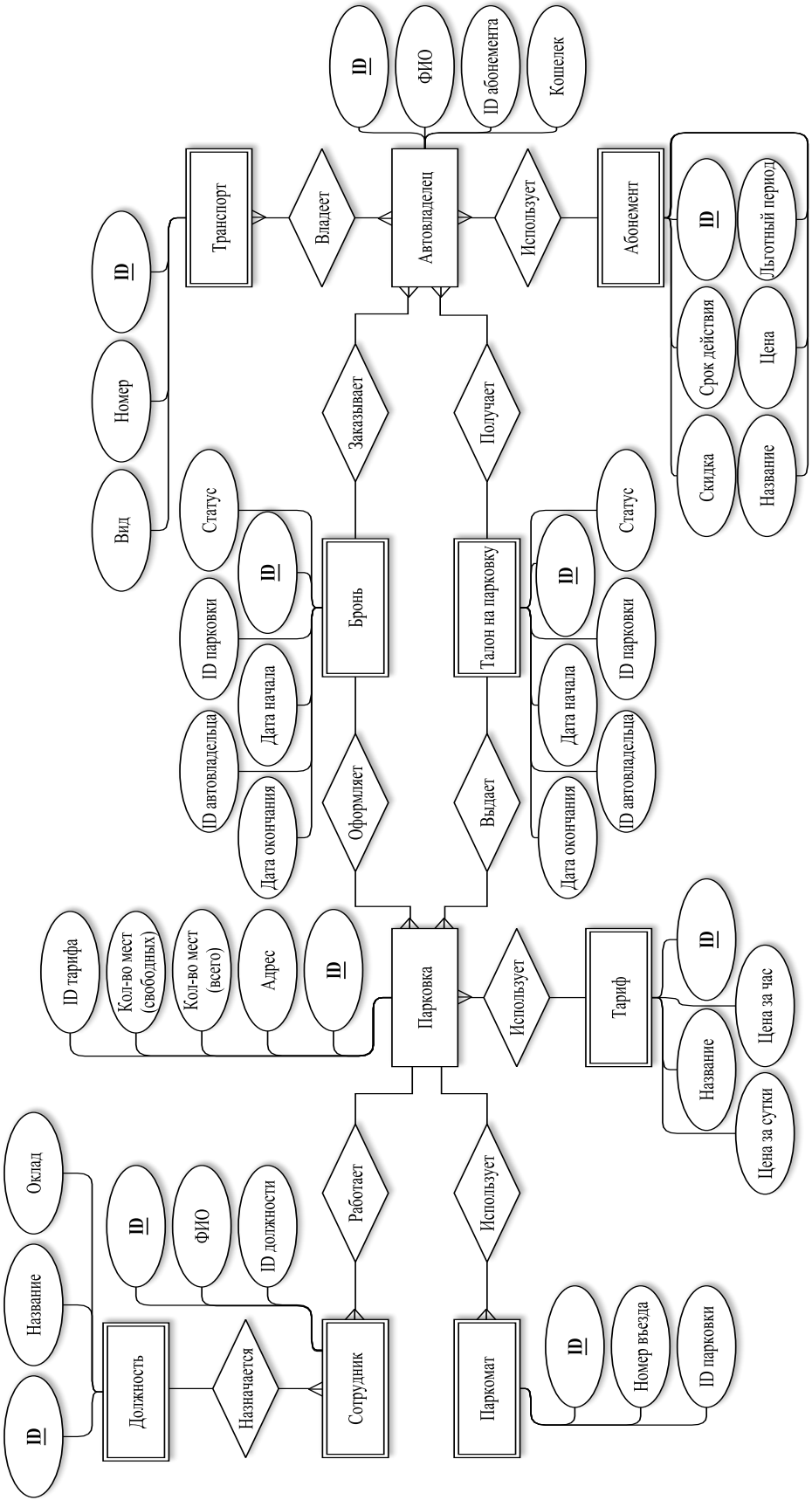

Рисунок 1.2 — ER-диаграмма сущностей в нотации Чена

1.5 Анализ баз данных

База данных – это некоторый набор перманентных данных, используемых прикладными программными системами какого-либо предприятия [6].

По модели хранения базы данных делятся на три группы:

— дореляционные

— реляционные

— постреляционные.

Дореляционные базы данных

Дореляционные модели близки к управлению данными во внешней памяти на низком уровне, что приводит к наилучшему использованию памяти, но такие модели данных имеют сложность использования. Так как логика процедуры выбора данных зависит от физической организации этих данных, то данная модель не является полностью независимой от приложения и как следствие приводит к усложнению действий изменений в базе данных [7].

Реляционные базы данных

Данные реляционных баз хранятся в виде таблиц, которые могут иметь связи с другими таблицами через внешние ключи, таким образом образуя отношения [7].

Структура реляционных баз данных позволяет связывать информацию из разных таблиц с помощью внешних ключей, которые используются для уникальной идентификации любого атомарного фрагмента данных в этой таблице. Другие таблицы могут ссылаться на этот внешний ключ.

Постреляционные базы данных

Классическая реляционная модель предполагает неделимость данных, хранящихся в полях записей таблиц. Это означает, что информация в таблице представляется в первой нормальной форме. Существует ряд случаев, когда это ограничение мешает эффективной реализации приложений. Постреляционная база данных использует трехмерные структуры, позволяя хранить в полях таблицы другие таблицы, расширяя таким образом возможности по описанию сложных объектов реального мира [8].

Вывод

В данном разделе был проведен анализ предметной области и существующих, формализованы задача и используемые данные. Также был проведен анализ баз данных, в результате которого было решено использовать реляци-

онную базу данных. Такой выбор обусловлен необходимостью использования разнообразных запросов различной сложности.

2 Конструкторский раздел

В данном разделе приведена диаграмма проектируемой базы данных, описана ролевая модель и хранимая процедура базы данных.

2.1 Описание сущностей базы данных

Исходя из формализации данных, приведенной в подразделе 1.4, база данных должна состоять из следующих таблиц:

— parkings – таблица парковок;

— auto_owners – таблица автовладельцев (пользователей);

— bookings – таблица броней;

— tickets – таблица парковочных талонов;

— cars – таблица машин;

— subscriptions – таблица абонементов;

— tariffs – таблица парковочных тарифов;

— parking_meters – таблица паркоматов;

— employees – таблица сотрудников;

— jobs – таблица должностей;

На рисунке 2.1 представлена ER-диаграмма базы данных.

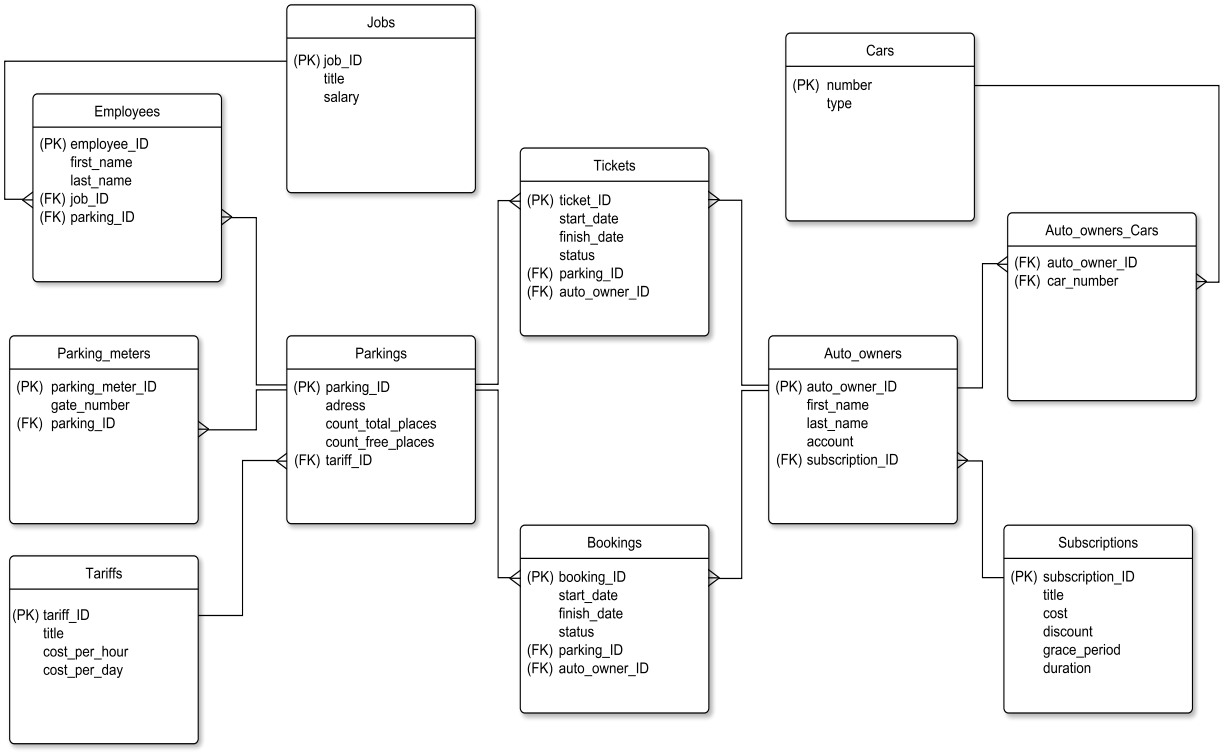

Рисунок 2.1 — ER-диаграмма базы данных

В связи с тем, что один и тот же автомобиль может принадлежать нескольким пользователям, и при этом у одного пользователя может быть несколько автомобилей, таблицы cars и auto_owners находятся в отношении многие-ко-многим. Для реализации такой связи была введена таблица-связка auto_owners_cars.

Информация о структуре и ограничениях каждой из перечисленных таблиц указана в таблицах 2.1 – 2.11.

Таблица 2.1 – Инормация о таблице auto_owners

Столбец	Тип данных	Ограничение	Описание
auto_owner_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор пользователя
first_name	VARCHAR(32)	NOT NULL	Имя
last_name	VARCHAR(32)	NOT NULL	Фамилия
account	INT	NOT NULL, ≥ 0	Счет
subscription_ID	INT	FOREIGN KEY	Идентификатор абонемента

Таблица 2.2 – Инормация о таблице auto_owner_cars

Столбец	Тип данных	Ограничение	Описание
auto_owner_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор пользователя
car_number	VARCHAR(32)	NOT NULL, FOREIGN KEY	Номер авто

Таблица 2.3 – Инормация о таблице cars

Столбец	Тип данных	Ограничение	Описание
number	VARCHAR(32)	NOT NULL, PRIMARY KEY	Номер авто
model	VARCHAR(32)	NOT NULL	Модель авто

Таблица 2.4 – Инормация о таблице subscriptions

Столбец	Тип данных	Ограничение	Описание
subscription_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор абонемента
title	VARCHAR(32)	NOT NULL	Название
discount	FLOAT	NOT NULL,≥ 0, < 1	Скидка
cost	INT	NOT NULL, > 0	Стоимость
grace_period	INT	NOT NULL,≥ 0	Льготные часы
duration	INTERVAL	NOT NULL	Срок действия абонемента

Таблица 2.5 – Инормация о таблице parkings

Столбец	Тип данных	Ограничение	Описание
parking_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор парковки
adress	VARCHAR(32)	NOT NULL	Адрес
cnt_total_places	INT	NOT NULL, > 0	Количество мест всего
cnt_free_places	INT	NOT NULL, ≥ 0	Количество свободных мест
tariff_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор тарифа

Таблица 2.6 – Инормация о таблице tariffs

Столбец	Тип данных	Ограничение	Описание
tariff_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор тарифа
title	VARCHAR(32)	NOT NULL	Название
cost_per_hour	INT	NOT NULL, ≥ 0	Стоимость часа

Таблица 2.7 – Инормация о таблице parking_meters

Столбец	Тип данных	Ограничение	Описание
parking_meter_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор паркомата
gate_number	INT	NOT NULL, ≥ 0	Номер въезда
parking_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор парковки

Таблица 2.8 – Инормация о таблице employees

Столбец	Тип данных	Ограничение	Описание
employee_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор сотрудника
first_name	VARCHAR(32)	NOT NULL	Имя
last_name	VARCHAR(32)	NOT NULL	Фамилия
job_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор должности
parking_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор парковки

Таблица 2.9 – Инормация о таблице jobs

Столбец	Тип данных	Ограничение	Описание
job_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор должности
title	VARCHAR(32)	NOT NULL	Название
salary	INT	NOT NULL, > 0	Оклад

Таблица 2.10 – Инормация о таблице bookings

Столбец	Тип данных	Ограничение	Описание
booking_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор брони
start_date	TIMESTAMP	NOT NULL	Дата начала
finish_date	TIMESTAMP	> start_date	Дата окончания
status	BOOLEAN	NOT NULL	Статус
parking_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор парковки
auto_owner_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор пользователя

Таблица 2.11 – Инормация о таблице tickets

Столбец	Тип данных	Ограничение	Описание
ticket_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор талона
start_date	TIMESTAMP	NOT NULL	Дата начала
finish_date	TIMESTAMP	> start_date	Дата окончания
status	BOOLEAN	NOT NULL	Статус
parking_ID	INT	NOT NULL, FOREIGN KEY	Идентификатор парковки
auto_owner_ID	INT	NOT NULL, PRIMARY KEY	Идентификатор пользователя

2.2 Ролевая модель

Согласно диаграмме использования приложения, представленной на рисунке 1.1, необходимо выделить следующие роли:

— гость – обладает правами только на просмотр таблиц parkings, tariffs, subscriptions;

— автовладелец – обладает правами на просмотр и изменение записей, связанных с ним, во всех таблицах, кроме employees, jobs и parking_meters;

— паркомат – обладает правами на просмотр всех таблиц, создание и обновление записей в таблицах bookings и tickets;

— администратор – обладает всеми правами доступа ко всем таблицам.

2.3 Хранимая процедура

В проектируемой базе данных таблицы cars и auto_owners связанны отношением многие-ко-многим. При создании нового авто необходимо обновить сразу две таблицы: таблицу cars и таблицу-связку auto_owners_cars. Для повешения надежности и вместе с тем упрощения процесса добавления нового авто, было решено написать хранимую процедуру insert_car, выполняющую эту задачу. При этом, гарантируется наличие пользователя с переданным id.

2.4 Диаграмма классов приложения

На рисунке 2.2 приведена UML-диаграмма классов.

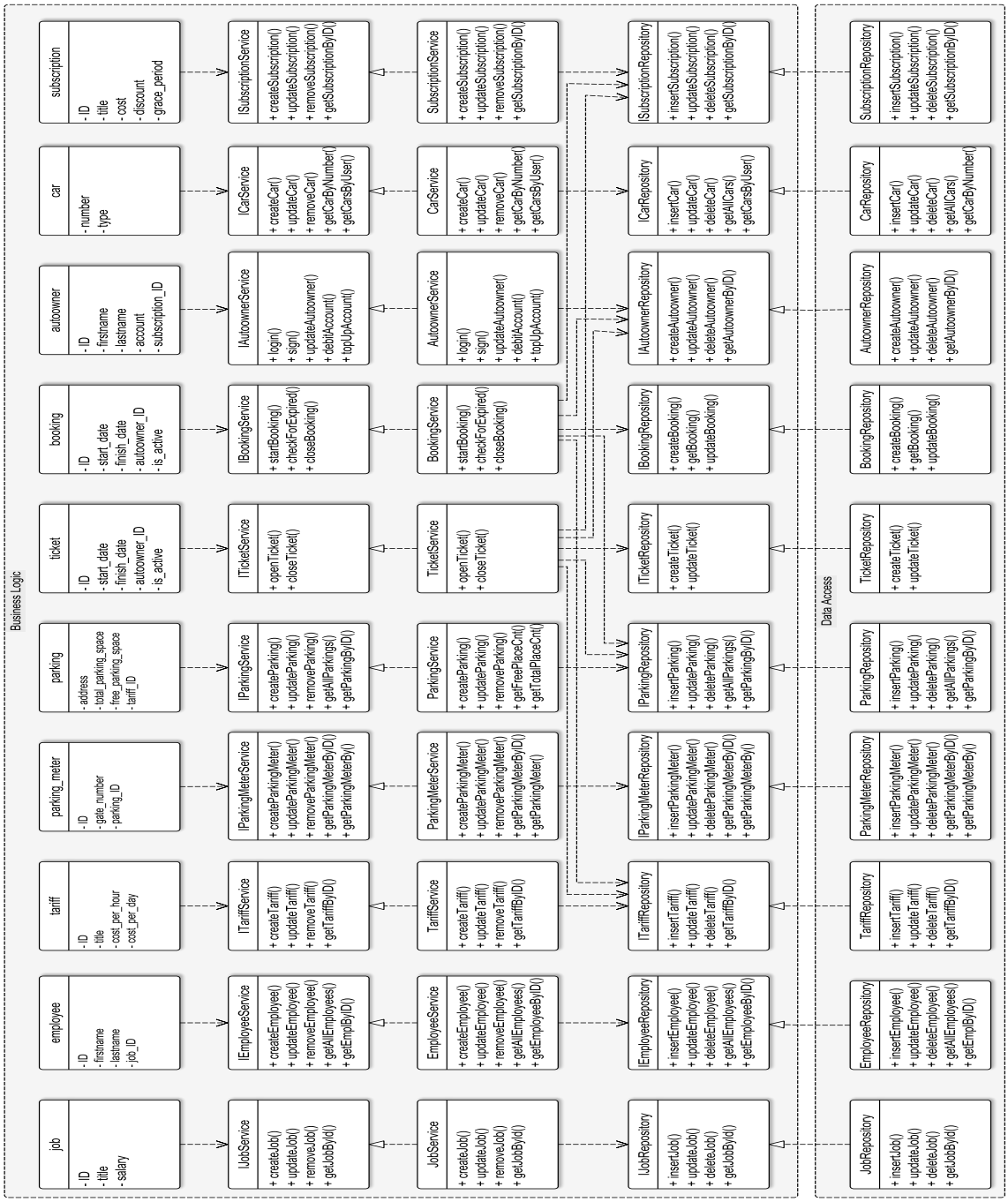

Рисунок 2.2 — UML-диаграмма классов

Проектирование велось в соответствии с принципами «чистой архитектуры» [9], в частности был спроектирован компонент бизнес-логики, полностью независимый от компонента доступа к данным и пользовательского интерфейса.

Вывод

В данном разделе были описаны сущности проектируемой базы данных, ролевая модель и хранимая процедура. Также приведена ER-диаграмма базы данных и UML-диаграмма классов.

3 Технологический раздел

В данном разделе приведен выбор средств реализации программного обеспечения, исходные коды создания таблиц и ролей базы данных.

3.1 Выбор системы управления базами данных

Были выбраны следующие критерии при выборе системы управления базами данных (СУБД):

1) СУБД должна быть бесплатной;
2) СУБД должна поддерживать процедурные расширения языка SQL;
3) СУБД должна работать на операционных системах Linux и Windows.

В таблице 3.1 приведено сравнение популярных СУБД.

Таблица 3.1 – Сравнение СУБД

СУБД	Критерий № 1	Критерий № 2	Критерий № 3
PostgreSQL [10]	+	+	+
MySQL [11]	+	-	+
Oracle [12]	-	+	+

В результате проведенного анализа в качестве СУБД была выбрана PostgreSQL.

3.2 Средства реализации

В качестве средств реализации программного обеспечения были выбраны следующие технологии:

— язык программирования – Java [13];
— СУБД – PostgreSQL;
— расширение языка SQL для написания процедуры – PL/pgSQL [14];
— для подключения к базе данных из приложения – Hibernate [15].

3.3 Создание базы данных

В листингах 3.1 и 3.2 приведены коды создания таблиц, описанных в разделе 2 и установки ограничений целостности соответственно.

Листинг 3.1 — Создание таблиц

```
CREATE TABLE IF NOT EXISTS jobs
(
    jobID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    title VARCHAR(64),
    salary int
);

CREATE TABLE IF NOT EXISTS employees
(
    employeeID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    first_name VARCHAR(16),
    last_name VARCHAR(16),
    jobFID INT,
    parkingFID INT
);

CREATE TABLE if NOT EXISTS parking_meters
(
    parking_meterID INT PRIMARY KEY GENERATED ALWAYS AS
       IDENTITY,
    gate_number INT,
    parkingFID INT
);

CREATE TABLE if NOT EXISTS tariffs
(
    tariffID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    title VARCHAR(32),
    cost_per_hour INT,
    cost_per_day INT
);

CREATE TABLE if NOT EXISTS parkings
(
    parkingID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    address VARCHAR(64),
    count_total_places INT,
    count_free_places INT,
    tariffFID INT
);
```

```
40
41 CREATE TABLE IF NOT EXISTS tickets
42 (
43     ticketID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
44     start_date TIMESTAMP,
45     finish_date TIMESTAMP,
46     is_active BOOLEAN,
47     parkingFID INT,
48     auto_ownerFID INT
49 );
50
51 CREATE TABLE IF NOT EXISTS bookings
52 (
53     bookingID INT PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
54     start_date TIMESTAMP,
55     finish_date TIMESTAMP,
56     is_active BOOLEAN,
57     parkingFID INT,
58     auto_ownerFID INT
59 );
60
61 CREATE TABLE IF not EXISTS auto_owners
62 (
63     auto_ownerID INT PRIMARY KEY GENERATED ALWAYS AS
          IDENTITY,
64     first_name VARCHAR(16),
65     last_name VARCHAR(16),
66     account INT,
67     login VARCHAR(32),
68     password_ VARCHAR(32),
69     date_subscription_expire TIMESTAMP,
70     subscriptionFID INT
71 );
72
73 CREATE TABLE IF NOT EXISTS subscriptions
74 (
75     subscriptionID INT PRIMARY KEY GENERATED ALWAYS AS
          IDENTITY,
76     title VARCHAR(32),
77     cost_ INT,
78     discount DOUBLE PRECISION,
```

```
79      period_ INTERVAL,
80      grace_hours INT
81  );
82
83  CREATE TABLE IF NOT EXISTS auto_owner_cars
84  (
85      auto_ownerFID INT,
86      car_number VARCHAR(16)
87  );
88
89  CREATE TABLE IF NOT EXISTS cars
90  (
91      car_number VARCHAR(16) PRIMARY KEY,
92      model VARCHAR(32)
93  );
```

Листинг 3.2 — Установка ограничений целостности

```
1   ALTER TABLE jobs
2       ALTER COLUMN title SET
3   NOT NULL,
4       ALTER COLUMN salary SET
5   NOT NULL,
6       ADD CONSTRAINT check_title CHECK (
7       title <> ''
8   ),
9       ADD CONSTRAINT check_salary CHECK (
10      salary > 0
11  );
12
13  ALTER TABLE employees
14      ALTER COLUMN first_name SET
15  NOT NULL,
16      ALTER COLUMN last_name SET
17  NOT NULL,
18      ALTER COLUMN jobFID SET
19  NOT NULL,
20      ALTER COLUMN parkingFID SET
21  NOT NULL,
22      ADD CONSTRAINT check_name CHECK (
23      first_name <> ''
24      AND
```

```
25      last_name <> ''
26  ),
27      ADD CONSTRAINT FK_job FOREIGN KEY (jobFID) REFERENCES
            jobs (jobID),
28      ADD CONSTRAINT FK_parking FOREIGN KEY (parkingFID)
            REFERENCES parkings (parkingID);
29
30  ALTER TABLE parking_meters
31      ALTER COLUMN gate_number SET
32  NOT NULL,
33      ALTER COLUMN parkingFID SET
34  NOT NULL,
35      ADD CONSTRAINT check_gate_number CHECK (
36      gate_number > 0
37  ),
38      ADD CONSTRAINT FK_parking FOREIGN KEY (parkingFID)
            REFERENCES parkings (parkingID);
39
40  ALTER TABLE tariffs
41      ALTER COLUMN title SET
42  NOT NULL,
43      ALTER COLUMN cost_per_hour SET
44  NOT NULL,
45      ALTER COLUMN cost_per_day SET
46  NOT NULL,
47      ADD CONSTRAINT check_cost CHECK (
48      cost_per_hour >= 0
49      AND cost_per_day >= 0
50  );
51
52  ALTER TABLE parkings
53      ALTER COLUMN address SET
54  NOT NULL,
55      ALTER COLUMN count_total_places SET
56  NOT NULL,
57      ALTER COLUMN count_free_places SET
58  NOT NULL,
59      ALTER COLUMN tariffFID SET
60  NOT NULL,
61      ADD CONSTRAINT check_count_places CHECK (
62      count_total_places > 0
```

```
63      AND count_free_places > 0
64      AND count_free_places <= count_total_places
65  ),
66      ADD CONSTRAINT FK_tariff FOREIGN KEY (tariffFID)
            REFERENCES tariffs (tariffID);
67
68  ALTER TABLE tickets
69      ALTER COLUMN start_date SET
70  NOT NULL,
71      ALTER COLUMN finish_date SET
72  NOT NULL,
73      ALTER COLUMN is_active SET
74  NOT NULL,
75      ALTER COLUMN parkingfid SET
76  NOT NULL,
77      ALTER COLUMN auto_ownerfid SET
78  NOT NULL,
79      ADD CONSTRAINT check_date CHECK (
80      start_date < finish_date
81  ),
82      ADD CONSTRAINT FK_parking FOREIGN KEY (parkingFID)
            REFERENCES parkings (parkingID),
83      ADD CONSTRAINT FK_auto_owner FOREIGN KEY (auto_ownerFID)
            REFERENCES auto_owners (auto_ownerID);
84
85  ALTER TABLE bookings
86      ALTER COLUMN start_date SET
87  NOT NULL,
88      ALTER COLUMN finish_date SET
89  NOT NULL,
90      ALTER COLUMN is_active SET
91  NOT NULL,
92      ALTER COLUMN parkingfid SET
93  NOT NULL,
94      ALTER COLUMN auto_ownerfid SET
95  NOT NULL,
96      ADD CONSTRAINT check_date CHECK (
97      start_date < finish_date
98  ),
99      ADD CONSTRAINT FK_parking FOREIGN KEY (parkingFID)
            REFERENCES parkings (parkingID),
```

```
100     ADD CONSTRAINT FK_auto_owner FOREIGN KEY (auto_ownerFID)
            REFERENCES auto_owners (auto_ownerID);
101
102 ALTER TABLE auto_owners
103     ALTER COLUMN first_name SET
104 NOT NULL,
105     ALTER COLUMN last_name SET
106 NOT NULL,
107     ALTER COLUMN account SET
108 NOT NULL,
109     ALTER COLUMN login SET
110 NOT NULL,
111     ALTER COLUMN password_ SET
112 NOT NULL,
113     ALTER COLUMN date_subscription_expire SET
114 NOT NULL,
115     ALTER COLUMN subscriptionFID SET
116 NOT NULL,
117     ADD CONSTRAINT check_name CHECK (
118     first_name <> ''
119     AND
120     last_name <> ''
121 ),
122     ADD CONSTRAINT check_account CHECK (
123     account >= 0
124 ),
125     ADD CONSTRAINT FK_subscription FOREIGN KEY
            (subscriptionFID) REFERENCES subscriptions
            (subscriptionID);
126
127 ALTER TABLE subscriptions
128     ALTER COLUMN title SET
129 NOT NULL,
130     ALTER COLUMN cost_ SET
131 NOT NULL,
132     ALTER COLUMN discount SET
133 NOT NULL,
134     ALTER COLUMN period_ SET
135 NOT NULL,
136     ALTER COLUMN grace_hours SET
137 NOT NULL,
```

```
138     ADD CONSTRAINT check_title CHECK (
139     title <> ''
140 ),
141     ADD CONSTRAINT check_cost_ CHECK (
142     cost_ >= 0
143 ),
144     ADD CONSTRAINT check_discount CHECK (
145     discount >= 0
146     AND discount <= 1
147 ),
148     ADD CONSTRAINT check_grace_hours CHECK (
149     grace_hours >= 0
150 );
151
152 ALTER TABLE cars
153     ALTER COLUMN model SET
154 NOT NULL,
155     ADD CONSTRAINT check_model CHECK (
156     model <> ''
157 );
158
159 ALTER TABLE auto_owner_cars
160     ALTER COLUMN auto_ownerFID SET
161 NOT NULL,
162     ALTER COLUMN car_number SET
163 NOT NULL,
164     ADD CONSTRAINT FK_auto_owner FOREIGN KEY (auto_ownerFID)
        REFERENCES auto_owners (auto_ownerID),
165     ADD CONSTRAINT FK_car_number FOREIGN KEY (car_number)
        REFERENCES cars (car_number);
```

В листинге 3.3 приведен код для создания хранимой процедуры, спроектированной в разделе 2.

Листинг 3.3 — Создание таблиц

```
1 CREATE OR REPLACE PROCEDURE insert_car(auto_ownerid INT,
    car_number VARCHAR(16), model VARCHAR(16))
2 AS $$
3 DECLARE
4     car_id INT;
5 BEGIN
6     INSERT INTO cars (model, car_number) VALUES (model,
```

```
        car_number);
7
8       INSERT INTO auto_owner_cars VALUES (car_number,
            auto_ownerid);
9   END;
10  $$ LANGUAGE plpgsql;
```

В листинге 3.4 приведен код для создания ролевой модели, спроектированной в разделе 2.

Листинг 3.4 — Создание ролевой модели

```
1  CREATE ROLE guest login;
2  GRANT SELECT ON auto_owners, parkings, tariffs,
       subscriptions TO guest;
3
4  CREATE ROLE user_ login;
5  GRANT SELECT ON auto_owners, parkings, tariffs,
       subscriptions,
6  cars, auto_owner_cars, bookings, tickets TO user_;
7  GRANT UPDATE, INSERT, DELETE ON  cars, auto_owner_cars TO
       user_;
8
9  CREATE ROLE parking_meter login;
10 GRANT SELECT ON auto_owners, parkings, tariffs,
       subscriptions,
11 cars, auto_owner_cars, bookings, tickets TO parking_meter;
12 GRANT UPDATE, INSERT, DELETE ON  cars, auto_owner_cars TO
       parking_meter;
13 GRANT UPDATE, INSERT ON bookings, tickets TO parking_meter;
14
15 CREATE ROLE admin_ login superuser;
```

3.4 Демонстрация работы программы

На рисунке 3.1 продемонстрирован интерфейс главного меню после успешной авторизации.

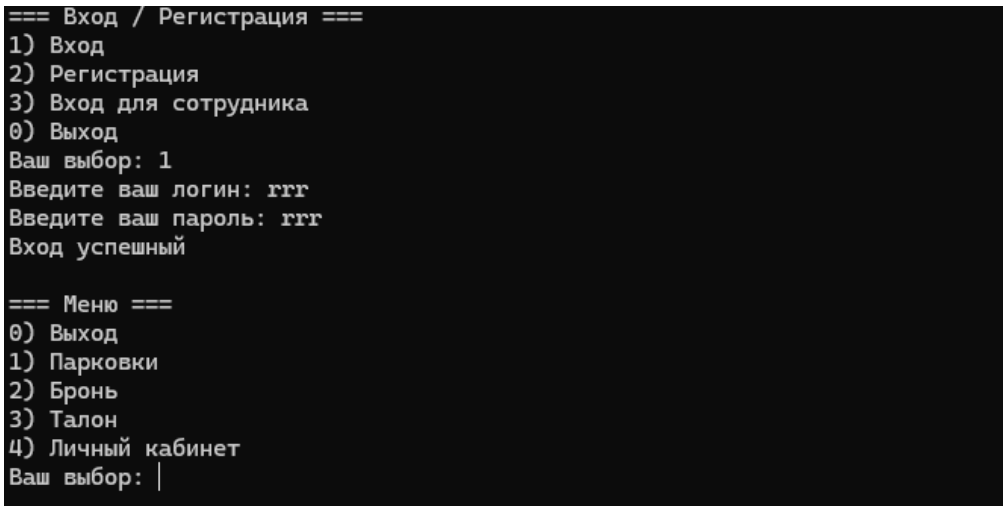

Рисунок 3.1 — Демонстрация работы программы

Вывод

В данном разделе были выбраны средства реализации программного обеспечения, а также приведены исходные коды создания таблиц и хранимой процедуры базы данных.

4 Исследовательская часть

В данном разделе приведено исследование и анализ полученных результатов, а также технические характеристики устройства, на котором осуществлялось исследование.

4.1 Технические характеристики

Технические характеристики устройства, на котором осуществлялось исследование:

— операционная система – Windows 10;

— оперативная память – 16 Гб;

— процессор – AMD Ryzen 7 4700U with Radeon Graphics;

— количество физических ядер – 8;

— количество логических ядер – 8.

Так как анализ проводился на ноутбуке, то для корректного замера времени ноутбук был подключен в сеть электропитания. Во время провидения анализа была обеспечена стабильная загруженность системы.

4.2 Описание исследования

Индекс базы данных – это специальная структура, позволяющая сократить время выполнения запроса на получение данных. Достигается это за счет создания, хранения и обработки дополнительных структур данных. В PostgreSQL существует несколько типов индексов, в частности B-Tree и Hash индекс. Кроме того, есть возможность создать частичный индекс [16].

Одна из возможностей разработанного приложения – создание брони. Для поддержания такой функциональности необходима функция, которая получает список всех активных броней и меняет статус на «неактивна» у тех броней, срок действия которых истекла.

Очевидно, что со временем количество активных броней будет намного меньше неактивных. В связи с этим возникает необходимость использования индекса для уменьшения времени выполнения запроса.

В ходе исследования происходил замер времени выполнения запроса при отсутствии индекса, наличии B-Tree индекса, Hash индекса и частичного B-Tree индекса. Также исследовался необходимый для хранения индекса объем памяти.

Исследование проводилось на таблице, количество записей в которой

последовательно увеличивалось с 1000 до 50000.

4.3 Результаты исследования

В таблице 4.1 приведены результаты замеров времени выполнения запросов. На рисунке 4.1 представлены графики зависимости времени выполнения запроса от типа индекса и количества записей в таблице.

Таблица 4.1 – Результаты замеров времени выполнения запроса в микросекундах

Количество записей в таблице	Без индекса	B-Tree индекс	Частичный B-Tree индекс	Hash индекс
1000	0.57	0.50	0.50	0.52
5000	0.99	0.86	0.75	0.85
10000	1.54	1.17	1.13	1.27
25000	3.45	2.48	2.31	2.63
50000	6.75	4.58	4.42	4.75

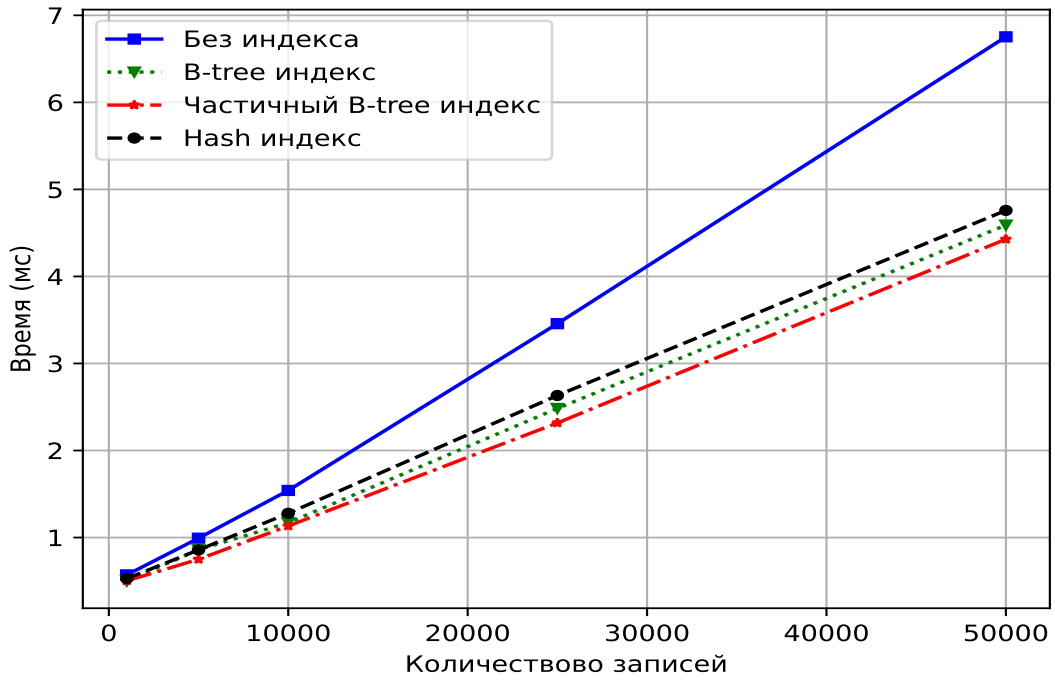

Рисунок 4.1 — Графики зависимости времени выполнения запроса от типа индекса и количества записей в таблице.

В таблице 4.2 приведены результаты замеров требуемой для хранения индексов памяти. На рисунке 4.2 представлены графики зависимости требуемой для хранения индексов памяти от типа индекса и количества записей в таблице.

Таблица 4.2 – Результаты замеров памяти в килобайтах

Количество записей в таблице	B-Tree индекс	Частичный B-Tree индекс	Hash индекс
1000	16	16	64
5000	56	16	440
10000	88	16	720
25000	176	32	1560
50000	336	32	3040

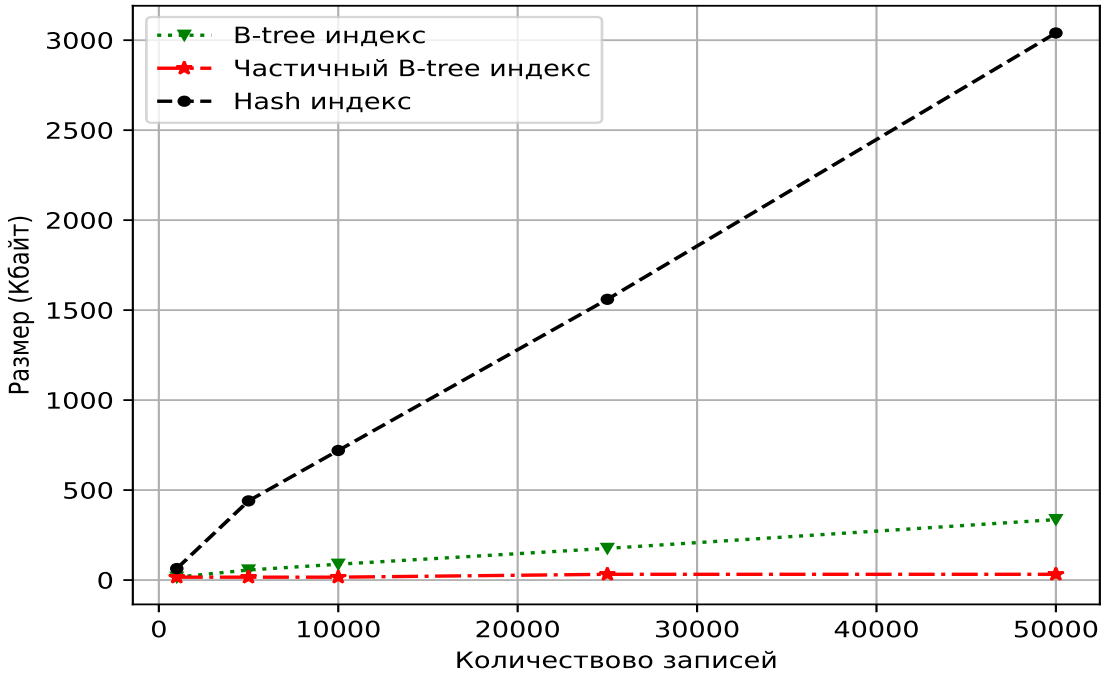

Рисунок 4.2 — Графики зависимости времени выполнения запроса от типа индекса и количества записей в таблице.

Вывод

Результаты исследования показали, что в ситуации, когда поиск записей осуществляется по полям, значения которых встречаются в таблице редко,

лучше всего использовать частичный индекс.

Hash-индекс требует для хранения в 10 раз больше памяти, чем B-Tree индекс и в 100 раз больше, чем Частичный B-Tree индекс.Это, объясняется большим количеством коллизий, вызванным наличием всего двух возможных значений в поле типа boolean.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы поставленная цель – разработка базы данных и приложения для сети автопарковок – была выполнена. Также, были решены все задачи:

1) проведен анализ предметной области;
2) формализованы требования к создаваемой системе;
3) спроектирована база данных и приложение для доступа к ней;
4) реализовано спроектированное программное обеспечение;
5) проведено исследование зависимости времени выполнения запроса и занимаемой индексом памяти от типа индекса и объема записей базы данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Вавринчук П. А.*, *Рябкова Е. Б.* Паркинг – основное решение дефицита парковочных мест // Новые идеи нового века. — 2014.

2. *Анисимова Н. А.*, *Потлова Л. А.* Обоснование инновационных преимуществ автоматических парковок для автомобилей // Научный вестник воронежского государственного архитектурно-строительного университета. — 2017.

3. Came Vector [Электронный ресурс]. — Режим доступа: https://www.vector-ap.ru/ (дата обращения: 23.04.2024).

4. Квазар [Электронный ресурс]. — Режим доступа: https://kvazar.ru/ (дата обращения: 23.04.2024).

5. Московский паркинг [Электронный ресурс]. — Режим доступа: https://parking.mos.ru/ (дата обращения: 23.04.2024).

6. *Дейт К. Д.* Введение в системы баз данных. — «Вильямс», 2006.

7. *Кузнецов С. Д.* Основы современных баз данных. — Центр Информационных Технологий, 1998.

8. *Парфенов Ю. П.* Постреляционные хранилища данных: учебное пособие. — Центр Информационных Технологий, 2016.

9. *Мартин Р.* Чистая архитектура. — «Питер», 2018.

10. PostgreSQL [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/ (дата обращения: 22.05.2024).

11. MySQL [Электронный ресурс]. — Режим доступа: https://www.mysql.com/ (дата обращения: 22.05.2024).

12. Oracle [Электронный ресурс]. — Режим доступа: https://www.oracle.com/cis/database/ (дата обращения: 22.05.2024).

13. Java [Электронный ресурс]. — Режим доступа: https://www.java.com/ru/ (дата обращения: 22.05.2024).

14. PL/pgSQL [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/docs/current/plpgsql.html (дата обращения: 22.05.2024).

15. Hibernate [Электронный ресурс]. — Режим доступа: https://hibernate.org/ (дата обращения: 22.05.2024).

16. Индексы PostgreSQL [Электронный ресурс]. — Режим доступа: https://www.postgresql.org/docs/current/indexes.html (дата обращения: 29.05.2024).