МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Введение в нереляционные базы данных»

Тема: Разработка сайта по заказу технологического транспорта

Студентка гр. 2381 _____ Романова К.А.
Студент гр. 2384 _____ Володченко М

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2025

ЗАДАНИЕ

Тема проекта: Разработка сайта по заказу технологического транспорта

Исходные данные:

Необходимо реализовать веб-приложение для бронирования технологического транспорта для СУБД (MongoDB) с возможностью переключения между ними.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Качественные требования к решению»

«Сценарий использования»

«Модель данных»

АННОТАЦИЯ

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания приложения для бронирования технологического транспорта для СУБД (MongoDB), т.к это отличная возможность научится работать на практике с нереляционной СУБД.

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

Цель работы — создать высокопроизводительное и удобное решение для хранения системы бронирования технологического транспорта.

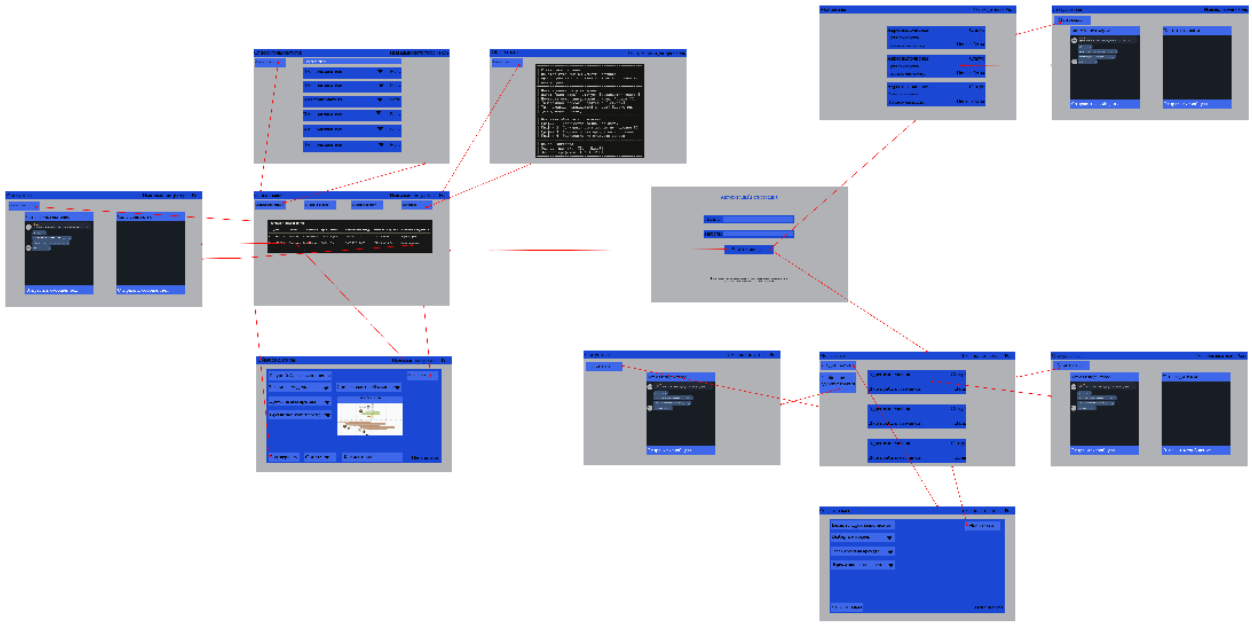
Было принято решение разработать веб-приложение, позволяющее хранить систему бронирования технологического транспорта, при этом позволяющую удобно с ней взаимодействовать.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать веб-приложение с использованием СУБД (MongoDB) со следующими основными функциями: создание заявки (веб-интерфейс заказчика), исполнение заявки (веб-интерфейс водителя), мониторинг и отчетность (веб-интерфейс диспетчера).

3. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Макеты UI



1. *Роль Заказчик*

Цель: оформление заказа на бронирование спецтехники

1.1. Создание заказа

1. Заказчик переходит на сайт и авторизуется/регестрируется
2. Переходит в раздел «Создать заказ»
3. Вводит:
    - Адрес
    - Дату и время начала/конца аренды
    - Выбирает технику из доступного списка
4. Видит цену услуги
5. Подтверждает заказ

Результат: заказ переходит в очередь на подтверждение диспетчером

1.2. Просмотр и отмена заказов

- У заказчика есть возможность открыть «Мои заказы» увидеть статусы:
    - «Ожидает подтверждения»

- «Подтвержден»
- «Выполняется»
- «Завершен»

- Если заказ еще не подтвержден диспетчером, заказчик может его отменить.

2. *Роль Администратор*

Цель: TODO

2.1. Массовый импорт базы водителей

2.2. Массовый экспорт базы водителей

- Выставление необходимых фильтров для получения статистики водителей

2.3. Массовый импорт базы клиентов

2.4. Массовый экспорт базы клиентов

- Выставление необходимых фильтров для получения статистики клиентов

2.5. TODO

3. *Роль Диспетчер*

Цель: Управление заказами и статусами техники.

3.1. Открывает «Список заявок».

- Видит детали каждого заказа.
- Проверяет доступность техники и водителей.
- Подтверждает или отклоняет заказ.
- В случае отклонения указывает причину.

Результат: Если подтвержден, заказ назначается водителю и переходит в статус «Выполняется».

3.2. Управление таймплайнами

- Открывает «Статусы техники и водителей».
- Просматривает занятость машин и водителей.
- Может вручную освободить или заблокировать ресурсы.

Результат: Обновленные статусы позволяют корректно назначать заказы.

4. *Роль Водитель*

Цель: Выполнение заказов.

4.1. Просмотр заказов

- Открывает «Мои заказы».
- Видит заказы с адресами, временем начала/окончания работы.
- Может принять к исполнению (если не назначено оператором).

4.2. Отказ от заказа

- Если водитель не может выполнить заказ, он может отказаться (при наличии причины).
- Оператор получает уведомление и переназначает заказ.

4. МОДЕЛЬ ДАННЫХ

Нереляционные модели данных

1. MongoDB

В данной модель есть четыре коллекции:

1. Коллекция - Заявка (Application)

- Хранение информации о заявках на бронирование ресурсов (ТС). Содержит все необходимые данные о заказе, включая заказчика, адрес назначения, вид ТС, дату выполнения, статус и связанный объект (водитель).

application_id: Number - 4 байта
date_of_creation: Date (ISO) - 8 байт
customer: [
{
customer_id: Number - 4 байта
customer_full_name: String - 50 байт
contact_information: String - 12 байт
customer_login: String - 20 байт
customer_password: String - 32 байта
}]
destination_address: String - до 100 байт
type_tc_id: Number - 4 байта
total_cost: Number - 8 байт
date_of_accomplishment: Date (ISO) - 8 байт
administrator_id: Number - 4 байта
working_shift_id: Number - 4 байта
application_status: String - до 50 байт

2. Коллекция - Administrator

administrator_id: Number - 4 байта
login: String - 20 байт
password: String - 32 байта

3. Коллекция - Рабочая смена (Working_shift)

- Хранение информации о водителях, включая контактные данные, статус и список доступных видов ТС. Связан с заявками и рабочими сменами.

working_shift_id: Number - 4 байта
driver: [
{
driver_id: Number - 4 байта
driver_full_name: String - до 150 байт
contact_information: String - 12 байт
status: String - 30 байт
}]
working_shift_date: Date (ISO) - 8 байт
status (opened, closed): String - 30 байт

4. Коллекция — Танспортное средство (Тс)

- Хранение информации об автомобилях, включая гос. номер, вид ТС, состояние, стоянку и показатели (данные из системы мониторинга транспорта). Связан с заявками и стоянками (вложенный документ).

tc_registration_number: String - до 12 байт
working_shift_id: Number - 4 байта
type_tc: [
{
type_tc_id: String - 4 байта
name_type_tc: String - 150 байта
type_tc_cost: Number - 30 байта
}]
condition: String - до 40 байт
parking_lot: [
{
parking_lot_id: Number - 4 байта
address: String - до 100 байт
coordinates: String - до 60 байт
}]

Оценка удельного объема информации хранимой в модели

Количество байт, равное m, необходимое для хранения 1 заявки БД:

m = Application + Administrator + Working_shift + TC = 308 + 56 + 256 + 404 = 1014

Тогда дял хранения N заявок необходимо:

V = 958 * N байт

Коллекция - Application

4 (application_id) + 8 (date_of_creation) + 4 (customer_id) + 50 (customer_full_name) + 12 (contact_information) + 20(customer_login) + 32 (customer_password) + 100 (destination_address) + 4 (type_tc_id) +8 (total_cost) + 8 (date_of_accomplishment) + 4 (administrator_id)+ 4 (working_shift_id) + 50 (application_status) = 308 байт

Коллекция - Administrator

4 (administrator_id) + 20 (login) + 32 (password) = 56 байт

Коллекция - Working_shift

4 (working_shift_id) + 4 (driver_id) + 150 (driver_full_name) + 12 (contact_information) + 30 (driver_status) + 8 (working_shift_date) + 30 (status (open,closed)) = 246 байт

Коллекция - TC

12 (tc_registration_number) + 4 (working_shift_id) + 4 (type_tc_id) + 150 (name_type_tc) + 30 (type_tc_cost) + 40 (condition) + 4 (parking_lot_id) + 100 (address) + 60 (coordinates) = 404 байта

Примеры запросов

Регистрация администратора

```
db.administrator.insertOne({
  login: "ivanov234",
  password: "8h1dac4s8fa241e83k9"
});
```

Запрос на добавление нового заказа

```
db.application.insertOne({
  application_id: "8",
  date_of_creation: new Date(),
  Customer: {
    customer_id: "6",
    customer_full_name: "Сергеев Петр Иванович",
    contact_information: "8 921 455-32-23"
    customer_login: "luk231",
    customer_password: "kl321455f"
  },
  destination_address: "г. Санкт-Петербург, Ленинский пр-т, д. 3",
  type_tc_id: "грузовик",
  total_cost: 12000,
  date_of_accomplishment: "31.04.2024",
  administrator_id: "7",
  working_shift_id: "4",
  application_status: "Подтверждено администратором",
})
```

Сортировка заказов по срочности исполнения

```
db.application.find().sort({ дата: 1 })
```

Импорт бызы данных водителей, клиентов

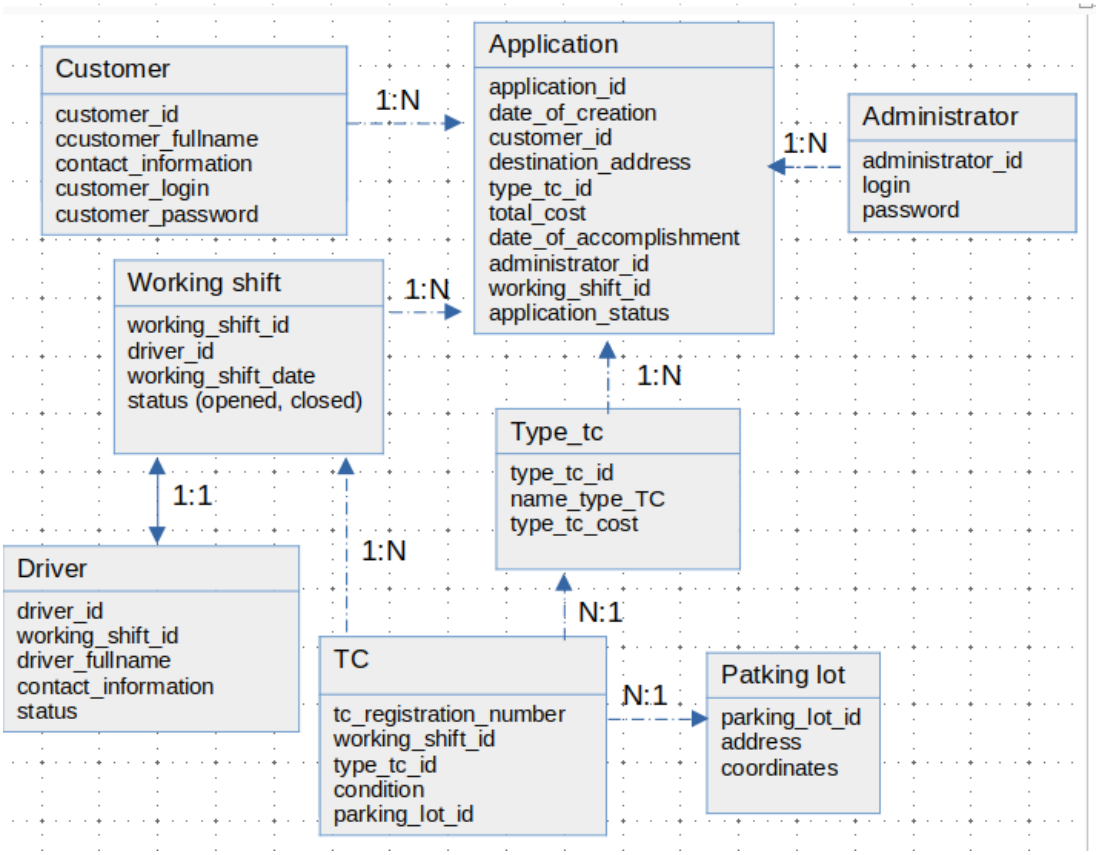
```
// Пример для базы водителей
db.Working_shift.insertMany([
  { driver_id: 1, driver_fullname: "Иванов Иван Иванович", ... },
  { driver_id: 2, driver_fullname: "Петров Петр Петрович", ... }
]);
```

Экспорт базы данных водителей, клиентов

```
db.Working_shift.find().toArray();
db.Application.aggregate([
{
 $group: {
  _id: "$Customer.customer_id",
  customer_full_name: { $first: "$Customer.customer_full_name" },
  contact_information: { $first: "$Customer.contact_information" },
  application_history: {
  $push: {
   application_id: "$application_id",
   date_of_creation: "$date_of_creation",
   destination_address: "$destination_address",
   type_tc_id: "$type_tc_id",
   total_cost: "$total_cost",
   date_of_accomplishment: "$date_of_accomplishment",
   application_status: "$application_status"
   }
  }
 }
},
{
 $project: {
  _id: 0,
  customer_id: "$_id",
  customer_full_name: 1,
  contact_information: 1,
 }
}
]).toArray()
```

Аналог модели данных для SQL СУБД

Вышеописанную модель данных также можно представить в виде реляционной модели с помощью таблиц:



Оценка удельного объема информации, хранимой в модели

Количество байт, равное m, необходимое для хранения 1 заявки БД:

m = Application + Administrator + Working_shift + TC = 308 + 56 + 256 + 404 = 1014

Тогда дял хранения N заявок необходимо:

V = 958 * N байт

Примеры запросов

Авторизация администратора

- Кол-во запросов - 1
- Кол-во зайдействованных колкций - 1

```
SELECT id FROM administrator
WHERE login = 'administrator' AND password = '7y4ucj37yaa455d61d8';
```

Запрос на добавление нового заказа

- Кол-во запросов - 1
- Кол-во зайдействованных колкций - 1

```
INSERT INTO Application (id, ,date_of_creation , customer_id,destination_address,type_tc_id, total_cost,
date_of_accomplishment,

administrator_id, working_shift_id , application_status)
VALUES (2, 20.12.2034, 3, 'г. Санкт-Петербург, Белы Куна ул., д. 3, кв. 78' , 'Грузовик', 12000, 21.12.2024, 1, 3,
'Подтверждена администратором');
```

Редактирование заказа

- Кол-во запросов - 1
- Кол-во зайдействованных колкций - 1

```
UPDATE Application
SET application_status = 'Отменена заказчиком', updated_at = datetime('now')
WHERE id = 1;
```

Сравнение моделей

1. Удельный обьем информации

У NoSQL обьем информации занимает меньше памяти, чем у SQL в рмаках 1 заявки на 22 байта. Это происходит, из-за отсутствия свзей между некоторыми сущностями, благодаря их включению в другие (влож. документы). Модель NoSQL выигрывает по компактности хранения при небольших потерях нормализации.

2. Запросы по отдельным use case

В SQL-модели для выполнения типовых юзкейсов (например, получения информации о фильме с актёрами, режиссёрами и жанрами) требуется больше отдельных запросов, так как данные распределены по разным таблицам и связаны между собой внешними ключами. В среднем на один юзкейс уходит 2–4 запроса с объединениями (JOIN).

В NoSQL-модели всё содержимое заявки достаточно одного запроса и одной коллекции.

Вывод

Исходя из всех сравнений, проведённых выше, можно сделать вывод, что нереляционные СУБД в данной задаче имеют преимущество в количестве занимаемой памяти, количестве запросов и сложности запросов.