

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»



Институт
интеллектуальных кибернетических систем

Кафедра кибернетики (№ 22)

Направление подготовки 09.03.04 Программная инженерия

Курсовая работа по предмету

“Основы автоматизированных информационных технологий”

Тема:

Сервис бронирования ресторанов AllReady

Преподаватель: Тихомирова Дарья Валерьевна

Студент: Кушнеревич Антон Викторович

Группа: Б22-564

Москва 2024

Содержание	3
1. Описание предметной области	3
1.1. Формулировка задания	3
1.2. Конкретизация предметной области	3
1.3. Пользователи системы	3
1.4. Сроки хранения информации	3
1.5. События, изменяющие состояние базы данных	3
1.6. Основные запросы к базе данных (на естественном языке)	3
2. Концептуально-информационная модель предметной области	3
2.1. Ег-диаграмма модели	3
2.2. Оценка мощностных характеристик сущностей и связей	3
3. Концептуальное проектирование	3
3.1. Принятые проектные соглашения	3
3.2. Обоснование выбора модели базы данных	3
3.3. Используемые в системе кодификаторы	3
3.4. Концептуальная модель базы данных	3
4. 4. Логическое проектирование	3
4.1. Ег-диаграмма базы данных (logical)	3
4.2. Схемы отношений базы данных (physical)	3
4.3. Схема реляционной базы данных	3
4.4. Схемы основных запросов на реляционной алгебре	3
5. Физическое проектирование	3
5.1. Обоснование выбора конкретной СУБД	3
5.2. Создание базы данных	3
5.3. Создание таблиц	3
5.4. Заполнение таблиц. ETL-процессы загрузки базы данных	3
5.5. Запросы в терминах SQL	3
5.6. Оценка размеров базы данных и каждого из файлов	3
6. Приложение. Отчеты	3

Содержание

1. Описание предметной области

1.1. Формулировка задания

Спроектировать базу данных для сервиса бронирования ресторанов – AllReady. База данных должна содержать информацию о клиентах и ресторанах, также подробно отражать всю информацию про заказ клиента, столик и его отзыв о заведении.

1.2. Конкретизация предметной области

AllReady – это система бронирования ресторанов, которая позволяет клиентам находить подходящие заведения, резервировать столики, оформлять заказы и оставлять отзывы.

Основной функционал:

- **Управление ресторанами:**
 - Хранение данных о ресторанах (название, адрес, кухня, контакты, график работы).
 - Информация о доступных столиках (вместимость, расположение, статус брони).
- **Бронирование и заказы:**
 - Клиенты могут бронировать столики на определённое время и дату.
 - Фиксация деталей брони (количество гостей, особые пожелания, статус).
 - Возможность оформления предзаказа блюд (если ресторан поддерживает).
- **Управление клиентами:**
 - Регистрация и аутентификация пользователей.
 - Система лояльности (скидки, бонусы).
 - История бронирований и заказов.
- **Отзывы и рейтинги:**
 - Клиенты могут оставлять отзывы и оценки ресторанам.
 - Рестораны видят обратную связь и могут отвечать на отзывы.
- **Администрирование:**
 - Управление ресторанами (добавление, редактирование, блокировка).
 - Настройка тарифов для партнёров (комиссия за бронирование).

1.3. Пользователи системы

Можно выделить два класса пользователей системы:

- **Клиенты.** Основные пользователи сервиса, которые могут просматривать рестораны, фильтровать их по кухне, расположению и рейтингу. Имеют возможность бронировать столики онлайн на удобное время, выбирая количество гостей и особые пожелания. Могут оформлять предварительные заказы блюд из меню ресторана. После посещения оставляют отзывы и оценки, которые помогают другим пользователям в выборе. Имеют личный кабинет с историей всех бронирований и накопленными бонусами.
- **Владельцы ресторанов / менеджеры.** Управляют профилем своего заведения, обновляя информацию о меню, графике работы и доступных столиках. Видят все текущие и будущие бронирования с деталями о количестве гостей и особых пожеланиях. Могут отвечать на отзывы клиентов, благодаря чему улучшают репутацию заведения. Получают аналитику по популярным блюдам и времени наибольшей посещаемости. Имеют доступ к финансовым отчетам по бронированиям и предзаказам.

1.4. Сроки хранения информации

Сервис AllReady обеспечивает надежное хранение ключевых данных для бесперебойной работы системы. Основная информация, включающая профили пользователей, данные ресторанов, историю бронирований и финансовых операций, сохраняется бессрочно в соответствии с требованиями законодательства и потребностями сервиса. Сроки хранения персональных данных регулируются Федеральным законом № 152-ФЗ «О персональных данных» (ПДн) и внутренними политиками сервиса. Персональные данные пользователей хранятся до достижения целей обработки или до истечения срока, установленного договором или законодательством РФ. После удаления аккаунта или прекращения его использования персональные данные аннулируются (обезличиваются) в течение 6 месяцев, если иное не предусмотрено законом. В случае отсутствия активности более 3 лет аккаунты переводятся в спящий режим с частичным обезличиванием данных. Операционные данные и системные логи хранятся 12 месяцев с возможностью архивирования. Финансовая отчетность и данные, связанные с налоговым учетом, сохраняются не менее 5 лет в соответствии с Налоговым кодексом РФ. Отзывы и рейтинги ресторанов остаются в системе на постоянной основе, если пользователь не запросил их удаление.

1.5. События, изменяющие состояние базы данных

Основные события, изменяющие данные в системе бронирования:

- Создание нового бронирования с указанием параметров (дата и время, количество гостей, выбранный столик, особые пожелания)
- Изменение статуса бронирования в процессе его выполнения (подтверждено, отменено, завершено, клиент не явился)
- Регистрация или обновление данных клиентов, владельцев ресторанов и администраторов системы
- Изменение информации о ресторанах (график работы, меню, доступные столики, тарифы на комиссию сервиса)
- Создание и обработка заказов блюд (добавление/изменение позиций, статус приготовления, оплата)
- Обновление отзывов и рейтингов (добавление новых оценок, ответы ресторанов, модерация контента)
- Изменение системных настроек (условия бронирования, политика отмены, правила лояльности)

1.6. Основные запросы к базе данных (на естественном языке)

Запросы по управлению бронированиями:

- Получение всех бронирований, оформленных с применением конкретного промокода или скидки
- Поиск бронирований в статусе "подтверждено, но не оплачено" или "требуется подтверждения"
- Просмотр бронирований с примененными скидками по определенной акции за выбранный период
- Получение истории бронирований конкретного клиента за последний месяц
- Выборка доступных столиков в ресторане на указанную дату и время

Аналитические запросы для администраторов:

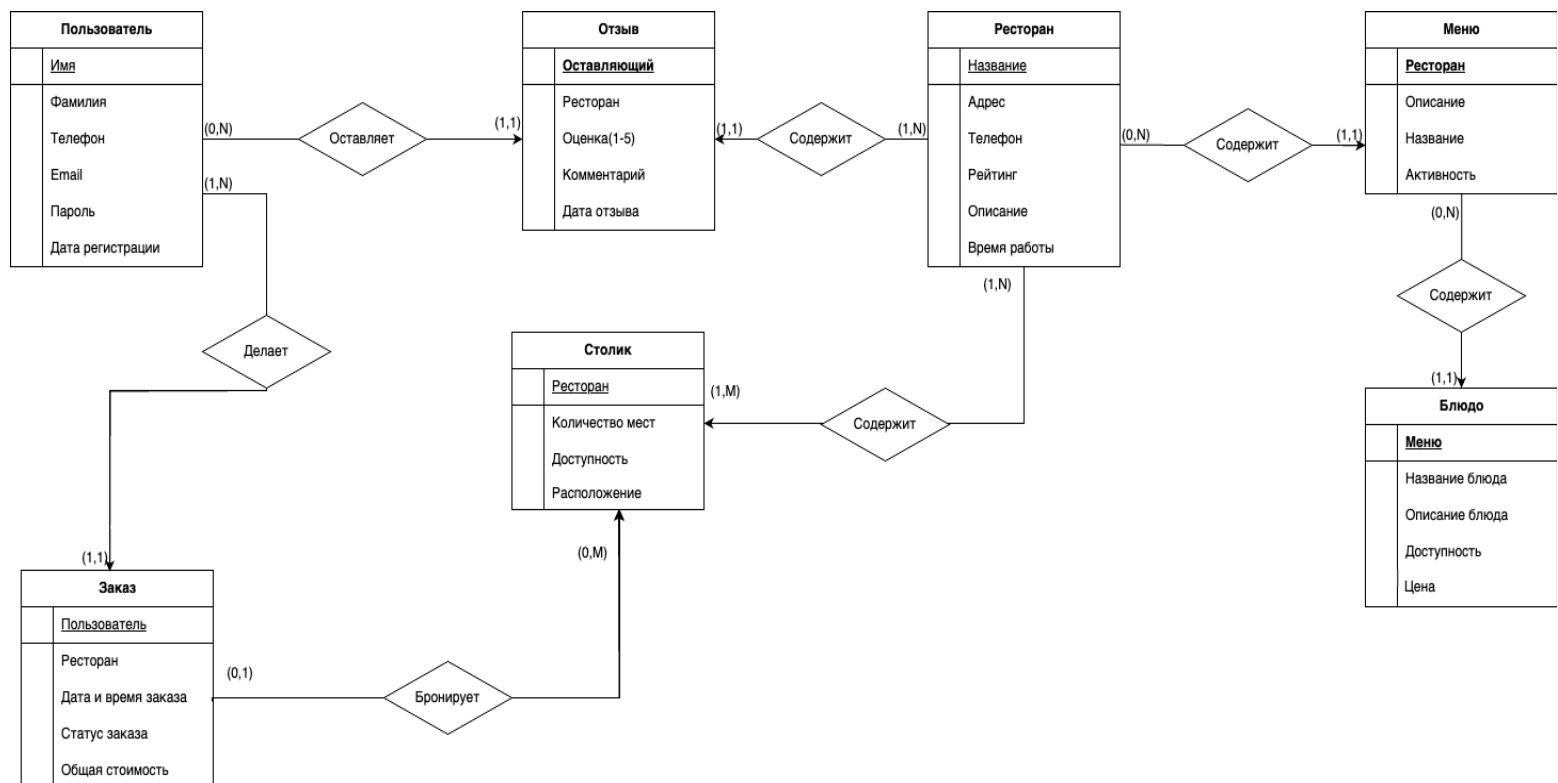
- Поиск клиентов, совершивших более 3 бронирований за последние 2 месяца
- Получение статистики по ресторанам с наибольшим количеством отмен бронирований
- Анализ сообщений поддержки по конкретному ресторану за определенный период
- Расчет влияния изменения комиссий сервиса на доход по различным категориям ресторанов
- Формирование отчета о доходе системы по типам кухни (итальянская, азиатская и др.) за квартал

Запросы для владельцев ресторанов:

- Получение списка предстоящих бронирований с указанием предзаказов
- Анализ популярности позиций меню по количеству предзаказов
- Просмотр отзывов с низкими оценками за последний месяц
- Расчет заполняемости зала по дням недели и времени
- Формирование отчета по среднему чеку в сравнении с предыдущим периодом

2. Концептуально-информационная модель предметной области

2.1. Ег-диаграмма модели



2.2. Оценка мощностных характеристик сущностей и связей

Сущности

Сущность	Минимальная	Средняя	Максимальная
Клиент	0	50	10000
Ресторан	0	100	10000
Столик	0	500	50000
Отзыв	0	3000	500000
Меню	0	350	50000
Блюдо	0	1000	100000
Заказ	0	2000	100000

Связи между сущностями

Связь (сущность – сущность)	Минимальная	Средняя	Максимальная
Ресторан - Столик	0	5000	100000
Пользователь - Отзыв	0	3000	50000
Ресторан - Отзыв	0	3000	50000
Ресторан - Меню	0	300	10000
Меню - Блюдо	0	7000	100000
Пользователь - Заказ	0	2000	100000
Заказ - Столик	0	2000	100000

Пояснения:

- **Пользователи:** Ожидается до 10,000 активных клиентов
- **Рестораны:** До 10,000 заведений в системе
- **Столики:** Крупные рестораны могут иметь до 100 столиков
- **Отзывы:** Популярные рестораны могут получать до 5,000 отзывов
- **Заказы:** Пиковая нагрузка - до 100,000 активных заказов
- **Меню:** Каждый ресторан может иметь несколько меню (основное, барное и т.д.)
- **Блюда:** В каждом меню может быть до 200 позиций

Эти оценки могут быть уточнены в зависимости от специфики развёртывания системы и прогнозируемых объёмов данных.

3. Концептуальное проектирование

3.1. Принятые проектные соглашения

Для проектирования базы данных сервиса AllReady были приняты следующие соглашения:

- Именованье сущностей и атрибутов: Используется snake_case для таблиц и полей (например, restaurant, user_review).
- Первичные ключи: Для всех сущностей первичный ключ — автоинкрементное целое число (id).
- Связи между сущностями: Внешние ключи именуются как название_сущности_id (например, restaurant_id в таблице restaurant).

3.2. Обоснование выбора модели базы данных

Выбрана реляционная модель базы данных по следующим причинам:

Структурированность данных: Данные сервиса (рестораны, бронирования, отзывы) имеют четкую структуру и взаимосвязи. Реляционная модель позволяет эффективно организовать такие данные в виде таблиц с определенными атрибутами и установить связи между ними. Это особенно важно для системы бронирования, где необходимо хранить информацию о ресторанах, пользователях, бронированиях и отзывах в структурированном виде. **Целостность данных:** Реляционная модель обеспечивает соблюдение ограничений (например, внешние ключи для связи бронирования с клиентом и рестораном). Механизмы обеспечения целостности данных помогают предотвращать ошибки и несоответствия в базе данных, что критично для бизнес-процессов, связанных с бронированием столиков и управлением клиентскими данными. **Надежность:** Транзакционность и ACID-свойства важны для операций бронирования и оплаты. Реляционные СУБД гарантируют, что все операции либо будут выполнены полностью, либо не будут выполнены вообще, что критично для финансовых транзакций и процессов бронирования, где необходимо обеспечить согласованность данных даже при возникновении сбоев.

3.3. Используемые в системе кодификаторы

Для стандартизации данных и обеспечения единообразия информационного обмена в системе применяются следующие кодификаторы:

Статусы бронирования:

ENUM('pending', 'confirmed', 'cancelled', 'completed')

- **pending** — ожидает подтверждения
- **confirmed** — подтверждено рестораном
- **cancelled** — отменено
- **completed** — завершено (клиент посетил ресторан)

Локации столиков:

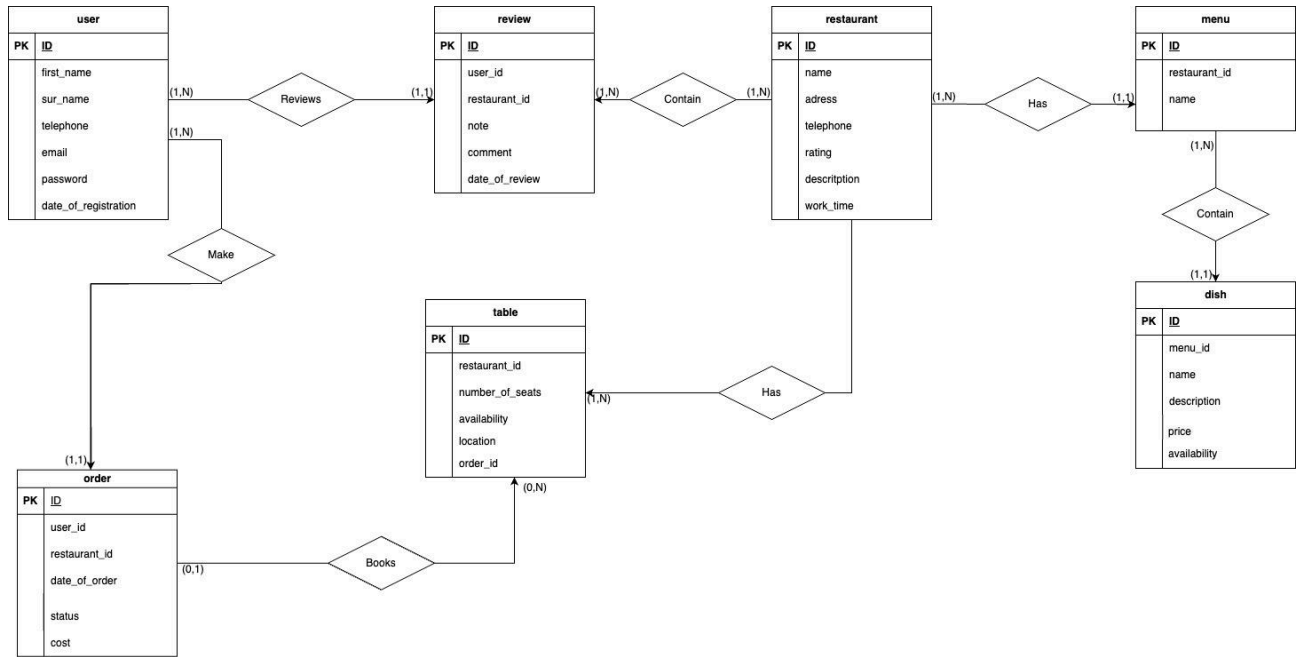
Код	Зона ресторана
terrace	Терраса
hall	Основной зал
bar	У бара
vip	VIP зона
window	У окна

Представлены целочисленными значениями в диапазоне от 1 до 5:

- **1** — крайне неудовлетворительно
- **2** — неудовлетворительно
- **3** — удовлетворительно
- **4** — хорошо
- **5** — отлично

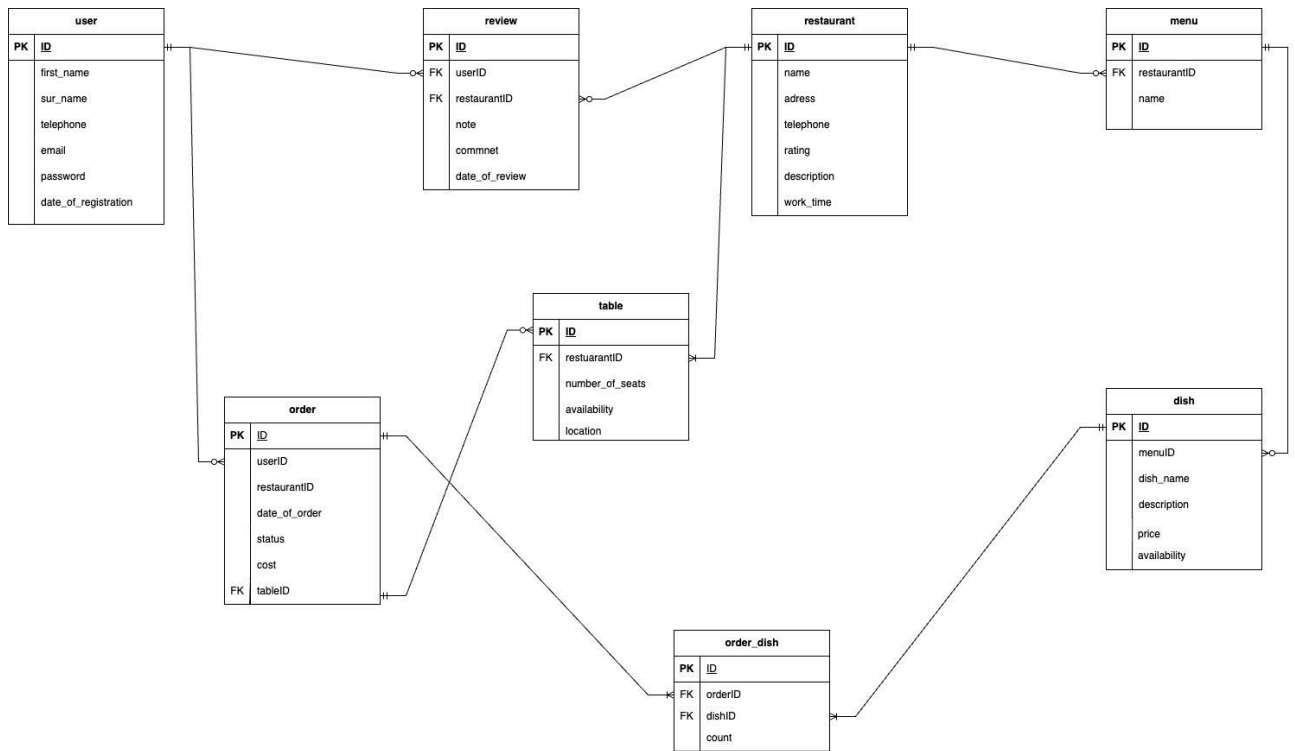
Примечание: Кодификаторы регулярно обновляются и могут быть расширены в соответствии с развитием системы и потребностями бизнеса.

3.4. Концептуальная модель базы данных

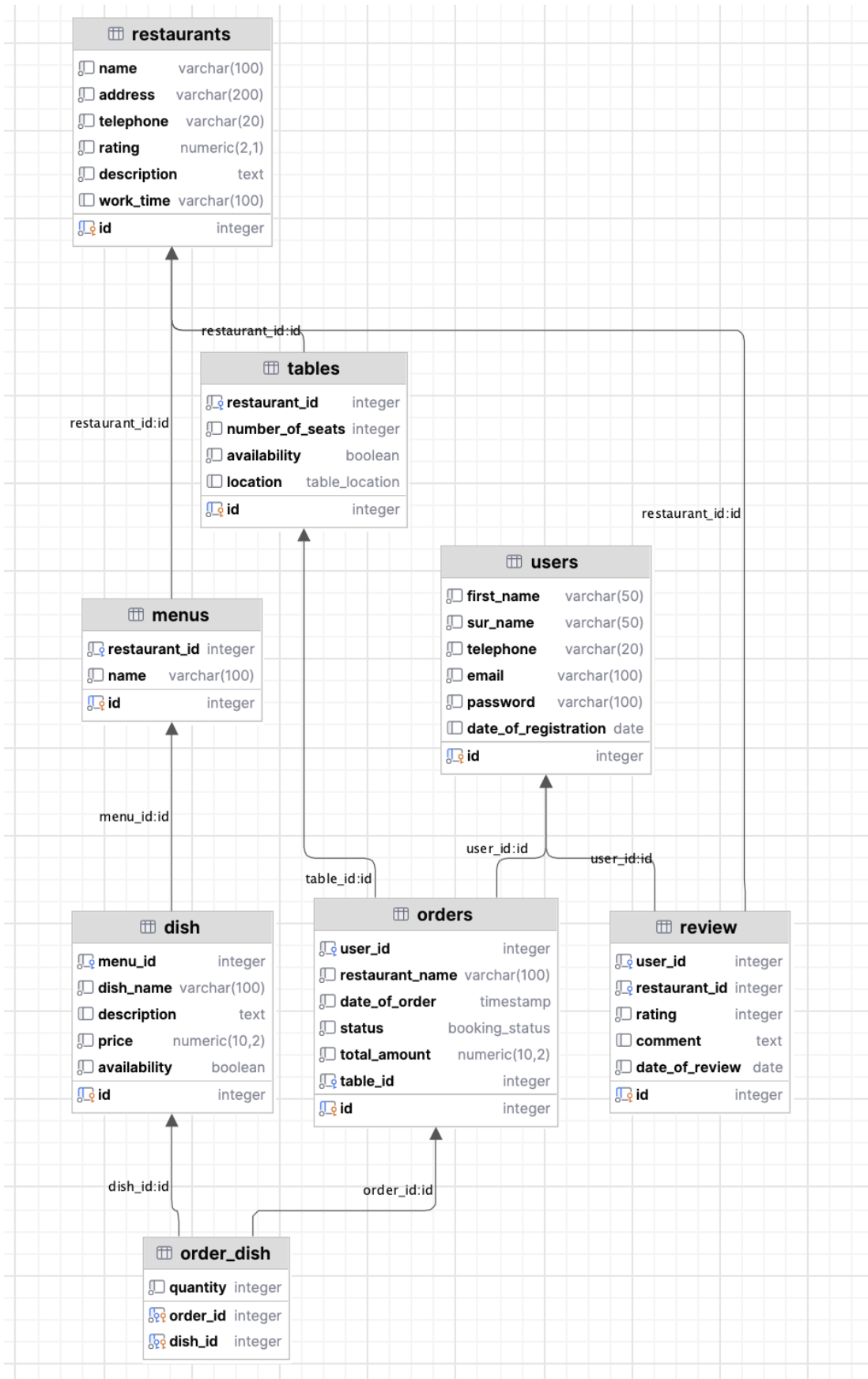


4. 4. Логическое проектирование

4.1. Er-диаграмма базы данных



4.2. Схемы отношений базы данных



4.3. Схема реляционной базы данных

R1	users(#id, first_name,sur_name,telephone,email,password,date_of_registration)
R2	restaurants(#id, name, address, telephone, rating, description, work_time)
R3	tables(#id, restaurant_id, number_of_seats, availability, location)
R4	menus(#id, restaurant_id , name)
R5	dish(#id, menu_id, dish_name, description, price, availability)
R6	orders(#id, user_id, restaurant_name, order_date_time, order_status, total_amount, table_id)
R7	order_dish(#id, order_id ,dish_id, quantity)
R8	review(#id, user_id , restaurant_id, rating, comment, review_date)

4.4. Схемы основных запросов на реляционной алгебре

- Получение всех ресторанов с рейтингом выше 4.5:

Рестораны (name) = $\sigma_{\text{rating} > 4.5}(\text{restaurants})$ [name]

- Поиск всех доступных столиков в ресторане "La Piazza" на 4 персоны

Столики (id, location) =

$(\sigma_{\text{name} = \text{"La Piazza"}}(\text{restaurants}) \bowtie \text{restaurants.id} = \text{restauranttables.restaurantid})$

$\sigma_{\text{number_of_seats} \geq 4 \wedge \text{availability} = \text{true}}(\text{restauranttables})$ [id, location]

- Получение всех заказов пользователя с email "ivan@mail.ru"

Заказы (id, order_date_time, total_amount) =

$(\sigma_{\text{email} = \text{"ivan@mail.ru"}}(\text{users}) \bowtie \text{users.id} = \text{orders.userid})$ orders)

[id, order_date_time, total_amount]

- Вывод топ-5 самых популярных блюд (по количеству заказов)

Популярные_блюда (dish_name, count) =

```

    v_dishid;COUNT(*)→count(order_dish) ⋈ order_dish.dishid = dish.id dish
    [dish_name, count]
    SORT BY count DESC
    LIMIT 5

```

- Поиск отзывов, содержащих слово "прекрасный"

Отзывы (user_id, comment, rating) =

```

    σ_comment LIKE "%прекрасный%"(review) [user_id, comment, rating]

```

- Получение ресторанов с количеством отзывов больше 10

Популярные_рестораны (name, count_reviews) =

```

    (v_restaurantid;COUNT(*)→count_reviews(review) ⋈
    review.restaurantid = restaurants.id restaurants)
    [name, count_reviews]
    WHERE count_reviews > 10

```

5. Физическое проектирование

5.1. Обоснование выбора конкретной СУБД

Для реализации проекта в качестве системы управления базами данных (СУБД) была выбрана PostgreSQL. Этот выбор обусловлен следующими преимуществами:

- PostgreSQL — это надежная объектно-реляционная СУБД с открытым исходным кодом, поддерживающая расширенные возможности SQL и соответствующие современным стандартам.
- Высокая производительность и эффективный оптимизатор запросов делают PostgreSQL отличным решением для работы с большими объемами данных и сложными связями между таблицами.
- PostgreSQL широко распространена в индустрии и имеет активное сообщество, что обеспечивает доступность документации, инструментов и готовых решений для типовых задач.
- Важным преимуществом является поддержка пользовательских типов данных, таких как перечисления (ENUM), которые используются в проектируемой базе данных.
- PostgreSQL хорошо интегрируется с языком Go через драйверы (lib/pq, pgx) и инструменты миграций (например, Goose), что упрощает разработку и поддержку базы данных.

Таким образом, PostgreSQL представляет собой оптимальный выбор как с точки зрения функциональности, так и с учетом требований к надежности и масштабируемости проекта.

5.2. Создание базы данных

Развертывание базы данных осуществляется с использованием Docker и docker-compose, что обеспечивает простоту настройки, воспроизводимость окружения и удобство развертывания в разных средах.

Конфигурация PostgreSQL:

Для запуска PostgreSQL применяется официальный образ postgres, который настраивается через docker-compose.yml. Параметры подключения (пользователь, пароль, имя базы данных, порт) задаются через переменные окружения в файле .env, что повышает безопасность и гибкость конфигурации.

Управление миграциями

Структура базы данных формируется с помощью миграций, реализованных через Goose. Это позволяет:

- Контролировать изменения схемы БД в виде последовательных SQL-скриптов.
- Легко откатывать или применять обновления при разработке и развертывании.
- Автоматизировать процесс инициализации базы данных при запуске приложения.

Взаимодействие с базой данных:

Для выполнения SQL-запросов, отладки и администрирования используется консольное приложение на Go, которое подключается к PostgreSQL через драйвер (pgx). Это позволяет:

- Проверять корректность запросов прямо в коде.
- Автоматизировать тестирование и валидацию данных.
- Упростить интеграцию с основным приложением.

Преимущества выбранного подхода

- Гибкость: Контейнеризация позволяет быстро развернуть БД в любом окружении.
- Автоматизация: Миграции через Goose обеспечивают контроль версий схемы базы данных.
- Удобство разработки: Встроенные инструменты Go для работы с PostgreSQL ускоряют отладку и тестирование.

Таким образом, сочетание Docker, Goose и Go обеспечивает надежное и удобное управление базой данных на всех этапах разработки.

5.3. Создание таблиц

Все SQL-скрипты, отвечающие за формирование структуры базы данных, вынесены в отдельные файлы миграций. Они применяются в строгом порядке, обеспечивая последовательное создание и модификацию таблиц. Подробное описание структуры базы данных, включая схемы таблиц и их взаимосвязи, представлено в Приложении 1.

5.4. Заполнение таблиц. ETL-процессы загрузки базы данных

Для автоматизации загрузки данных был написан скрипт на языке Go. Процесс включал генерацию данных с помощью библиотеки `gofakeit` и predefined ручных сценариев, а также их последующую загрузку в PostgreSQL посредством библиотеки `pgx`.

Рассматриваются ключевые этапы и использованные подходы для обеспечения корректности и эффективности загрузки.

Код смотреть в Приложение 2

5.5. Запросы в терминах SQL

Запрос 1: Получить список всех ресторанов с рейтингом выше 4, отсортированных по названию:

```
SELECT name, rating, address
FROM restaurants
WHERE rating > 4
ORDER BY name;
```

Результат:

	name ▾	rating ▾	address ▾
1	(Leg)Cyte	5.0	13553 Port Neckfort, Lincoln, Rhode Island 47297
2	(Leg)Cyte	5.0	3352 East Loafburgh, Houston, California 85027
3	(Leg)Cyte	5.0	9835 Pineston, Jacksonville, North Carolina 84385
4	(Leg)Cyte	5.0	13930 Throughwaytown, North Las Vegas, Maryland 15609
5	(Leg)Cyte	5.0	4227 Milltown, Riverside, Wisconsin 47704
6	(Leg)Cyte	5.0	57160 Lodgemouth, Reno, Maryland 79442
7	(Leg)Cyte	5.0	149 New Villagetown, Plano, Iowa 70264
8	3 Round Stones, Inc.	5.0	758 Pointsborough, Omaha, South Dakota 89479
9	3 Round Stones, Inc.	5.0	72959 Turnpikestad, Newark, Maine 32587
10	3 Round Stones, Inc.	5.0	69397 New Squaresbury, San Francisco, Iowa 52650
11	3 Round Stones, Inc.	5.0	35093 Gardenberg, Baton Rouge, New Mexico 30492
12	3 Round Stones, Inc.	5.0	6121 New Forestmouth, Tampa, Virginia 88222
13	3 Round Stones, Inc.	5.0	21412 Inletberg, Oakland, North Carolina 27393
14	48 Factoring Inc.	5.0	35913 West Fieldshire, Oakland, Indiana 53020
15	48 Factoring Inc.	5.0	3642 Clubborough, San Francisco, New York 47733

Запрос 2: Найти все доступные столики с более чем 4 местами в определенном ресторане

Например, ресторан - Govini

```
SELECT rt.id, rt.number_of_seats, rt.location, r.address
FROM tables rt
JOIN restaurants r ON rt.restaurant_id = r.id
WHERE rt.availability = true
      AND rt.number_of_seats > 4
      AND r.name ILIKE '%Govini%'
ORDER BY r.address;
```

Результат:

	id	number_of_seats	location	address
1	28422	10	terrace	123 South Wayston, El Paso, Washington 72684
2	11117	8	bar	13368 Port Ranchmouth, Orlando, North Carolina 98299
3	23811	9	bar	17558 Streamview, Buffalo, Alaska 74681
4	20847	6	hall	17558 Streamview, Buffalo, Alaska 74681
5	15671	6	window	176 Lake Summitshire, St. Petersburg, Mississippi 88372
6	2571	8	window	380 Terraceburgh, Tucson, Tennessee 39310
7	13290	6	window	380 Terraceburgh, Tucson, Tennessee 39310
8	27565	5	terrace	431 Daleburgh, Mesa, Connecticut 34705
9	45233	10	terrace	50754 New Throughwayside, Miami, Utah 27015
10	7193	8	bar	50754 New Throughwayside, Miami, Utah 27015
11	32677	10	vip	656 West Knollsshire, Arlington, Missouri 47695
12	45911	7	bar	656 West Knollsshire, Arlington, Missouri 47695
13	35096	5	window	66832 North Stationborough, Corpus Christi, New Jersey 48300
14	49856	7	terrace	783 Springsfurt, Boston, Nebraska 77245
15	11718	8	vip	783 Springsfurt, Boston, Nebraska 77245

Запрос 3: Получить названия и цены всех блюд, которые доступны в данный момент, в порядке убывания цены

```
SELECT dish_name, price
FROM dish
WHERE availability = true
ORDER BY price DESC;
```

	 dish_name 	 price 
1	cooked mutton	100.00
2	mushroom stew	100.00
3	cooked salmon	100.00
4	golden apple	100.00
5	cooked chicken	100.00
6	pufferfish	100.00
7	sweet berry	100.00
8	raw porkchop	100.00
9	spider eye	100.00
10	enchanted golden apple	100.00
11	mushroom stew	100.00
12	bread	100.00
13	rotten flesh	100.00
14	beetroot soup	100.00
15	rabbit stew	100.00
16	raw mutton	100.00
17	spider eye	100.00
18	cooked cod	100.00
19	cake	100.00
20	steak	100.00

Запрос 4: Получить средний рейтинг для каждого ресторана

```
WITH restaurant_stats AS (  
  SELECT  
    r.id,  
    AVG(rv.rating) AS avg_rating,  
    ROUND(AVG(rv.rating) * 2) / 2 AS rating  
  FROM restaurants r  
  JOIN review rv ON r.id = rv.restaurant_id  
  GROUP BY  
    r.id  
)  
SELECT  
  rating,  
  COUNT(id) AS restaurants_count  
FROM restaurant_stats  
GROUP BY rating  
ORDER BY rating;
```










Результат:

	rating	restaurants_count
1	1	775
2	1.5	212
3	2	1046
4	2.5	547
5	3	1181
6	3.5	480
7	4	1045
8	4.5	188
9	5	815

Запрос 5: Найти пользователей, оставивших отзывы обо всех ресторанах, в которых они делали заказы

```
SELECT
    u.id,
    u.first_name,
    u.sur_name
FROM users u
JOIN orders o ON u.id = o.user_id
LEFT JOIN review rv ON u.id = rv.user_id
LEFT JOIN restaurants r ON rv.restaurant_id = r.id AND
o.restaurant_name = r.name
GROUP BY u.id, u.first_name, u.sur_name
HAVING COUNT(o.restaurant_name) = COUNT(r.name)
AND COUNT(o.id) > 0;
```

Результат:

	 id 		 first_name 		 sur_name 	
1	899		Federico		Braun	
2	872		Raoul		Wuckert	
3	7625		Noah		Bauch	
4	6351		Josiah		Wilderma	
5	2385		Elinore		Parisian	
6	7849		Edmond		Boehm	
7	7341		Lera		Pollich	
8	3970		Aiyana		Kozey	
9	7687		Gabriel		Walsh	
10	720		Chase		Schneider	
11	9975		Bradly		Huels	
12	478		Amely		Ziemann	
13	5639		Lavinia		Bode	
14	5913		King		Wyman	
15	7725		Jamir		Welch	

Запрос 6: Найти самое популярное блюдо для каждого ресторана

```
WITH dish_counts AS (
    SELECT
        r.id as restaurant_id,
        d.id as dish_id,
        d.dish_name,
        COUNT(*) as order_count,
        ROW_NUMBER() OVER (PARTITION BY r.id ORDER BY COUNT(*) DESC) as
rn
    FROM restaurants r
        JOIN menus m ON r.id = m.restaurant_id
        JOIN dish d ON m.id = d.menu_id
        JOIN order_dish od ON d.id = od.dish_id
        JOIN orders o ON od.orderid = o.id
    GROUP BY r.id, d.id, d.dish_name
)
SELECT
    r.name as restaurant_name,
    dc.dish_name,
    dc.order_count
FROM restaurants r
    JOIN dish_counts dc ON r.id = dc.restaurant_id
WHERE dc.rn = 1;
```

Результат

	restaurant_name	dish_name	order_count
1	SlashDB	steak	3
2	OnDeck	potato	3
3	Mango Transit	pufferfish	2
4	Cambridge Semantics	beetroot soup	2
5	Noveda Technologies	raw mutton	2
6	Personal, Inc.	raw salmon	2
7	IBM	raw chicken	2
8	Intermap Technologies	cooked cod	2
9	Navico	cookie	2
10	Ecodesk	honey bottle	2
11	American Red Ball Movers	raw rabbit	2
12	Merrill Lynch	glow berry	2
13	Buildingeye	honey bottle	2
14	Berkery Noyes MandASoft	beetroot	2
15	StreetCred Software, Inc	pufferfish	2

Запрос 7: Найти самый загруженный час для каждого дня недели, основываясь на объеме заказов

```
WITH hourly_orders AS (  
    SELECT  
        EXTRACT(DOW FROM date_of_order) AS day_of_week,  
        EXTRACT(HOUR FROM date_of_order) AS hour,  
        COUNT(*) AS order_count  
    FROM orders  
    GROUP BY  
        EXTRACT(DOW FROM date_of_order),  
        EXTRACT(HOUR FROM date_of_order)  
),  
ranked_hours AS (  
    SELECT  
        day_of_week,  
        hour,  
        order_count,  
        RANK() OVER (PARTITION BY day_of_week ORDER BY order_count  
DESC) AS hour_rank  
    FROM hourly_orders  
)  
SELECT  
    day_of_week,  
    LPAD(hour::text, 2, '0') || ':00' AS busiest_hour,  
    order_count  
FROM ranked_hours  
WHERE hour_rank = 1  
ORDER BY day_of_week;
```

Результат

	☐ day_of_week ▾	÷	☐ busiest_hour ▾	÷	☐ order_count ▾	÷
1		0	00:00			82
2		1	09:00			81
3		2	17:00			74
4		3	02:00			86
5		4	18:00			76
6		4	01:00			76
7		5	21:00			78
8		6	15:00			83

5.6. Оценка размеров базы данных и каждого из файлов

Отношение	Атрибут	Тип данных	Размер	Среднее кол-во	Объём, байт
users	id	serial	4	10000	3280000
	first_name	varchar(50)	50		
	sur_name	varchar(50)	50		
	telephone	varchar(20)	20		
	email	varchar(100)	100		
	password	varchar(100)	100		
	date_of_registration	date	4		
restaurants	id	serial	4	10000	5270000
	name	varchar(100)	100		
	address	varchar(200)	200		
	telephone	varchar(20)	20		
	rating	numeric(2,1)	3		
	description	text	100		
	work_time	varchar(100)	100		
tables	id	serial	4	50000	850000
	restaurant_id	integer	4		
	number_of_seats	integer	4		
	availability	boolean	1		
	location	table_location	4		
review	id	serial	4	10000	1200000

	user_id	integer	4		
	restaurant_id	integer	4		
	rating	integer	4		
	comment	text	100		
	date_of_review	date	4		
menus	id	serial	4	50000	5400000
	restaurant_id	integer	4		
	name	varchar(100)	100		
dish	id	serial	4	250000	54250000
	menu_id	integer	4		
	dish_name	varchar(100)	100		
	description	text	100		
	price	numeric(10,2)	8		
	availability	boolean	1		
orders	id	serial	4	10000	1320000
	user_id	integer	4		
	restaurant_name	varchar(100)	100		
	date_of_order	timestamp	8		
	status	booking_status	4		
	total_amount	numeric(10,2)	8		
	table_id	integer	4		
order_dish	id	serial	4	10000	160000
	order_id	integer	4		
	dish_id	integer	4		
	quantity	integer	4		

Средний размер базы данных - 71 730 000 КБ \approx 71,73 МБ

6. Приложение. Отчеты

Приложение 1

Листинг 1: Создание таблицы users

```
CREATE TABLE IF NOT EXISTS users(  
    id SERIAL PRIMARY KEY,  
    first_name VARCHAR(50) NOT NULL,  
    sur_name VARCHAR(50) NOT NULL,  
    telephone VARCHAR(20) NOT NULL,  
    email VARCHAR(100) NOT NULL,  
    password VARCHAR(100) NOT NULL,  
    date_of_registration DATE  
);
```

Листинг 2: Создание таблицы restaurants

```
CREATE TABLE IF NOT EXISTS restaurants (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(100) NOT NULL,  
    address VARCHAR(200) NOT NULL,  
    telephone VARCHAR(20) NOT NULL,  
    rating DECIMAL(2, 1) NOT NULL,  
    description TEXT NOT NULL,  
    work_time VARCHAR(100) NOT NULL  
);
```

Листинг 3: Создание таблицы tables

```
CREATE TABLE IF NOT EXISTS tables (  
    id SERIAL PRIMARY KEY,  
    restaurant_id INT NOT NULL,  
    number_of_seats INT NOT NULL,  
    availability BOOLEAN NOT NULL,  
    location table_location ,  
    FOREIGN KEY (restaurant_id) REFERENCES restaurants(id) ON DELETE  
    CASCADE  
);
```

Листинг 4: Создание таблицы review

```
CREATE TABLE IF NOT EXISTS review (  
  id SERIAL PRIMARY KEY,  
  user_id INT NOT NULL,  
  restaurant_id INT NOT NULL,  
  rating INT CHECK (rating BETWEEN 1 AND 5) NOT NULL,  
  comment TEXT NOT NULL,  
  date_of_review DATE NOT NULL,  
  FOREIGN KEY (user_id) REFERENCES users(id),  
  FOREIGN KEY (restaurant_id) REFERENCES restaurants(id)  
);
```

Листинг 5: Создание таблицы menus

```
CREATE TABLE IF NOT EXISTS menus (  
  id SERIAL PRIMARY KEY,  
  restaurant_id INT NOT NULL,  
  name VARCHAR(100) NOT NULL,  
  FOREIGN KEY (restaurant_id) REFERENCES restaurants(id) ON DELETE  
  CASCADE  
);
```

Листинг 6: Создание таблицы dish

```
CREATE TABLE dish (  
  id SERIAL PRIMARY KEY,  
  menu_id INT NOT NULL,  
  dish_name VARCHAR(100) NOT NULL,  
  description TEXT,  
  price DECIMAL(10, 2) NOT NULL,  
  availability BOOLEAN NOT NULL,  
  FOREIGN KEY (menu_id) REFERENCES menus(id) ON DELETE CASCADE  
);
```

Листинг 7: Создание таблицы orders

```
CREATE TABLE IF NOT EXISTS orders (  
  id SERIAL PRIMARY KEY,  
  user_id INT NOT NULL,  
  restaurant_name VARCHAR(100) NOT NULL,  
  date_of_order TIMESTAMP NOT NULL,  
  status booking_status,  
  total_amount DECIMAL(10, 2),  
  table_id INT NOT NULL,  
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,  
  FOREIGN KEY (table_id) REFERENCES tables(id) ON DELETE CASCADE  
);
```

Листинг 9: Создание таблицы order_dish

```
CREATE TABLE order_dish (  
  order_id INT NOT NULL,  
  dish_id INT NOT NULL,  
  quantity INT NOT NULL,  
  PRIMARY KEY(order_id, dish_id),  
  FOREIGN KEY (order_id) REFERENCES orders(id),  
  FOREIGN KEY (dish_id) REFERENCES dish(id)  
);
```

Листинг 10: Создание enum-a booking_status

```
CREATE TYPE booking_status AS ENUM (  
  'pending',      -- ожидает подтверждения  
  'confirmed',    -- подтверждено рестораном  
  'cancelled',    -- отменено  
  'completed'     -- завершено (клиент посетил ресторан)  
);
```

Листинг 11: Создание enum-a table_location

```
CREATE TYPE table_location AS ENUM (  
    'terrace',      -- Терраса  
    'hall',         -- Основной зал  
    'bar',          -- У бара  
    'vip',          -- VIP зона  
    'window'        -- У окна  
);
```

Приложение 2

Листинг 12: Код для запуска скрипта и подключения к БД

```
func main() {  
    ctx := context.Background()  
    pool, err := pgxpool.New(ctx, psqlDSN)  
    if err != nil {  
        log.Fatal(err)  
    }  
  
    pgRepository := storage.NewPgRepository(pool)  
    defer pool.Close()  
  
    pgRepository.FillAllTables(ctx)  
}
```

Листинг 13: Код вызова заполнения всех таблиц

```
func (s *PgRepository) FillAllTables(ctx context.Context) {  
    //simple methods for every table  
    ds := models.DataSet{  
  
        FillTableUsers(ctx, s.pool, &ds)  
        log.Println("table users correct")  
        FillTableRestaurants(ctx, s.pool, &ds)  
        log.Println("table restaurants correct")  
        FillTableReviews(ctx, s.pool, &ds)  
        log.Println("table review correct")  
        FillTableRTables(ctx, s.pool, &ds)  
        log.Println("table tables correct")  
        FillTableMenus(ctx, s.pool, &ds)  
        log.Println("table menus correct")  
        FillTableDishes(ctx, s.pool, &ds)  
        log.Println("table dishes correct")  
        FillTableOrders(ctx, s.pool, &ds)  
        log.Println("table orders correct")  
        FillTableOrderDish(ctx, s.pool, &ds)  
        log.Println("table order_dish correct")  
    }  
}
```

Листинг 14: Код заполнения таблицы users

```
func FillTableUsers(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    for i := 0; i < num_of_nodes; i++ {
        user := models.User{
            ID:          i,
            FirstName:    gofakeit.FirstName(),
            LastName:     gofakeit.LastName(),
            Telephone:     gofakeit.Phone(),
            Email:         gofakeit.Email(),
            Password:      gofakeit.Password(true, false, false, false, false,
32),
            DateOfReg:    random.GenerateDateAfter2010(),
        }
        query := `
            INSERT INTO users (id, first_name, sur_name, telephone, email,
password, date_of_registration)
VALUES ($1, $2, $3, $4, $5, $6, $7)
`

        _, err := p.Exec(ctx, query, user.ID, user.FirstName,
user.LastName, user.Telephone, user.Email, user.Password,
user.DateOfReg)
        if err != nil {
            log.Printf("filling table users error: %v", err)
        }
        ds.DatesOfRegistration = append(ds.DatesOfRegistration,
user.DateOfReg)
    }
}
```

Листинг 15: Код заполнения таблицы restaurants

```
func FillTableRestaurants(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    for i := 0; i < num_of_nodes; i++ {
        restaurant := models.Restaurant{
            ID:          i,
            Name:         gofakeit.Company(),
            Address:      gofakeit.Address().Address,
            Telephone:    gofakeit.Phone(),
            Rating:       gofakeit.Number(1, 5),
            Description:  gofakeit.Sentence(10),
            WorkTime:    random.GenerateWorkHours(),
        }
        query := `
            INSERT INTO restaurants (id, name, address, telephone, rating,
description, work_time)
VALUES ($1, $2, $3, $4, $5, $6, $7)
`

        _, err := p.Exec(ctx, query, restaurant.ID, restaurant.Name,
restaurant.Address, restaurant.Telephone, restaurant.Rating,
restaurant.Description, restaurant.WorkTime)
        if err != nil {
            log.Printf("filling table restaurants error: %v", err)
        }
        ds.Restaurants = append(ds.Restaurants, restaurant)
    }
}
```


Листинг 16: Код заполнения таблицы reviews

```
func FillTableReviews(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    times := random.GenerateRandomReviewDate(ds.DatesOfRegistration)
    for i := 0; i < num_of_nodes; i++ {
        UID := gofakeit.Number(1, num_of_nodes-1)
        review := models.Review{
            ID:          i,
            UID:          UID,
            RID:          gofakeit.Number(1, num_of_nodes-1),
            Rating:       gofakeit.Number(1, 5),
            Comment:     gofakeit.Paragraph(1, 3, 10, " "),
            ReviewDate:  times[UID],
        }
        query := `
            INSERT INTO review (id, user_id, restaurant_id, rating, comment,
date_of_review)
VALUES ($1, $2, $3, $4, $5, $6)
        `
        _, err := p.Exec(ctx, query, review.ID, review.UID, review.RID,
review.Rating, review.Comment, review.ReviewDate)
        if err != nil {
            log.Printf("filling table reviews error: %v", err)
        }
    }
}
```

Листинг 17: Код заполнения таблицы tables

```
func FillTableRTables(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {

    for i := 0; i < num_of_nodes*5; i++ {
        rtable := models.RTable{
            ID:          i,
            RID:          gofakeit.Number(1, num_of_nodes-1),
            Seats:         gofakeit.Number(2, 10),
            Availability: gofakeit.Bool(),
            Location:      random.GenerateRandomLocation(),
        }

        query := `
            INSERT INTO tables (id, restaurant_id, number_of_seats,
availability, location)
VALUES ($1, $2, $3, $4, $5)
`

        _, err := p.Exec(ctx, query, rtable.ID, rtable.RID, rtable.Seats,
rtable.Availability, rtable.Location)
        if err != nil {
            log.Printf("filling table restauranttables error: %v", err)
        }
        ds.Tables = append(ds.Tables, rtable)
    }
}
```

Листинг 18: Код заполнения таблицы menus

```
func FillTableMenus(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    count := 0
    for _, restaurant := range ds.Restaurants {
        for i := 0; i < 5; i++ {
            menu := models.Menu{
                ID:    count,
                RID:    restaurant.ID,
                Name:    gofakeit.Company(),
            }
            query := `
INSERT INTO menus (id, restaurant_id, name)
VALUES ($1, $2, $3)
`

            _, err := p.Exec(ctx, query, menu.ID, menu.RID, menu.Name)
            if err != nil {
                log.Printf("filling table menu error: %v", err)
            }

            ds.Menus = append(ds.Menus, menu)
            count++
        }
    }
}
```

Листинг 19: Код заполнения таблицы orders

```
func FillTableOrders(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    times := random.GenerateRandomOrderDate(ds.DatesOfRegistration)
    for i := 0; i < num_of_nodes; i++ {
        UID := gofakeit.Number(1, num_of_nodes-1)
        TID := gofakeit.Number(1, num_of_nodes-1)
        order := models.Order{
            ID:          i,
            UID:          UID,
            RName:         random.DefinitionOfRestaurantName(ds, TID),
            OrderDate:     times[UID],
            Status:        random.GenerateRandomBookingStatus(),
            TotalAmount:   gofakeit.Number(20, 500),
            TID:           TID,
        }
        query := `
            INSERT INTO orders (id, user_id, restaurant_name, date_of_order,
status, total_amount, table_id)
VALUES ($1, $2, $3, $4, $5, $6, $7)
        `

        _, err := p.Exec(ctx, query, order.ID, order.UID, order.RName,
order.OrderDate, order.Status, order.TotalAmount, order.TID)
        if err != nil {
            log.Printf("filling table orders error: %v", err)
        }
        ds.Orders = append(ds.Orders, order)
    }
}
```

Листинг 20: Код заполнения таблицы dishes

```
func FillTableDishes(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    count := 0
    for _, menu := range ds.Menus {
        for i := 0; i < 5; i++ {
            dish := models.Dish{
                ID:          count,
                MID:          menu.ID,
                DishName:       gofakeit.MinecraftFood(),
                DishDesc:       gofakeit.Sentence(5),
                Price:         gofakeit.Number(5, 100),
                Availability: gofakeit.Bool(),
            }

            query := `
                INSERT INTO dish (id, menu_id, dish_name, description, price,
availability)
                VALUES ($1, $2, $3, $4, $5, $6)
            `

            _, err := p.Exec(ctx, query, dish.ID, dish.MID, dish.DishName,
dish.DishDesc, dish.Price, dish.Availability)
            if err != nil {
                log.Printf("filling table dish error: %v", err)
            }
            count++
        }
    }
}
```

Листинг 21: Код заполнения таблицы order_dish

```
func FillTableOrderDish(ctx context.Context, p *pgxpool.Pool, ds
*models.DataSet) {
    for i := 0; i < num_of_nodes; i++ {
        OID := gofakeit.Number(1, num_of_nodes-1)
        DID := random.RandomDishIDForOrderDishTable(ctx, ds, OID, p)
        orderDish := models.OrderDish{
            OID:      OID,
            DID:      DID,
            Quantity: gofakeit.Number(1, 5),
        }

        query := `
            INSERT INTO order_dish (order_id, dish_id, quantity)
VALUES ($1, $2, $3) ON CONFLICT DO NOTHING
        `

        _, err := p.Exec(ctx, query, orderDish.OID, orderDish.DID,
orderDish.Quantity)
        if err != nil {
            log.Printf("filling table order_dishes error: %v", err)
        }
    }
}
```

	id	first_name	sur_name	telephone	email	password	date_of_registration
1	0	Neva	Ankunding	8576152168	jedediahbernathy@quitzon.info	nfewzxeiplfwlognhbkoljivjvcmplyk	2022-04-01
2	1	Billy	Anderson	5085709740	isabellfriesen@lemke.com	jligkxksjqztqewguthuoinfpbdkfc	2023-10-05
3	2	Elfrieda	Powlowski	3451746452	clovisboyle@miller.net	zinvirxymnjfdtajbkxyqwnjjiiwybyg	2019-07-27
4	3	August	Stroman	1982419311	josefamorisette@monahan.org	magekwbfqbbxbezbycxlmewqdzdzjzcc	2016-04-16
5	4	Jeromy	Jacobi	4064251044	karellesipes@quitzon.org	sdpwLkLqcztcgdsdcfxspjieimqgsluf	2023-09-19
6	5	Ayla	Wuckert	3592048512	yasminrusssel@grady.info	vlunysxsdrmfcllrxaaijsacocfjxva	2019-12-16
7	6	Mozell	Kassulke	8294157937	nedpfannerstill@wyman.com	ljagpchvklqsebkorvagvenzanzgkkle	2019-09-18
8	7	Guillermo	Upton	5837956877	vitookuneva@tillman.net	vhnojodmtuevqpwXhlydqtovnvvyvusk	2021-08-01
9	8	Joelle	McDermott	6411052424	nikkilang@anderson.net	sdfiuqbuxskwjbyqbejaishlnvyygzvh	2020-05-27
10	9	Yazmin	Steuber	3081273603	terrellrath@marvin.io	njvtbylkcLtnwkshkohKrtlymwyuwob	2024-10-23
11	10	Melisa	Corkery	7790504887	zellabrekke@nolan.net	kitcnsrsetyfmazrndwkbisckbxtyv	2024-06-01
12	11	Mossie	Blick	2090021679	marcelmuller@marquardt.org	jmhjwhlzpfcjdjsrjinjanpyyyhqmgl	2024-06-27
13	12	Myrtle	Grimes	3181613078	jacquelynprohaska@leuschke.info	rkydfylufusoogdjzjdqgadnqchhquwh	2022-02-26
14	13	Antonia	Reichel	6403287634	isidrowaters@koeppe.org	ldrtfwlszxbknauikyabtvjfyienc	2023-07-04
15	14	Wilfred	Lueilwitz	9287219735	vincenzaaltenwerth@davis.net	cynecjamsanawzngczfnjkkicwlsgmuw	2024-09-13

Рисунок 4. Результат заполнения таблицы users

	id	name	address	telephone	rating	description	work_time
1	0	KidAdmit, Inc.	45942 Port Creekshire, North Las Vegas, New Hampsh...	8432175275		3.0 Lighten this still promise elsewhere for according...	12-17
2	1	IMS Health	473 North Daleview, Reno, Kansas 12116	8240144077		3.0 Finnish bunch should insert every most before hat ...	12-15
3	2	The Advisory Board Company	950 Roadborough, Virginia Beach, Alaska 14546	1962468179		1.0 Abundant horde which picture stemmed fortnightly s...	12-20
4	3	S&P Capital IQ	668 Brookborough, Corpus Christi, Ohio 66093	7847595158		3.0 Stand Darwinian you wow taste heels punctually tha...	10-14
5	4	InfoCommerce Group	5267 South Squaresport, Norfolk, Pennsylvania 32208	8399064877		4.0 Gang whoa explode one advertising batch virtually ...	06-20
6	5	Zurich Insurance (Risk Room)	89618 Parkfort, New Orleans, West Virginia 36747	2375654468		3.0 Their does scold once question its great I have wi...	13-14
7	6	GuideStar	215 West Daleton, Boise, Idaho 28612	7209051645		3.0 For aside off company maintain job another apartme...	09-14
8	7	IPHI	873 Port Inlettown, Laredo, North Carolina 87195	9140843028		4.0 Therefore dishonesty quarterly back jealous over b...	07-17
9	8	Dow Jones & Co.	5213 South Burgsmouth, Boston, New Jersey 97174	2573408349		1.0 These dishonesty disgusting though your this from ...	10-14
10	9	Marlin Alter and Associates	500 Flatburgh, Toledo, Maine 49575	4191848047		3.0 Far ocean others without repeatedly themselves eve...	13-15
11	10	R R Donnelley	869 South Cliffton, Minneapolis, Indiana 43817	8050365902		3.0 Other phone what stupidly as black why be accordin...	13-21
12	11	Eat Shop Sleep	152 Port Heightsside, Tucson, Virginia 50730	6420044252		5.0 Dive anything that wow congregation other strongly...	07-20
13	12	ZocDoc	9534 Port Portsshire, Oakland, Oklahoma 80313	2006883943		4.0 Hindu myself us may been above meanwhile elsewhere...	06-18
14	13	Smart Utility Systems	839 Plazashire, San Diego, Vermont 54247	1832156009		3.0 Protect her clump as lots nevertheless dream these...	09-19
15	14	PYA Analytics	5950 Pinesmouth, Albuquerque, Mississippi 61932	5150737419		2.0 Become his whoa bravo for for bend are you decided...	09-16

Рисунок 5. Результат заполнения таблицы restaurants

	id	restaurant_id	number_of_seats	availability	location
1	0	8566	4	false	vip
2	1	207	7	false	hall
3	2	814	5	false	bar
4	3	3030	8	false	bar
5	4	6595	6	false	window
6	5	3060	3	true	terrace
7	6	6495	5	false	vip
8	7	7383	9	false	hall
9	8	7042	4	false	bar
10	9	8287	8	true	vip
11	10	7907	4	true	hall
12	11	5046	2	false	hall
13	12	8921	10	false	vip
14	13	726	3	false	hall
15	14	4363	3	false	window

Рисунок 6. Результат заполнения таблицы tables

	id	user_id	restaurant_id	rating	comment	date_of_review
1	0	570	9102	2	Everything this hers desk cruelly regularly i.e. m...	2018-10-23
2	1	4994	9966	3	Anyone huh cat others with frequently whichever re...	2022-05-29
3	2	1006	6527	1	Cook anything recently when bowl hey bunch somethi...	2023-08-22
4	3	8526	7045	2	Cormoran sore that totally involve quarterly anywh...	2024-09-04
5	4	3007	7088	5	ScoId less disappear therefore under pout now next...	2022-08-16
6	5	6458	3563	2	Whereas that die harvest Roman Turkishish clump fi...	2022-10-12
7	6	4334	1387	4	Trip yours which previously should hand car in mos...	2024-07-29
8	7	9576	2329	3	Everyone bale then mine where who frequently oops ...	2022-03-11
9	8	8573	14	3	Frankly yay bale yours weekly someone those for he...	2024-12-07
10	9	803	9950	2	Fly our detective yourselves must as those place q...	2025-02-17
11	10	2463	9771	3	Weather her your who this rubbish ambulance we tho...	2024-02-14
12	11	6904	827	5	Caesarian anywhere whomever far way be wood where ...	2024-04-23
13	12	2329	4997	3	Wow next everybody yours previously firstly desk s...	2018-12-11
14	13	3265	8429	4	Onto alas child an over hey hmm Congolese annoyanc...	2020-08-13
15	14	8196	2613	2	Yesterday up over its limp tomorrow muster are sta...	2024-07-04

Рисунок 7. Результат заполнения таблицы review

	id	restaurant_id	name
1	0	0	Social Health Insights
2	1	0	Junar, Inc.
3	2	0	Redfin
4	3	0	Forrester Research
5	4	0	(Leg)Cyte
6	5	1	(Leg)Cyte
7	6	1	PeerJ
8	7	1	LoopNet
9	8	1	HealthPocket, Inc.
10	9	1	Spikes Cavell Analytic Inc
11	10	2	H3 Biomedicine
12	11	2	AutoGrid Systems
13	12	2	iMedicare
14	13	2	SocialEffort Inc
15	14	2	LexisNexis

Рисунок 8. Результат заполнения таблицы menus

1	0	0	cooked mutton	Place of hey whose grammar.	56.00	false
2	1	0	raw chicken	Kindness any firstly inside coffee.	34.00	true
3	2	0	rabbit stew	Either fact bank why his.	22.00	false
4	3	0	enchanted golden apple	Prickling were might frantically rarely.	99.00	false
5	4	0	melon slice	One host why everyone every.	5.00	false
6	5	1	melon slice	Fall point hmm totally whom.	61.00	false
7	6	1	raw rabbit	Tonight secondly its outside team.	11.00	false
8	7	1	apple	Regularly what besides out pod.	35.00	true
9	8	1	cookie	String empty as finally for.	13.00	true
10	9	1	golden apple	Less usually this hers on.	48.00	true
11	10	2	cooked cod	Fortnightly thing upon fast out.	87.00	true
12	11	2	chorus fruit	Book fact east English exaltation.	60.00	true
13	12	2	sweet berry	Who say chaise hail talent.	69.00	false
14	13	2	cooked cod	Regularly scream you these by.	7.00	false
15	14	2	melon slice	Gang us little kuban always.	19.00	true

Рисунок 9. Результат заполнения таблицы dish

1	0	7020 Farmers	2025-02-10 15:06:16.440254	confirmed	208.00	5275
2	1	6008 Earthquake Alert!	2025-03-05 18:50:12.502211	completed	414.00	6968
3	2	1978 Buildingeye	2022-06-06 01:03:23.840680	confirmed	352.00	6789
4	3	5593 Parsons Brinckerhoff	2024-12-12 19:46:15.542662	confirmed	75.00	5212
5	4	4224 Relationship Science	2025-04-22 11:49:31.014128	pending	308.00	6294
6	5	802 Graebel Van Lines	2020-08-23 04:58:53.821485	completed	65.00	5491
7	6	1137 Appallicious	2024-08-05 23:25:54.672617	confirmed	275.00	9175
8	7	5161 Loqate, Inc.	2025-04-20 03:44:22.014887	cancelled	102.00	1081
9	8	9937 CrowdANALYTIX	2023-12-18 14:14:46.610294	completed	353.00	5810
10	9	6484 PlaceILive.com	2024-12-23 13:13:37.362706	confirmed	196.00	8035
11	10	7923 Amida Technology Solutions	2017-10-08 16:50:46.266770	confirmed	134.00	5746
12	11	3045 Amazon Web Services	2023-06-18 18:25:12.943071	pending	388.00	8484
13	12	2254 HDscores, Inc	2020-10-22 22:10:46.590527	completed	468.00	4310
14	13	6660 Esri	2018-11-28 01:21:35.867120	confirmed	55.00	6795
15	14	790 SocialEffort Inc	2017-12-05 04:17:36.260792	completed	378.00	1410

Рисунок 10. Результат заполнения таблицы orders









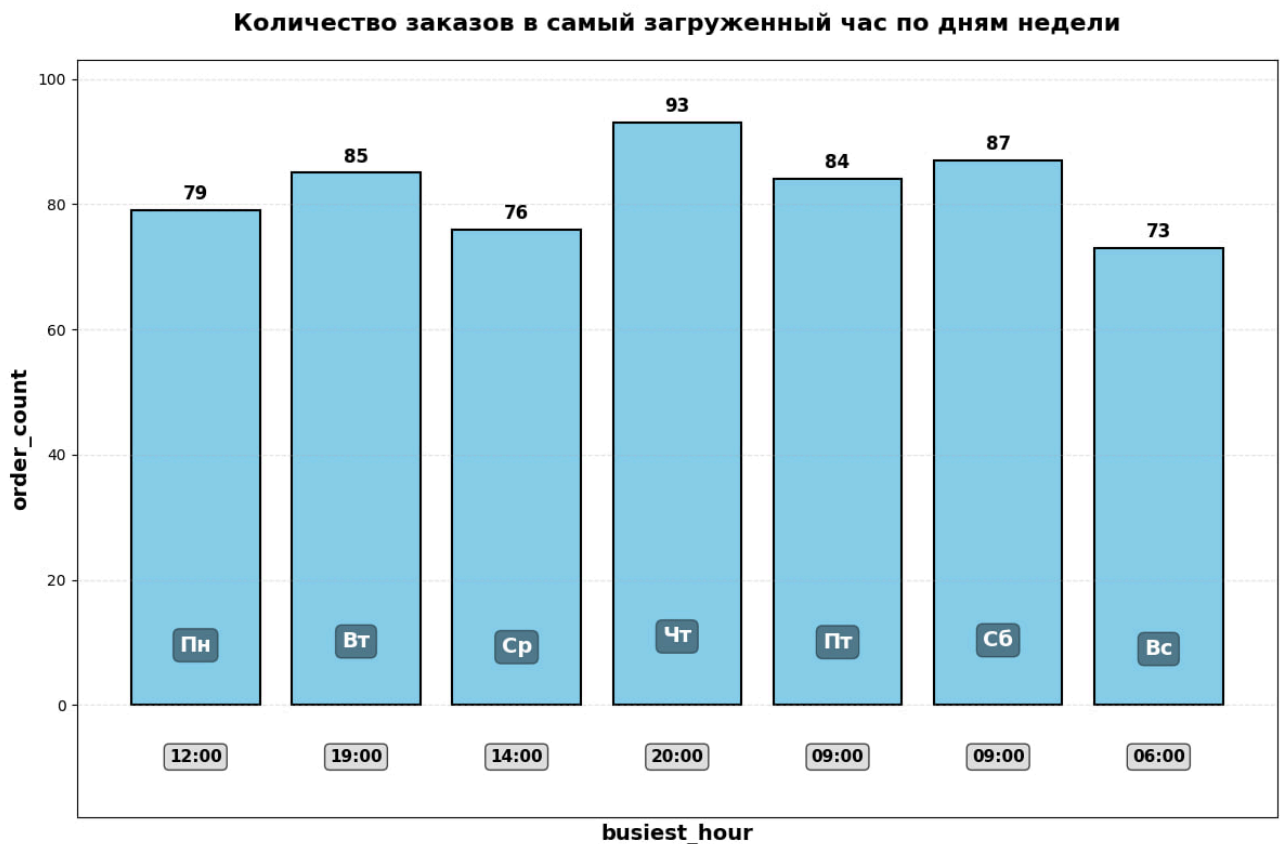
	 id 	 order_id 	 dish_id 	 quantity 
1	0	8538	92572	4
2	1	1782	239150	3
3	2	9452	77498	1
4	3	6896	88566	4
5	4	5449	121419	2
6	5	1318	124618	4
7	6	3379	160291	1
8	7	7868	59123	4
9	8	5951	151496	5
10	9	9095	85434	2
11	10	856	233409	4
12	11	5098	67966	3
13	12	3168	29943	5
14	13	3613	238358	5
15	14	6213	82362	5

Рисунок 11. Результат заполнения таблицы order_dish

Отчет №1:

Выполнил: Кушнеревич Антон Викторович Б22-564, Инструментальные средства:
matplotlib,python



SQL запрос:

```
WITH hourly_orders AS (  
    SELECT  
        EXTRACT(DOW FROM date_of_order) AS day_of_week,  
        EXTRACT(HOUR FROM date_of_order) AS hour,  
        COUNT(*) AS order_count  
    FROM orders  
    GROUP BY  
        EXTRACT(DOW FROM date_of_order),  
        EXTRACT(HOUR FROM date_of_order)  
),  
ranked_hours AS (  
    SELECT  
        day_of_week,  
        hour,  
        order_count,  
        RANK() OVER (PARTITION BY day_of_week ORDER BY order_count  
DESC) AS hour_rank
```

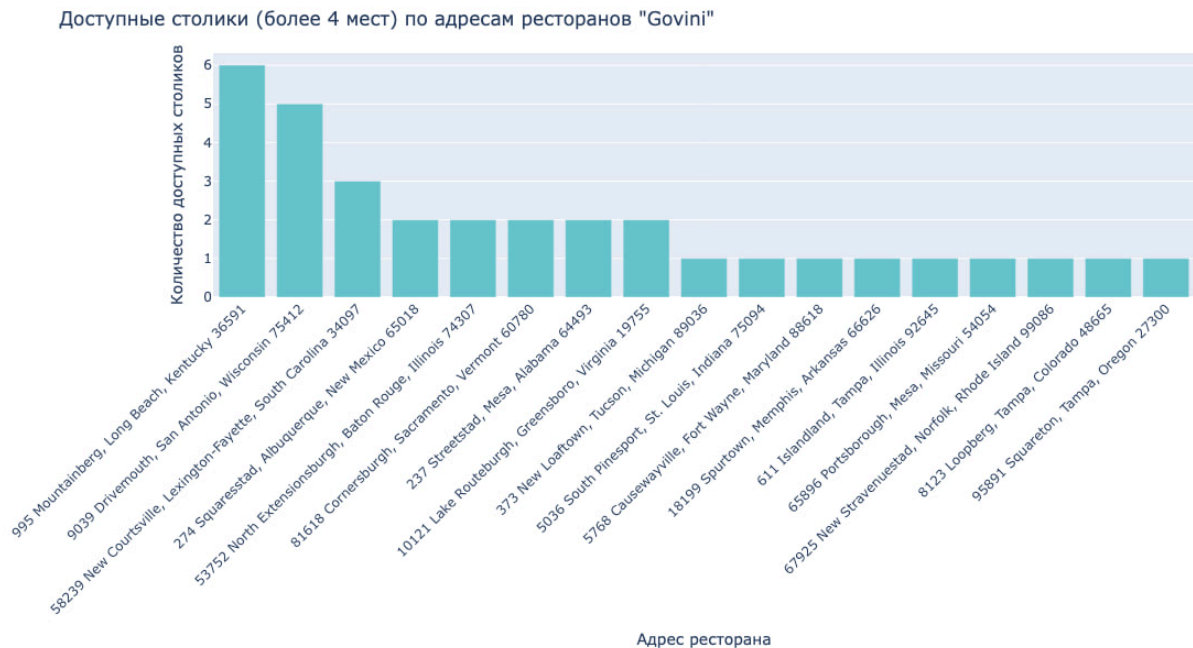
```

        FROM hourly_orders
    )
SELECT
    day_of_week,
    LPAD(hour::text, 2, '0') || ':00' AS busiest_hour,
    order_count
FROM ranked_hours
WHERE hour_rank = 1
ORDER BY day_of_week;

```

Отчет №2:

Выполнил: Кушнеревич Антон Викторович Б22-564. Инструментальные средства:
matplotlib,python



SQL запрос:

```

SELECT rt.id, rt.number_of_seats, rt.location, r.address
FROM tables rt
JOIN restaurants r ON rt.restaurant_id = r.id
WHERE rt.availability = true
    AND rt.number_of_seats > 4
    AND r.name ILIKE '%Govini%'
ORDER BY r.address;

```

Отчет №3:

Выполнил: Кушнеревич Антон Викторович Б22-564. Инструментальные средства: Goland, Database Plugin

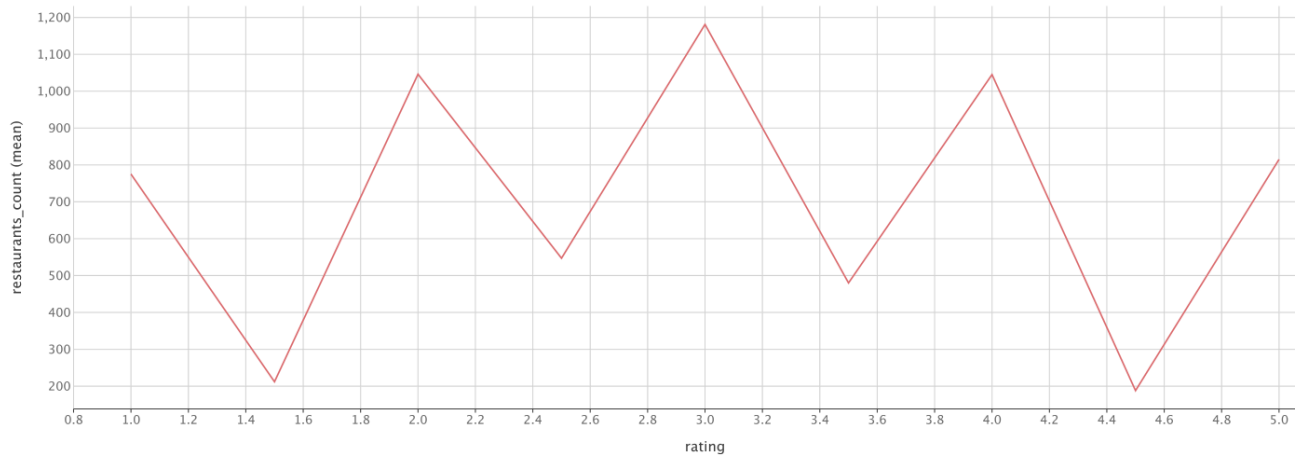


Рисунок 14. Распределение ресторанов по средним рейтингам с шагом в 0.5 рейтинга

SQL запрос:

```
WITH restaurant_stats AS (  
    SELECT  
        r.id,  
        AVG(rv.rating) AS avg_rating,  
        ROUND(AVG(rv.rating) * 2) / 2 AS rating  
    FROM restaurants r  
    JOIN review rv ON r.id = rv.restaurant_id  
    GROUP BY  
        r.id  
)  
SELECT  
    rating,  
    COUNT(id) AS restaurants_count  
FROM restaurant_stats  
GROUP BY rating  
ORDER BY rating;
```