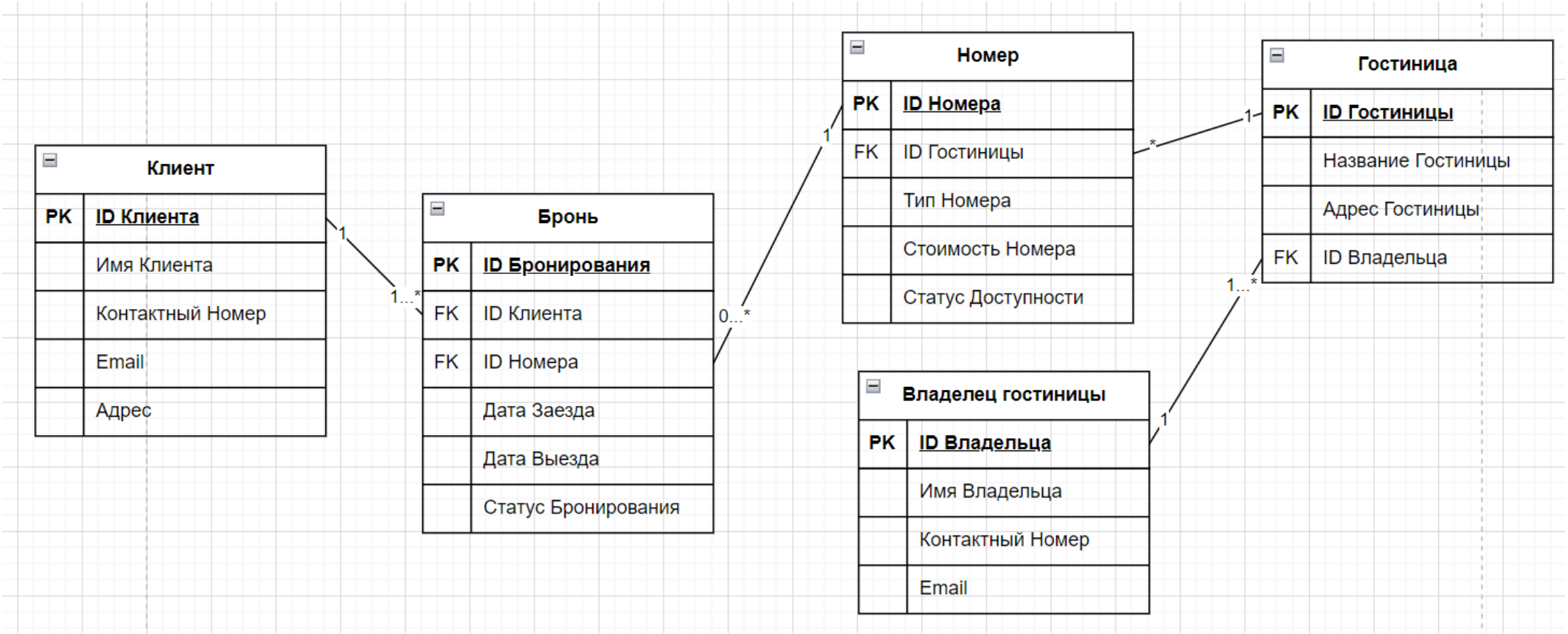
**Моделирование. Нормализация и денормализация**

**Практикуемся**

Отлично! Вот готовая логическая модель:




**Идём далее. Уже можно переходить к физической модели данных, но есть одно НО. НОРМАЛИЗАЦИЯ.**

Проверка нормализации не просто академическое упражнение, она имеет практическую ценность, помогая убедиться, что ваша база данных будет эффективной, гибкой и устойчивой к обычным проблемам, таким как повторение данных, сложности при обновлении и потенциальные противоречия в данных.

**Нормализация** — это процесс организации данных в базе таким образом, чтобы уменьшить дублирование и избыточность. Это делается путем разделения данных на логически связанные таблицы и установления отношений между этими таблицами с помощью ключей.

1. **Уменьшение избыточности**: Нормализация помогает избежать дублирования данных в разных частях базы данных.
2. **Повышение целостности**: Путем разделения данных и связывания их через ключи, нормализация помогает поддерживать точность и целостность данных.
3. **Упрощение обслуживания**: Обновление, удаление или добавление данных становится проще и надежнее, так как информация хранится централизованно.

**Денормализация** — это обратный процесс добавления избыточности в базу данных для улучшения скорости чтения данных. Это делается путем объединения таблиц, добавления избыточных данных или группировки.

1. **Улучшение производительности**: В некоторых случаях денормализация может значительно ускорить процесс чтения данных, снижая количество необходимых операций соединения таблиц.
2. **Оптимизация для специфических запросов**: Денормализация может быть полезной, когда определенные типы запросов выполняются очень часто, и важнее быстрота их выполнения, чем сохранение пространства хранения.
3. **Упрощение схемы базы данных**: В некоторых сценариях денормализация помогает упростить структуру базы данных, делая ее более понятной и легкой для работы.

Мы с вами для нашего проекта пойдём обычным путём - проверим его вплоть до 3ей нормальной формы (нормализации). Потому что у нас нет специфичных нефункциональных требований к проекту. Но прежде чем переходить к этому, давайте рассмотрим в каких случаях нужна денормализация и как это выглядит.

## Принятие решения о нормализации данных в базе данных — это важный этап проектирования, требующий взвешенного подхода. Вот несколько ключевых аспектов, которые следует учитывать:

#### Требования к данным

1. **Анализ требований**: Оцените, какие запросы будут выполняться наиболее часто. Если приоритетом являются операции вставки, обновления и удаления, (то есть не чтения) нормализация может быть более предпочтительна.

2. **Объем данных**: Если в базе данных предполагается большой объем данных, нормализация поможет управлять этими данными более эффективно.

3. **Целостность данных**: Если важна высокая целостность(согласованность) и точность данных, нормализация необходима.

**Требования к нагрузке**

1. **Скорость чтения**: Если основной упор делается на операции чтения и аналитические запросы, которые требуют быстрого доступа к большим объемам данных, денормализация может быть оправдана.

2. **Сложность запросов**: Нормализованные схемы часто требуют сложных запросов SQL с множественными операциями соединения. Если это недопустимо с точки зрения производительности, денормализация может помочь упростить запросы.

**Требования к изменениям**

1. **Масштабируемость**: Нормализованные базы данных обычно более масштабируемы и адаптируемы к изменениям.

2. **Обновления и изменения**: В базах данных, где часто происходят изменения структуры или содержания данных, нормализация облегчает внесение этих изменений.

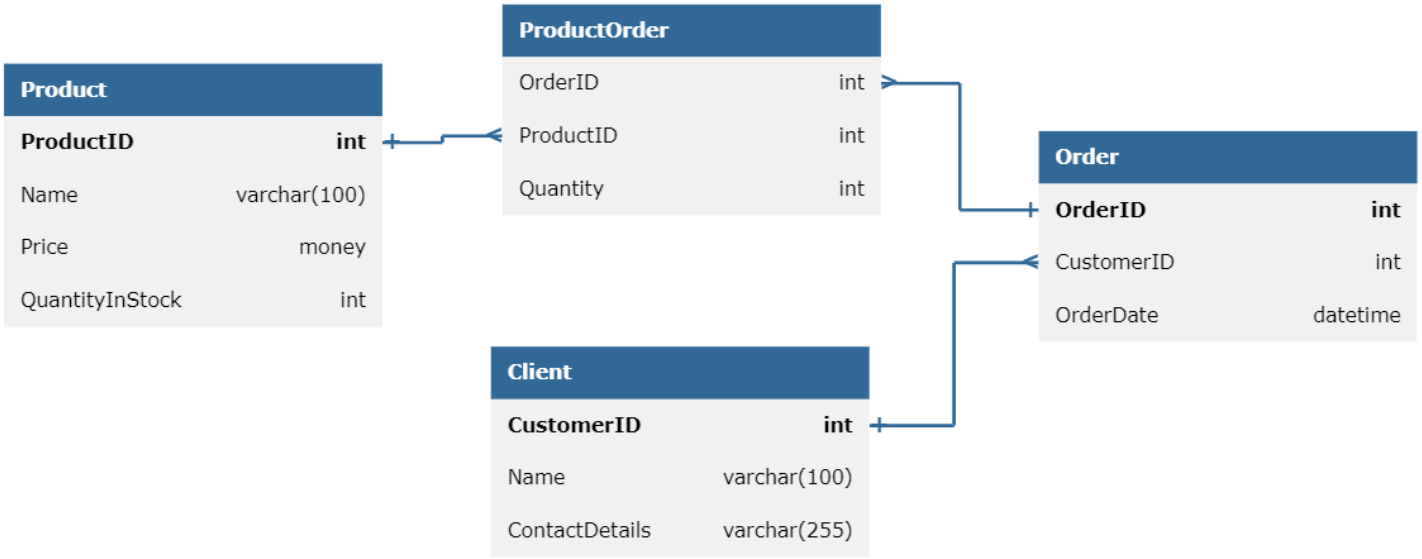
В итоге, учитывая разные требования по этим критериям, вы можете понять - нужна вам нормализация или нет. Пример.

**Бизнес сценарий для нормализации**: Небольшой магазин, который часто обновляет ассортимент товаров и управляет заказами клиентов. Важна точность и актуальность информации о товарах и заказах.

1. Таблица "Товары": Хранит информацию о товарах (ID товара, наименование, цена, количество на складе).
2. Таблица "Клиенты": Содержит данные о клиентах (ID клиента, имя, контактные данные).
3. Таблица "Заказы": Включает информацию о заказах (ID заказа, ID клиента, дата заказа).
4. Таблица "Заказанные Товары": Связывает заказы с товарами (ID заказа, ID товара, количество).

Преимущества:

- Уменьшение избыточности данных.
- Улучшение целостности данных, облегчение обновления информации о товарах.
- Легкость в управлении заказами и отслеживании запасов.




**Бизнес сценарий для денормализации**: Тот же магазин, но с акцентом на быстрое предоставление информации для аналитических целей, например, для выявления трендов покупок.

1. Таблица "Заказы": Объединяет данные о заказах и клиентах (ID заказа, дата заказа, имя клиента, контактные данные, список заказанных товаров с их ценами и количествами).

Преимущества:

- Ускорение аналитических запросов за счет снижения количества операций соединения.
- Упрощение отчетности по продажам и анализа поведения клиентов.

| Order | |
|---|---|
| **OrderID** | **int** |
| OrderDate | datetime |
| CustomerName | varchar(100) |
| ContactDetails | varchar(255) |
| OrderedProducts | text |

## Перейдём к проверке нормализации нашей логической модели данных

Проверка нормализации логической модели данных — это процесс, который следует проводить последовательно, начиная с первой нормальной формы (1НФ) и продвигаясь до третьей нормальной формы (3НФ) или даже до более высоких форм, если это необходимо (мы рассмотрим только 3, это общепринятый стандарт).

Вот общий алгоритм проверки нормализации:

1. **Проверка каждой сущности по очереди**:
    - Проанализируйте каждую сущность отдельно, убедившись, что она соответствует всем требованиям текущей нормальной формы, прежде чем переходить к следующей.
2. **Документирование**:
    - Важно документировать процесс нормализации, чтобы понимать, какие изменения были сделаны и почему.
3. **Итеративный процесс**:

- Нормализация — это итеративный процесс. Возможно, вам придется возвращаться к предыдущим шагам, если находите новые зависимости или избыточности.

4. **Баланс между нормализацией и практичностью**:

   - Не всегда необходимо достигать высоких уровней нормализации, особенно если это приводит к излишней сложности или снижению производительности. Найдите баланс, который лучше всего соответствует вашим требованиям. Помните - с каждым дальнейшим продвижением по нормальным формам (первая, вторая, третья...) вы всё дальше и дальше от денормализации.

**Давайте возьмём одну сущность (Клиент) и проверим её на соответствие трём первым нормальным формам.**

**Критерии 1НФ**:

1. Каждый атрибут должен содержать только атомарные значения, то есть значения не делимые на более мелкие части (также запрещены списки\массивы и так далее - только простые типы данных).
2. Все записи(строки) в таблице должны быть уникальными (то есть отличаться хотя бы по первичному ключу).

ID Клиента (PK) - например "aeabcbaa-e4ba-44bf-a706-89ceddbf580f", атомарное. Благодаря ему также достигается уникальность каждой строки.

Имя Клиента - например "Иван", атомарное.

Контактный Номер - например "+79874563212", атомарное.

Email - например "ivanmail@mail.ru", атомарное.

Адрес - например "Тамбов, Ивановская, 1, 95", не атомарное.

Нам нужно вместо атрибута Адрес добавить атрибуты, которые будут атомарными - Город, Улица, Дом, Квартира.

В итоге вот такая сущность получается, которая соответствует 1НФ:

- ID Клиента (PK)
- Имя Клиента

- Контактный Номер
- Email
- Город
- Улица
- Дом
- Квартира

**Критерии 2НФ**:

1. Таблица уже должна быть в 1НФ.
2. Все атрибуты должны зависеть от первичного ключа, а если он составной - то от всего составного первичного ключа, а не только от его части.

Учитывая, что "ID Клиента" является простым (не составным) первичным ключом, все остальные не ключевые атрибуты ("Имя Клиента", "Контактный Номер", "Email", "Город", "Улица", "Дом", "Квартира") полностью зависят от этого ключа. То есть наша сущность уже находится во 2НФ.

***Пример 2НФ для составного ключа. Исходная таблица (не в 2НФ):

| СтудентID | КурсID | Преподаватель | Оценка |
|---|---|---|---|
| 1 | 101 | Иванов | 5 |
| 1 | 102 | Петров | 4 |
| 2 | 101 | Иванов | 3 |
| 2 | 103 | Сидоров | 4 |

В этой таблице `(СтудентID, КурсID)` является составным ключом. Однако здесь есть проблема: информация о преподавателях зависит только от `КурсID`, а не от всего составного ключа. Это нарушает 2НФ, так как в 2НФ каждый не ключевой атрибут должен быть зависим от всего первичного ключа.

Чтобы привести таблицу в 2НФ, мы разделим её на две таблицы. Таблица `Курсы`:

| КурсID | Преподаватель |
|---|---|
| 101 | Иванов |
| 102 | Петров |
| 103 | Сидоров |

Таблица `Оценки`:

| СтудентID | КурсID | Оценка |
|---|---|---|
| 1 | 101 | 5 |
| 1 | 102 | 4 |
| 2 | 101 | 3 |
| 2 | 103 | 4 |

Теперь информация о преподавателях изолирована в таблице `Курсы`, где она зависит только от `КурсID`, а оценки студентов находятся в таблице `Оценки`, где каждая оценка функционально зависит от всего составного ключа `(СтудентID, КурсID)`. Это соответствует требованиям 2НФ.

**Критерии 3НФ**:

1. Таблица уже должна быть в 2НФ.
2. Не должно быть транзитивных зависимостей, то есть не ключевые атрибуты не должны зависеть от других не ключевых атрибутов (должны зависеть только от первичного ключа).

Необходимо убедиться, что нет зависимостей между не ключевыми атрибутами. Например, атрибут "Город" не должен зависеть от "Имя Клиента" или "Email". В нашей сущности каждый атрибут, зависит непосредственно только от "ID Клиента", а не от других не ключевых атрибутов. Значит наша сущность соответствует 3НФ.

***Рассмотрим пример для понимания приведения таблицы к 3НФ. Исходная таблица (не в 3НФ):

| Студент ID | Курс | Преподаватель Курса |
|---|---|---|
| 1 | Математика | Профессор Иванов |
| 2 | Физика | Профессор Петров |
| 1 | Физика | Профессор Петров |

В этой таблице информация о преподавателе курса зависит от названия курса, а не от студента. Это нарушает правила 3НФ.

Поэтому нужно разделить таблицы:

| Студент ID | Курс |
|---|---|
| 1 | Математика |
| 2 | Физика |
| 1 | Физика |

| Курс | Преподаватель Курса |
|---|---|
| Математика | Профессор Иванов |
| Физика | Профессор Петров |

**Вопрос для теста - какой атрибут мешает сущности Гостиница быть в 3НФ?**

(также попробуйте проверить на нормализацию до 3НФ всю логическую модель в draw.io, на следующем шаге вы получите её рисунок и файл для импорта)