1. Цель проекта

Разработать витрину данных passengers_info на базе PostgreSQL для аналитического подразделения авиакомпании. Витрина будет использоваться в различных запросах и дашбордах, что позволит:

- Оптимизировать выполнение сложных SELECT-запросов
- Обеспечить единый подход к определению ключевых показателей по пассажирам
- Снизить избыточность логики в аналитических отчётах

2. Описание задачи

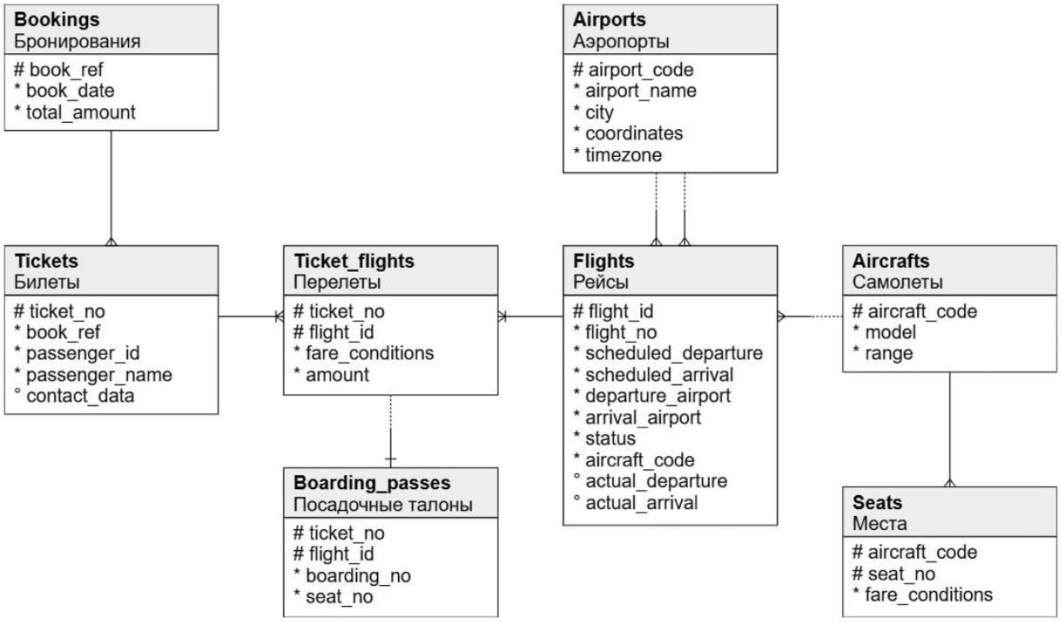
Руководитель аналитического подразделения запрашивает витрину по пассажирам с набором метрик и атрибутов, характеризующих активность и предпочтения клиентов авиакомпании.

3. Наименования и описание полей

- [ ] passenger_id -- идентификатор клиента
- [ ] total_tickets -- общее количество билетов
- [ ] total_tickets_amount -- общая сумма билетов
- [ ] avg_tickets_amount -- средняя стоимость билета
- [ ] average_flights -- среднее количество пересадок в билете
- [ ] more_often_city_from -- наиболее частый город вылета
- [ ] more_often_city_to -- наиболее частый город прилета
- [ ] preffered_airport -- предпочитаемый аэропорт (наиболее частый)
- [ ] preffered_seat -- наиболее частое место
- [ ] preffered_conditions -- наиболее частый класс билета
- [ ] phone_number -- номер телефона
- [ ] email, -- email
- [ ] fio, -- имя, фамилия, где можно
- [ ] total_range -- общее значение "налетанных" километров

4. Источники данных

Рисунок L.2. Диаграмма схемы Bookings



5. Логика формирования

1. Подсчёт общего количества билетов и суммы расходов на билетах.
2. Вычисление среднего значения стоимости билета и среднего количества сегментов:
    - Использовать CTE flight_counts: подсчитать число сегментов на каждый билет.
3. Определение наиболее частых городов вылета и прилёта:
    - CTE city_counts + RANK(); при наличии нескольких лидеров возвращать NULL.
4. Выбор предпочитаемых аэропорта, места и класса:
    - Аналогичный подход с оконными функциями и MODE().
5. Извлечение контактных данных из JSONB-поля contact_data.
6. Суммирование пройденного диапазона километров по всем рейсам.
7. Собрать итоговый запрос с группировкой по passenger_id и загрузить в материализованный вид.

6. Технические требования

- База: **PostgreSQL**, версия не ниже 12.

- Таблица-результат: создание **MATERIALIZED VIEW** passengers_info.

7. Критерии приёмки

- ALL поля заполнены корректно в соответствии с описанием.
- Производительность запроса: общее время сборки витрины не превышает 10 минут.
- Документация и код выложены в GitHub-репозиторий.