PostgresPro

Про-Shardman*

Алексей Борщев
Павел Конотопов

Postgres Professional

*** для SQL-разработчиков**

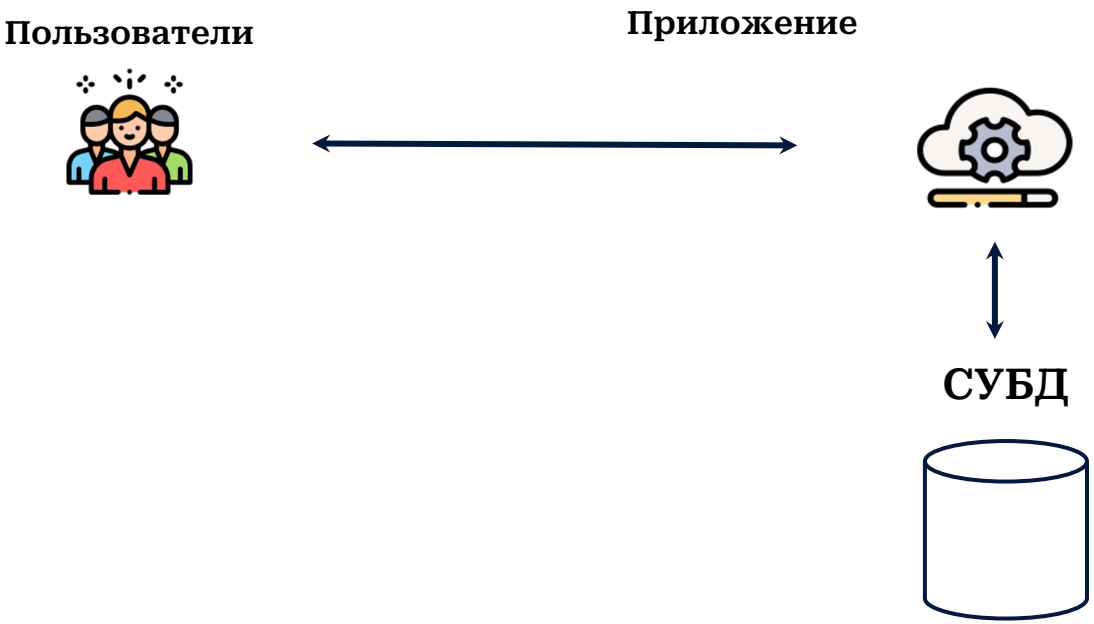
Про что этот доклад?

- Когда нужен Shardman?
- Архитектура
- Управление
- Подключение
- Служебные схемы, представления и функции
- Распределенные таблицы
- Ключ распределения
- Запросы
- Последовательности и функции
- Миграция данных

Когда нужен Shardman?

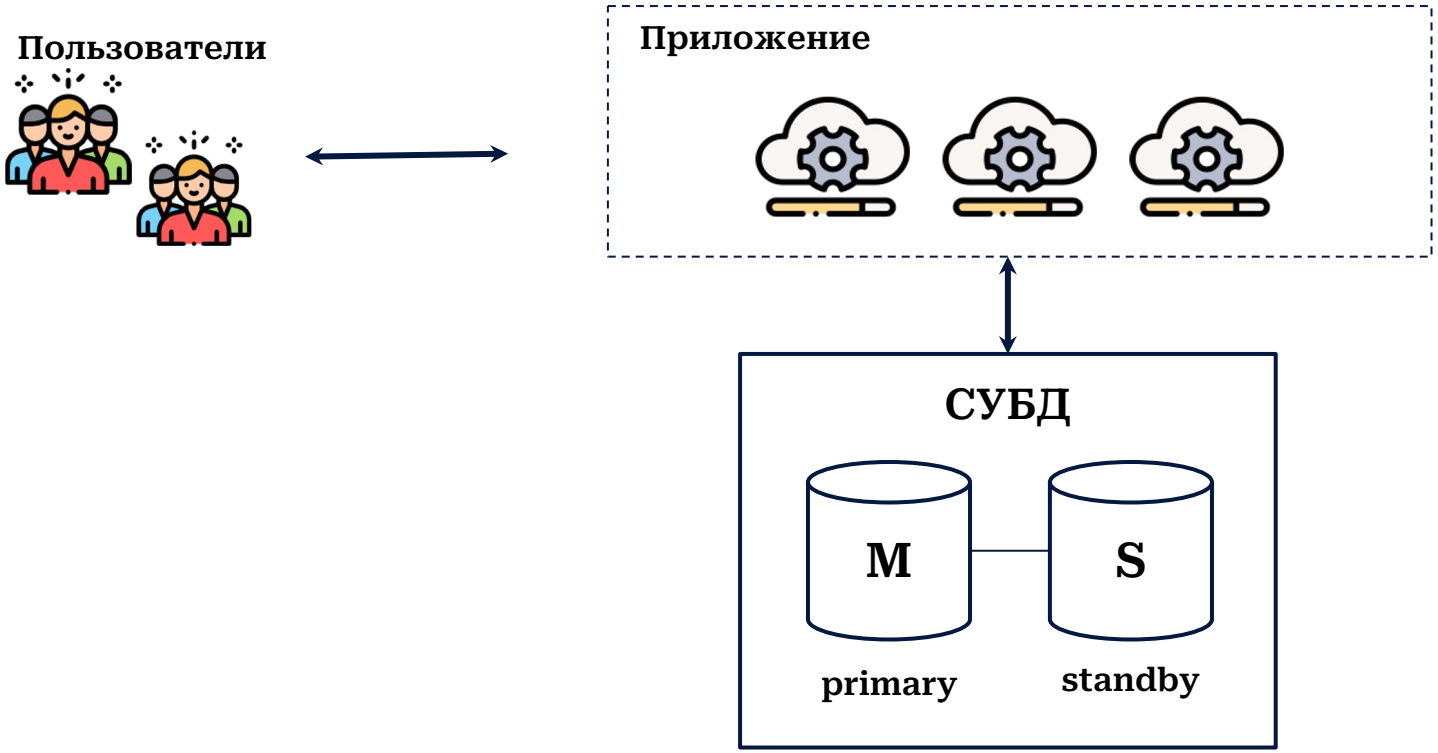
- **Большой рабочий объем данных**
- **Большое количество соединений**
- **Интенсивные операции ввода-вывода**
- **Логика, потребляющая слишком много CPU**
- **Не помещается в один сервер!**

Масштабирование

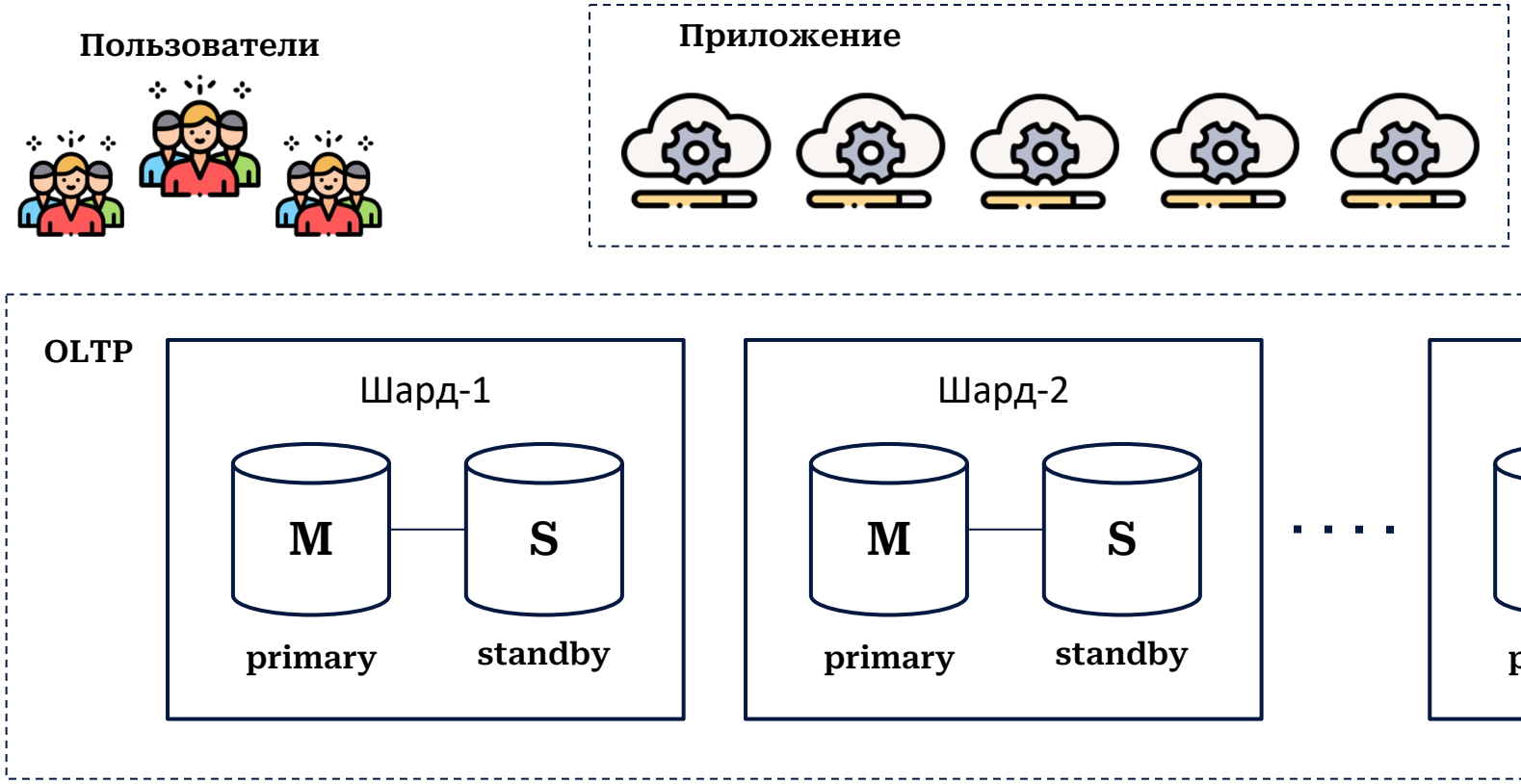


Масштабирование

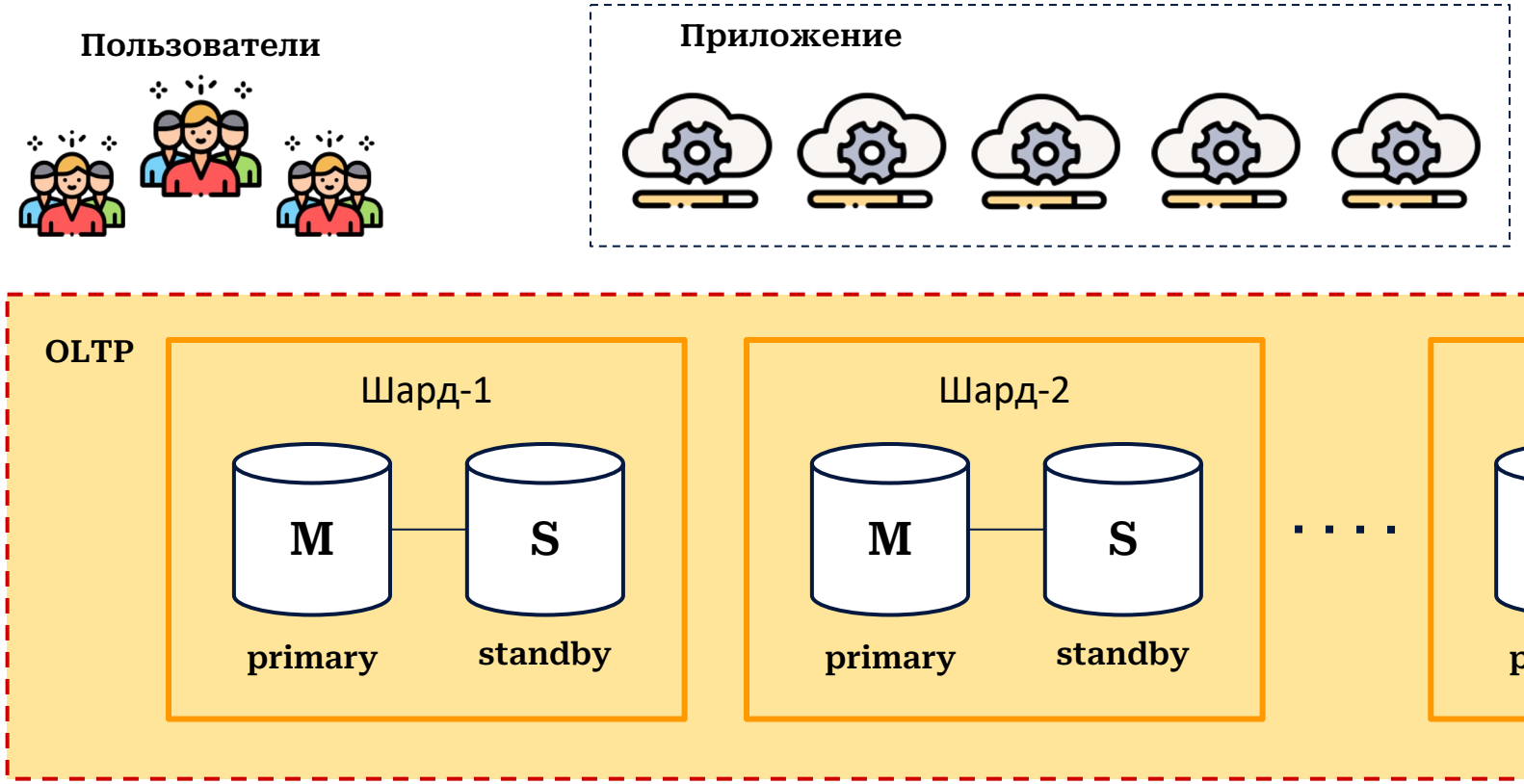
Масштабирование



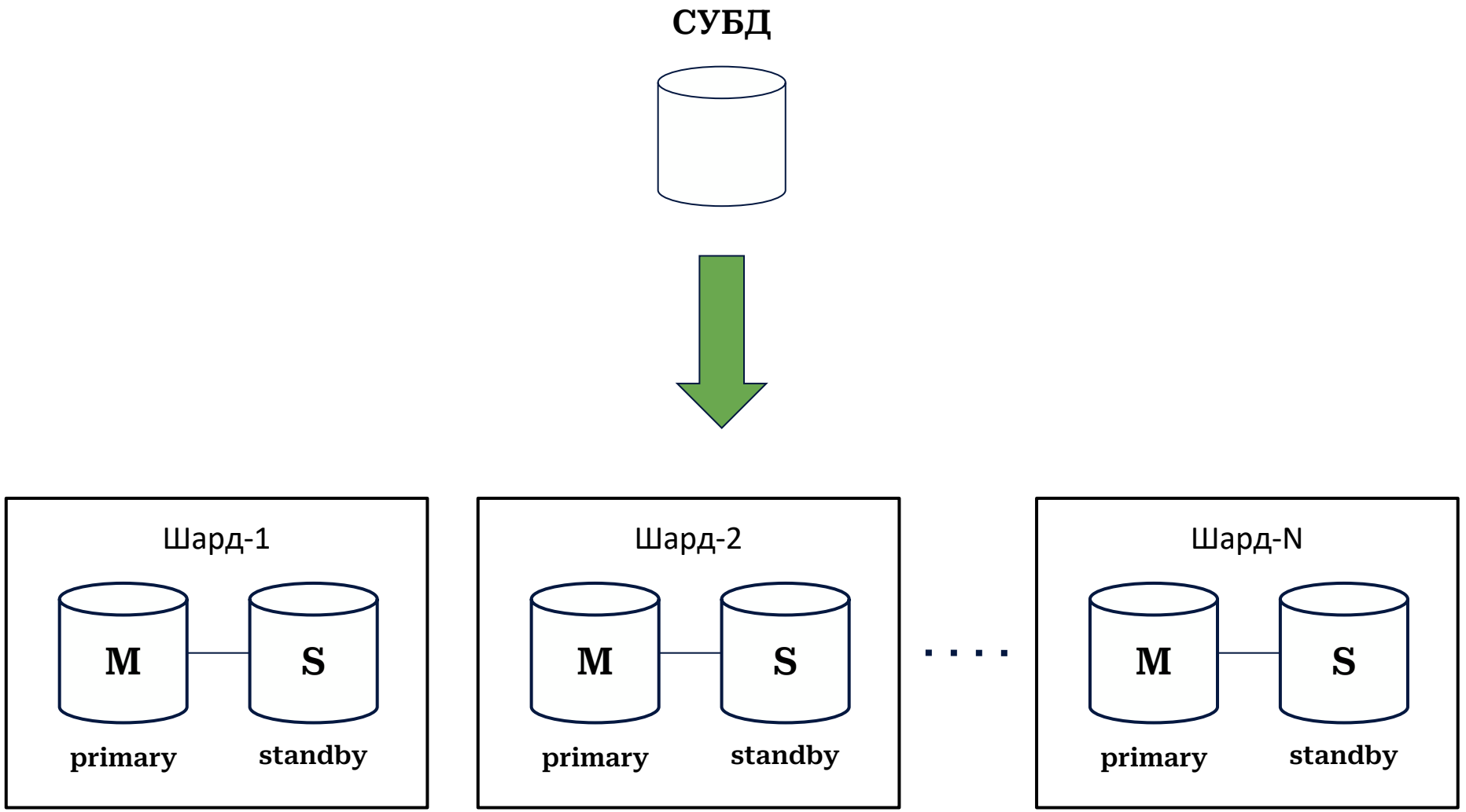
Масштабирование



Масштабирование



Масштабирование



Архитектура

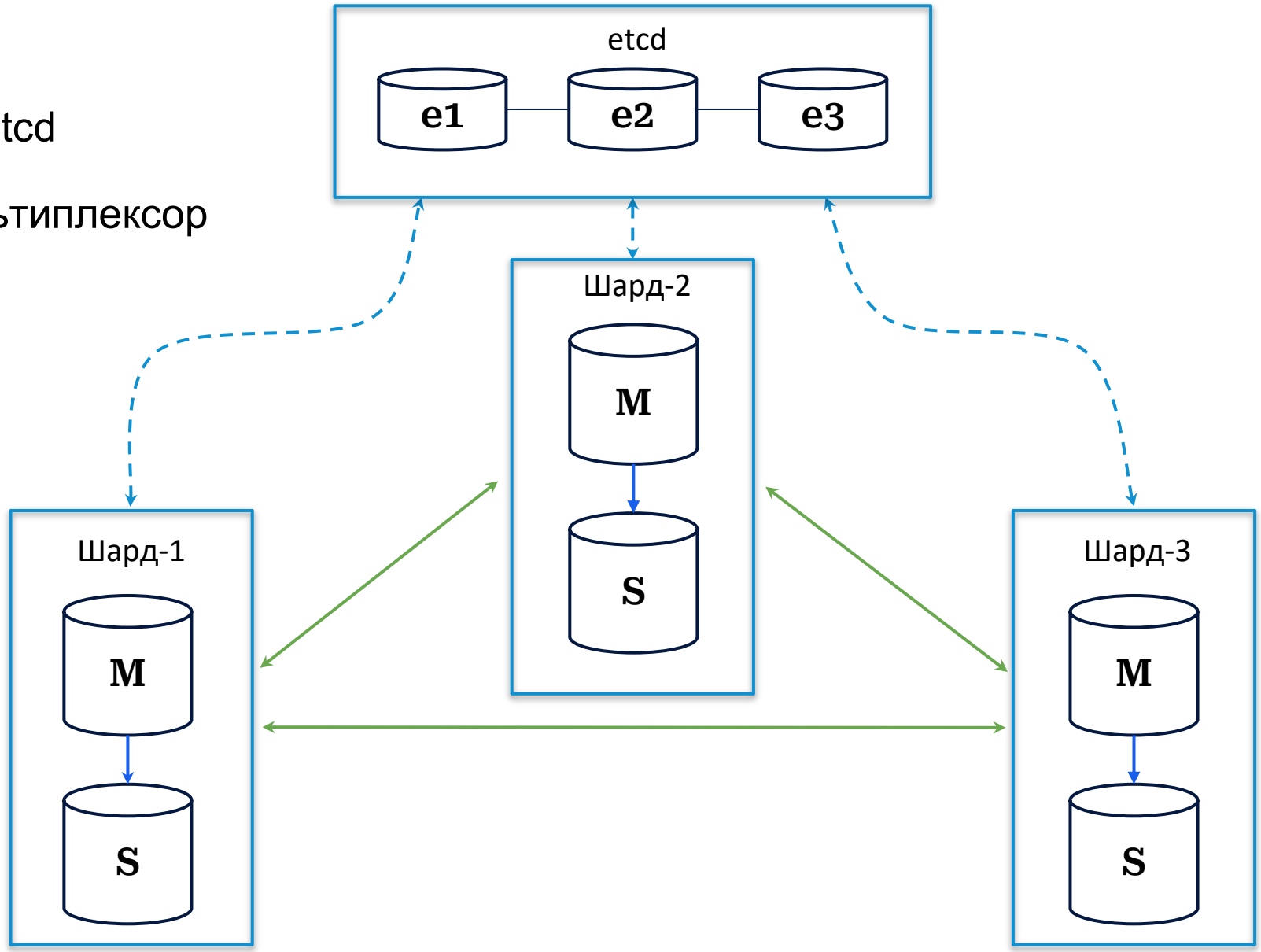
- **Секционирование таблиц**
- **Интерконнект**
  - Мультиплексор соединений (fdw + mp_fdw)
- **Репликация**
  - Физическая
  - Логическая
- **Распределенные транзакции**
  - 2PC
  - CSN – Commit Sequence Number

Архитектура

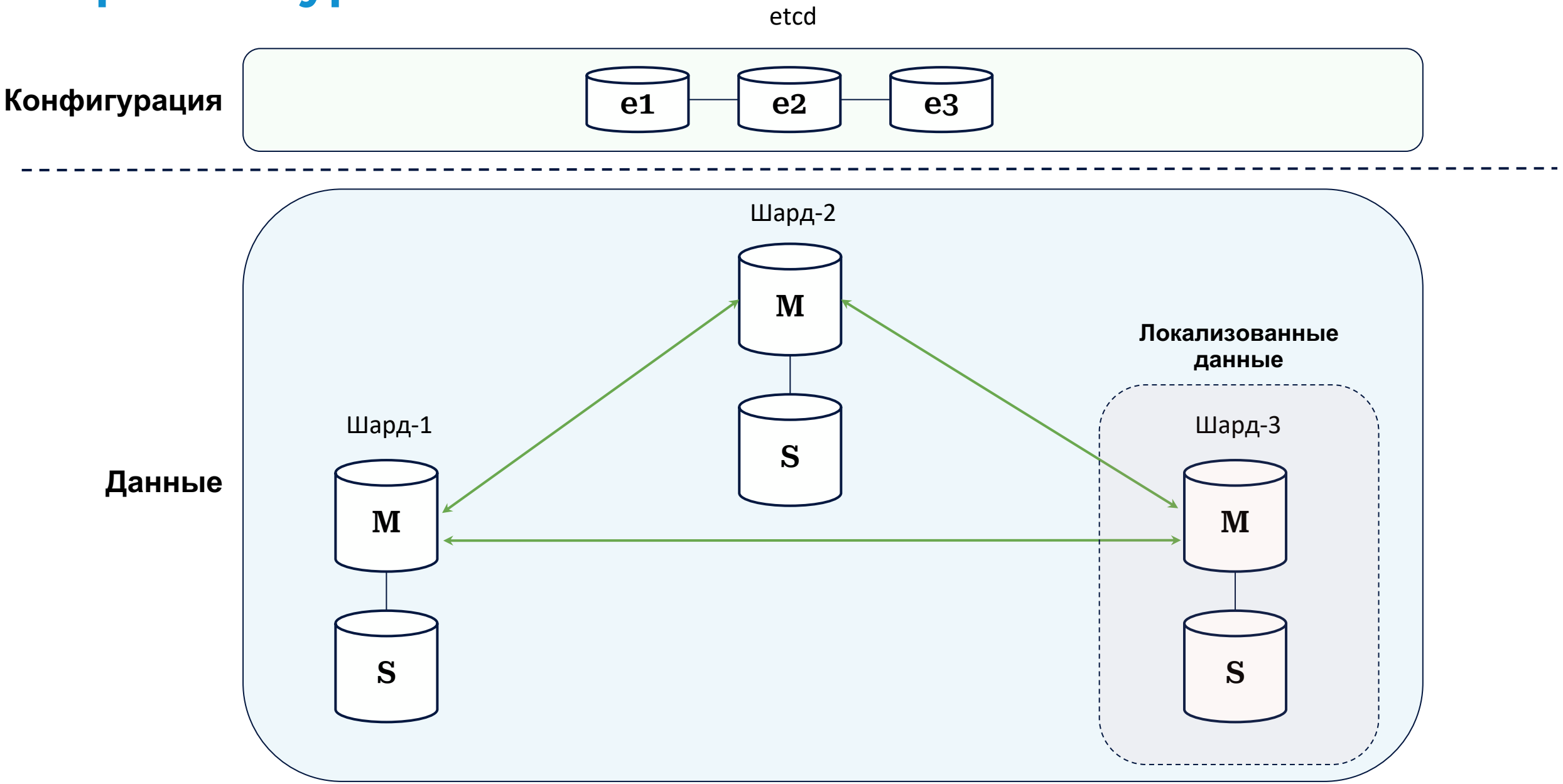
коммуникация с etcd

fdw/mp fdw – мультиплексор

репликация



Архитектура



Управление кластером

- **shardmand**
  - служба (daemon)
  - запуск и остановка
  - отказоустойчивость
  - репликация

- **shardmanctl**
  - утилита управления
  - топология кластера
  - управление
    - конфигурацией кластера
    - топологией шард
    - узлами

Подключение

- **Пока нет встроенного Proxy/Load Balancer**
  - Proxima – в разработке
- **Перечислить все узлы кластера**
- **Указать опции**
  - libpq
    - `node1,node2/postgres?target_session_attr=read-write`
  - libpq 16+
    - `node1,node2/postgres?target_session_attr=primary&load_balance_hosts=random`
  - jdbc
    - `node1,node2/postgres?loadBalanceHosts=true&targetServerType=primary`

А что внутри?

- **Параметры настройки –** `SHOW shardman.*`
- **Номер шарда –** `shardman.rgid`
- **Изменение параметров**
  - `shardmanctl config update -d @json`
  - `shardmanctl set parameter=value`
- **Схема** `shardman` **– полезные таблицы и функции**
- **Служебная схема** `shardman_internal`

Служебные схемы

- **shardman**
  - `sequence`
    список глобальных последовательностей
  - `sharded_tables`
    список сегментированных таблиц
  - `global_tables`
    список глобальных таблиц
    …

- **shardman_internal**
  - fdw для глобальных таблиц

- **руками не лезть!**

Полезные функции в схеме shardman

- **`global_analyze`**
  собрать статистику по распределенным таблицам

- **`broadcast_sql/broadcast_all_sql`**
  выполнить на всех узлах SQL

- **`get_partition_for_value`**
  получить имя партиции для ключа распределения

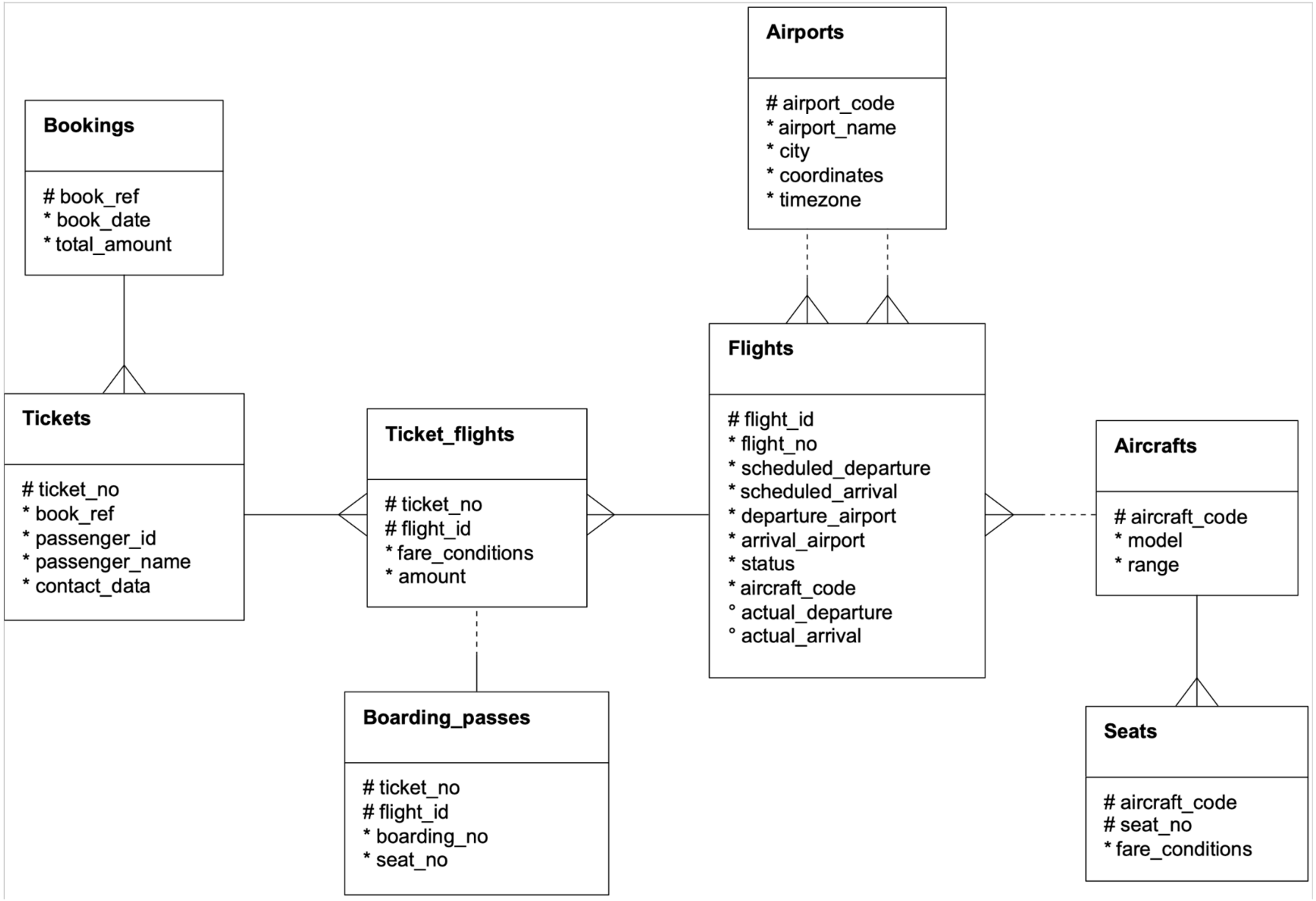
- **`execute_query`**
  использовать для построения view

Полезные представления в схеме shardman

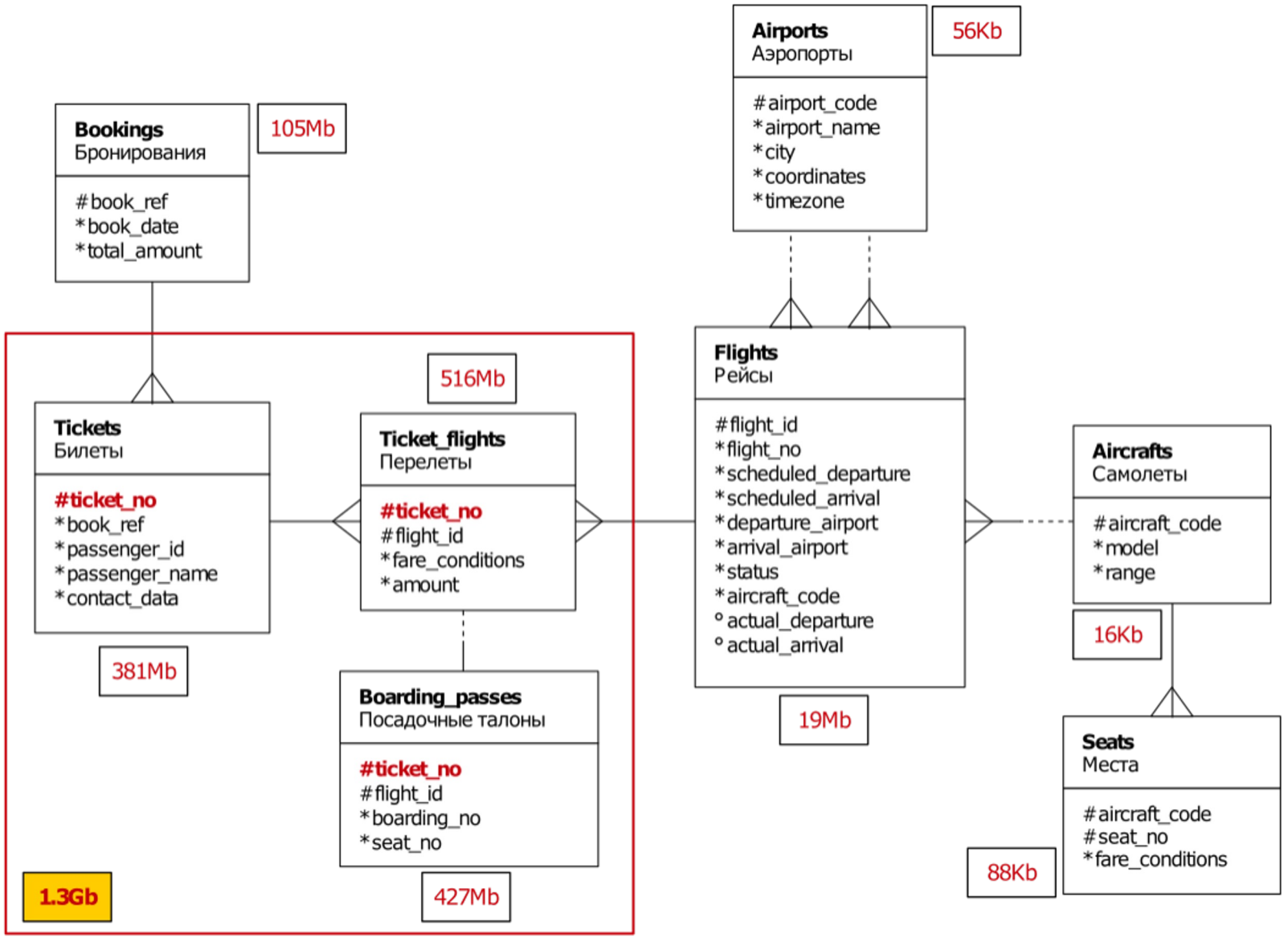
- **silk_**
  - backends
  - connects
  - routes
  - pending_jobs
  - lock_graph

- **gv_stat_** – добавлен **rgid**
  - activity
  - database
  - locks
  - shmem_allocations
  - и прочие!

БД «Авиаперевозки»



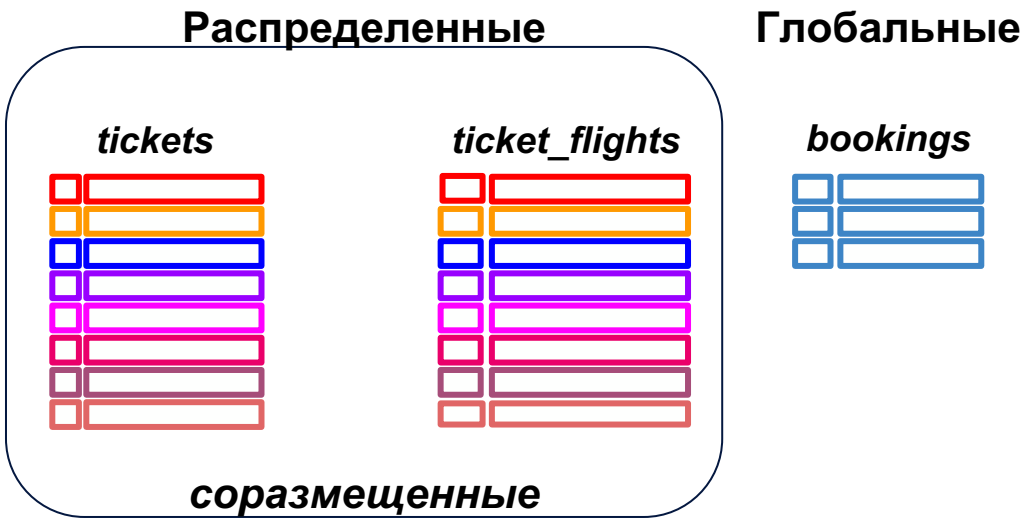
Простая миграция

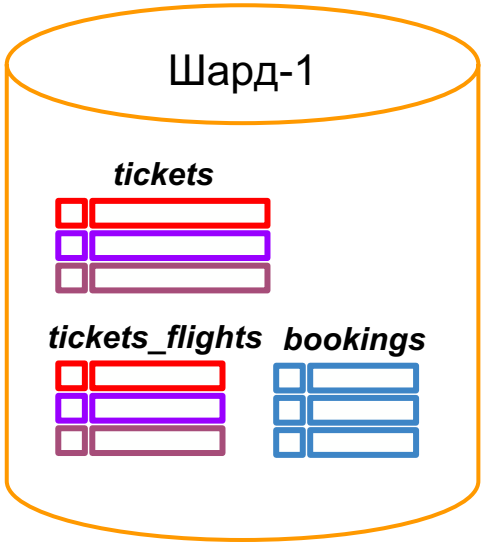


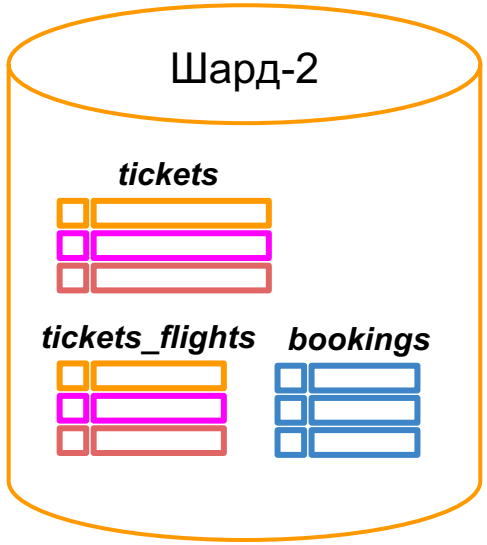
Простая миграция

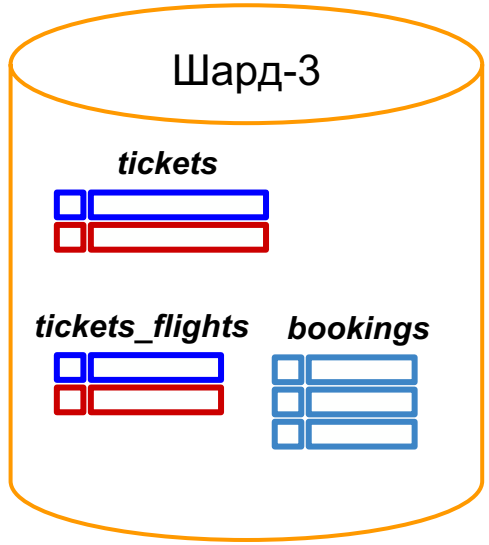
- Не требует изменение схемы
- Не требует переписывания запросов
- Миграция данных без трансформации
- Подходит для несложных схем

Типы таблиц


Распределенные
Глобальные
tickets
ticket_flights
bookings
соразмещенные


Шард-1
tickets
tickets_flights
bookings


Шард-2
tickets
tickets_flights
bookings


Шард-3
tickets
tickets_flights
bookings

Глобальные таблицы

- `CREATE TABLE bookings … WITH (global);`
  - На всех шардах – одинаковые данные
  - Под капотом – репликация на триггерах
  - Необходим Primary Key,
    причём прямо в CREATE TABLE

Подходят: для справочников

Не подходят: для больших или часто изменяемых таблиц

```
select * from shardman.global_tables;
```

Шардированные таблицы

```
CREATE TABLE tickets ...
WITH (
       distributed_by = 'ticket_no',
       num_parts = 12
      );
```

distributed_by – поле таблицы,
ключ шардирования

num_parts – количество частей разбиения

Уникальные индексы (в т.ч. PK) поддерживаются,
если в них входит ключ шардирования

Колоцированные таблицы

```
CREATE TABLE ticket_flights ...
WITH (
       distributed_by = 'ticket_no',
       colocate_with  = 'tickets'
      );
```

distributed_by – поле таблицы, ключ шардирования

colocate_with — имя распределенной таблицы

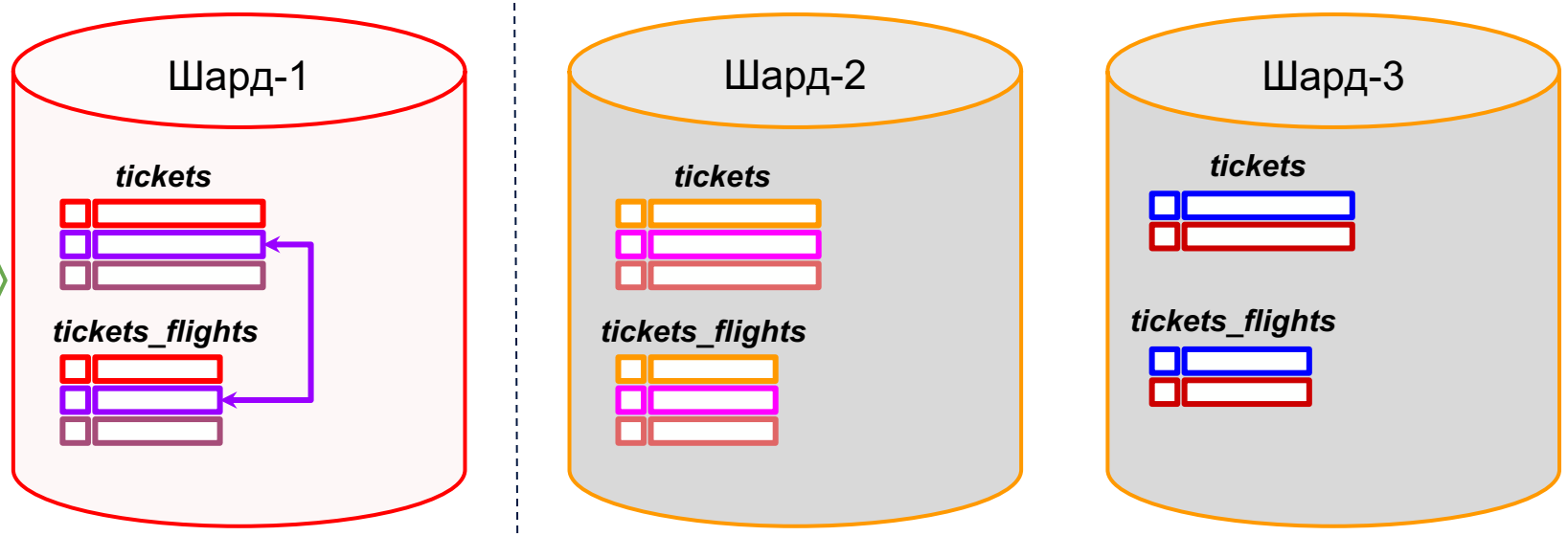
Рекомендуется (но не требуется), чтобы колоцированная таблица ссылалась на распределенную по Foreign Key

`select * shardman.sharded_tables;` – столбец `colocated_with`

Partition pruning (отсечение разделов)

- планировщик анализирует FROM и WHERE
- исключает ненужные разделы
- выполняет операции с разделами, содержащими данные
- prepare и pl/pgsql функции

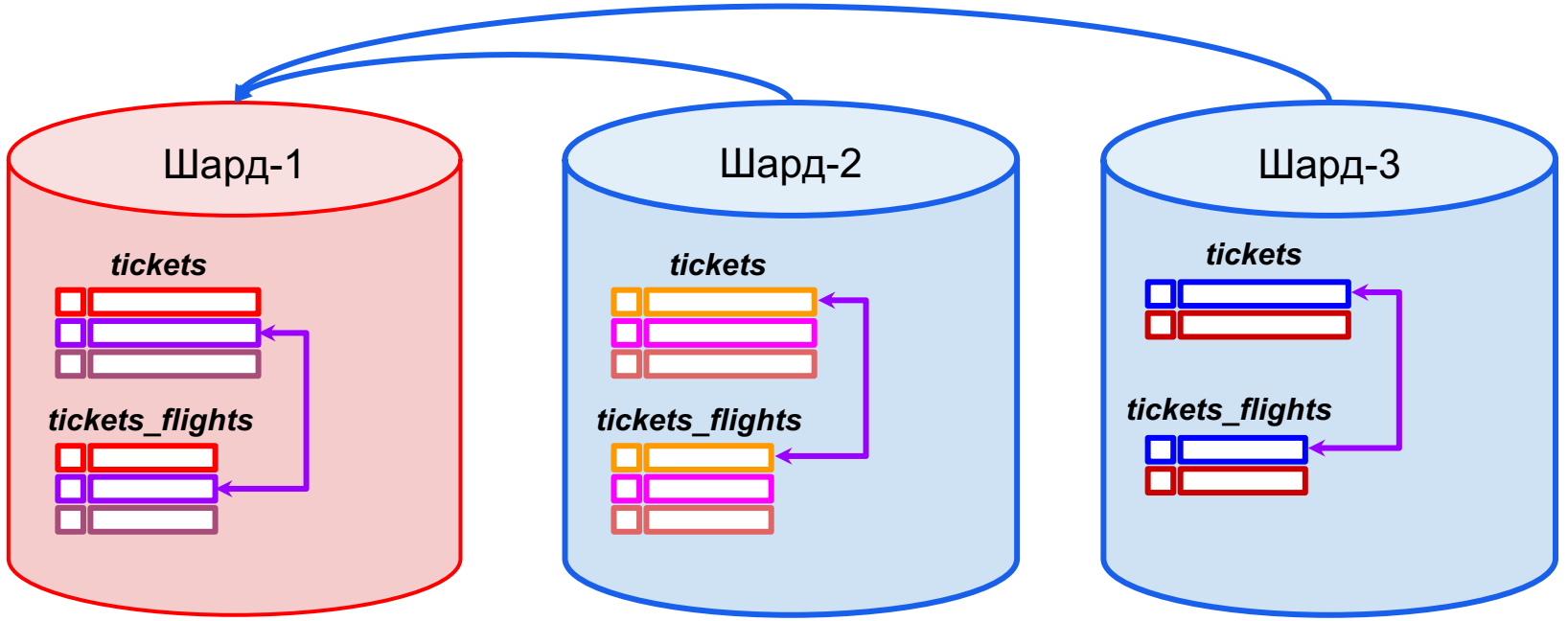
```
SELECT tf.*, t.*
FROM  tickets t
JOIN  ticket_flights tf
  ON  tf.ticket_no = t.ticket_no
WHERE t.ticket_no = '000543512';
```



Pushdown (проталкивание/спуск соединения)

- **JOIN**, должен быть указан ключ шардирования
- Агрегрегирования, с ограничениями
  - sum() – да
  - avg() – нет
- Подзапросы
- Асинхронное выполнение

```
SELECT tf.*, t.*
FROM  tickets t
JOIN  ticket_flights tf
  ON  tf.ticket_no = t.ticket_no;
```



Локальные и временные таблицы

- Локальные – **`CREATE TABLE …`**
  - Только на том шарде, где дали команду
  - Приёмник логической репликации
  - Временное хранение промежуточных данных
  - Можно создать одинаковые локальные таблицы на всех шардах
    - Содержимое — разное

- Временные – **`CREATE TEMPORARY TABLE …`**

Ограничения на Foreign Key

- **Допускаются FK**
  - Между глобальными таблицами
  - С шардированных таблиц на глобальные
  - Между шардированными колоцированными таблицами.

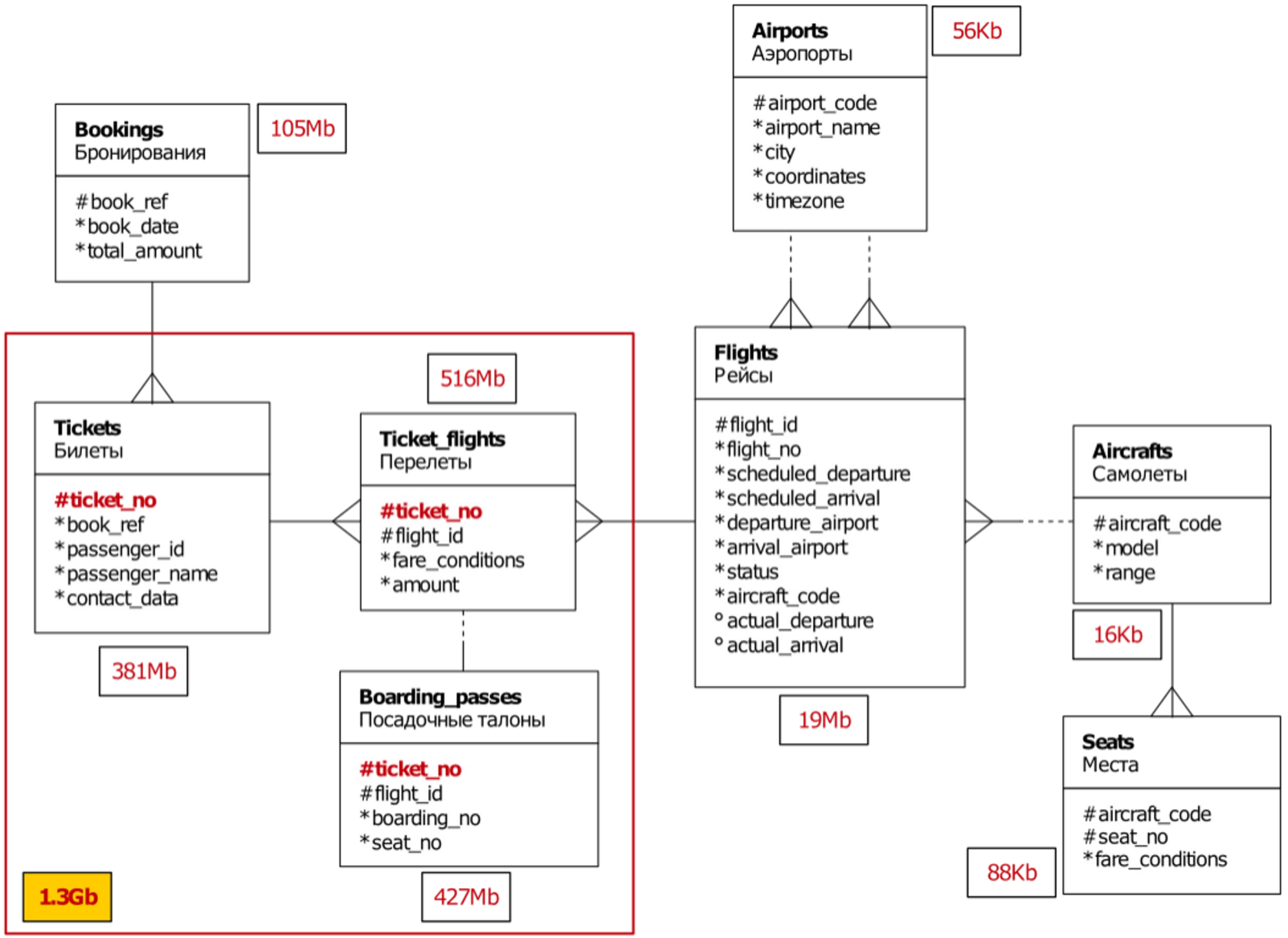
- **Не допускаются FK**
  - С глобальных на шардированные таблицы
  - Между шардированными таблицами,
    если они не колоцированы

Последовательности

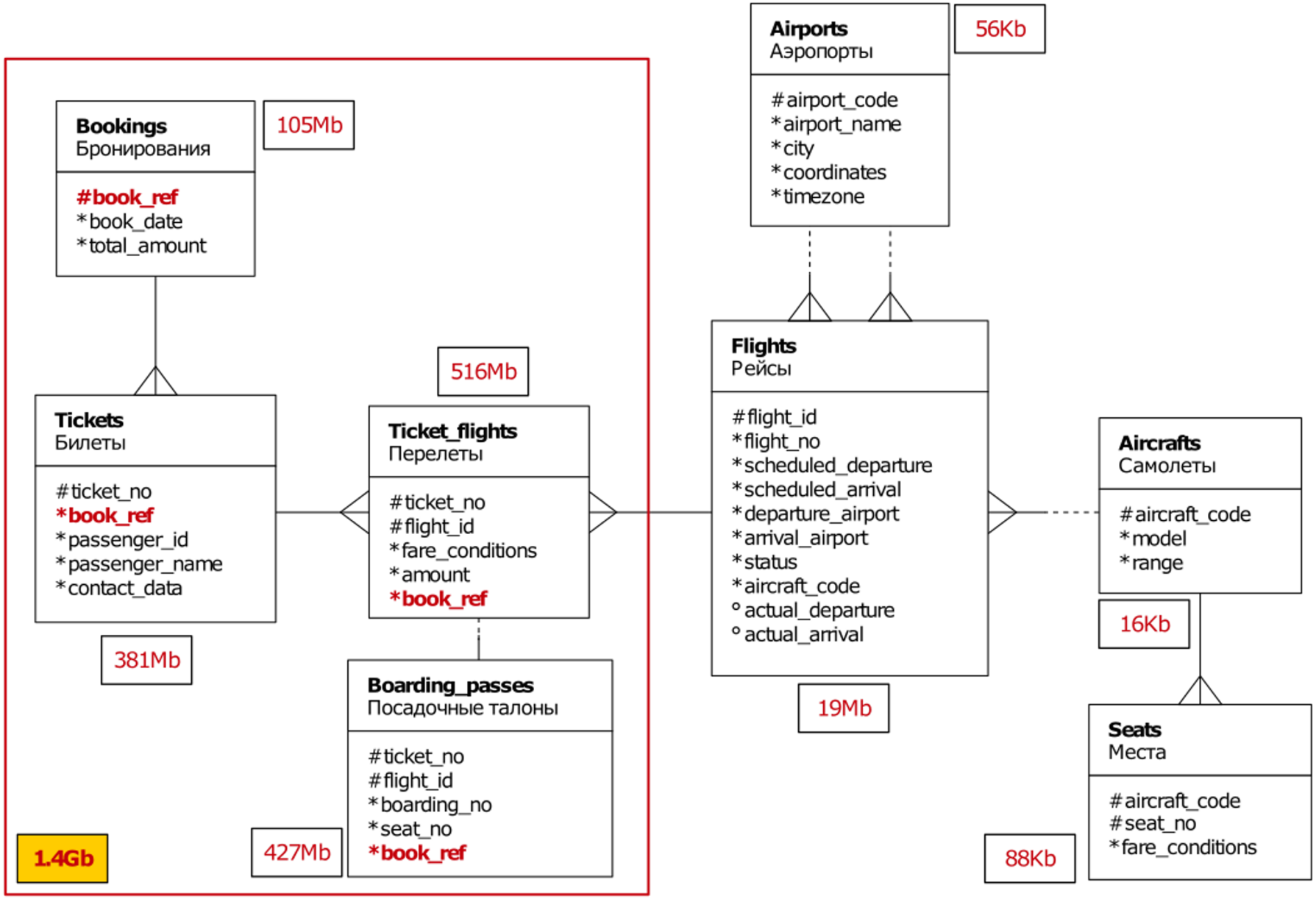
- `CREATE SEQUENCE … WITH (global)`
  - тип bigserial – отлично работает
  - под капотом – обычные сиквенсы на каждом шарде
  - выделяют последовательные блоки номеров (65536)
  - номера из глобального сиквенса уникальны
  - могут быть «дырки» и нет строгой монотонности

- `select * from shardman.sequences;`

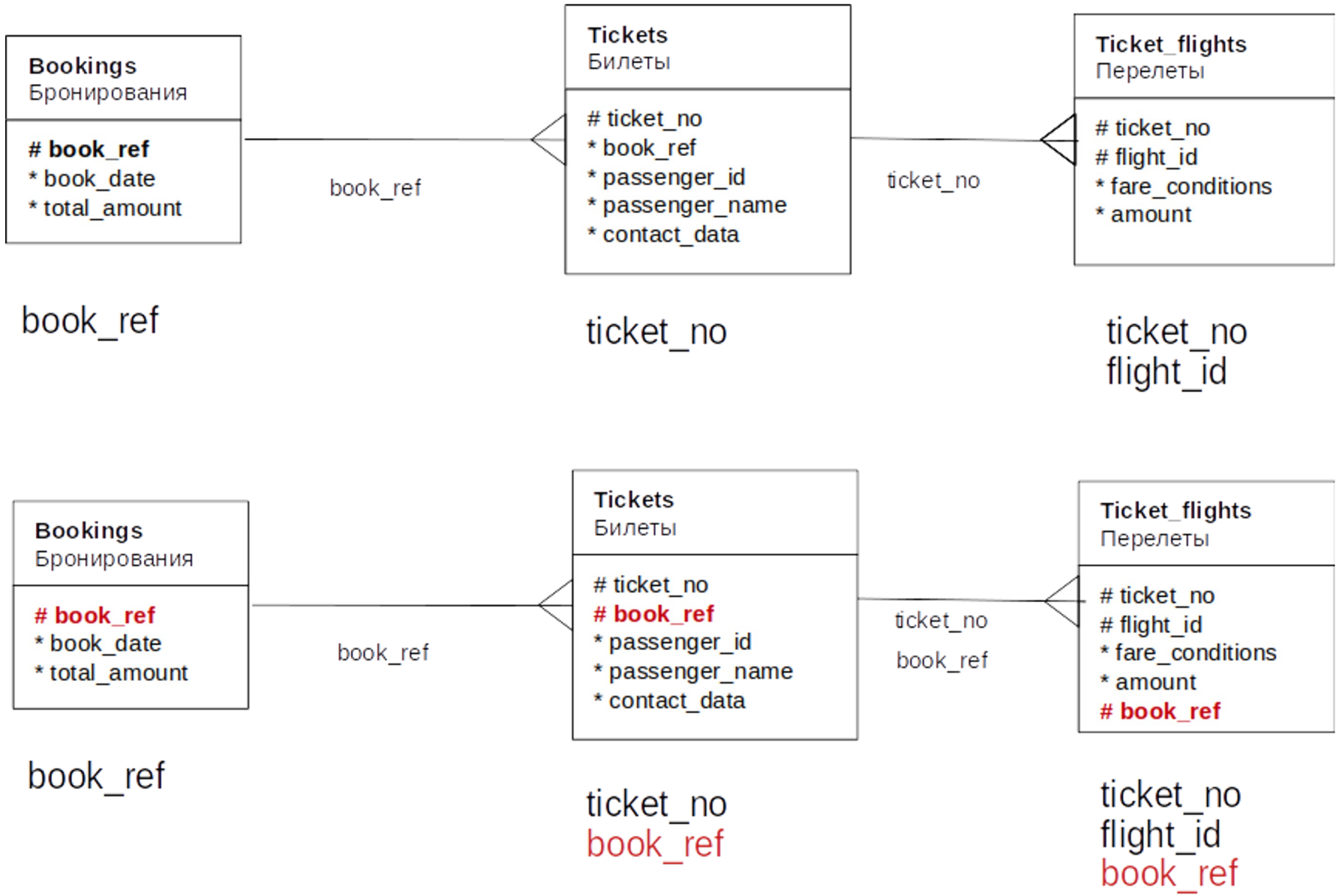
Простая миграция



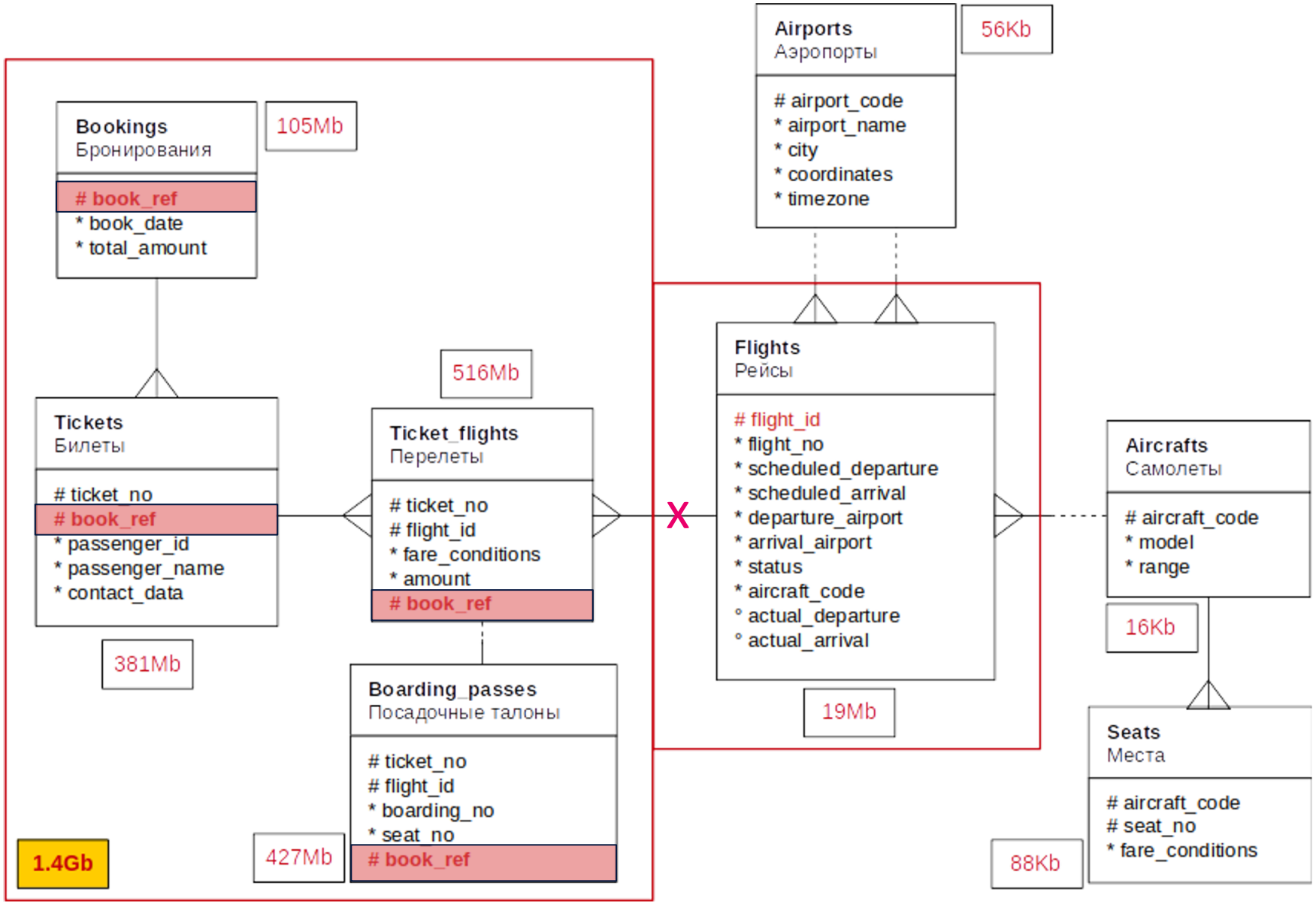
Миграция с трансформацией схемы



Проблема таблиц - «внучек»



Миграция с трансформацией схемы



Синхронизация объектов на шардах

- **Глобальные команды**
  - `create role <new_user>`
    `with login password 'pwd' in role global;`
  - `create table … with` (способ шардирования);
  - `create sequence … with (global);`
  - `grant\revoke` на глобальных объектах

Синхронизация объектов на шардах

PostgresPro

- Локальные команды
  - `create/alter schema, function, view,  trigger`
- `shardman.broadcast_all_sql(statement text)`
- Параметр `shardman.broadcast_ddl = on`
- Скрипт с идемпотентными командами
  - `create or replace` или `create if not exists`
  - запускать на каждом шарде
- Контролировать синхронность схем и кода на шардах!!!

Перенос данных в shardman

PostgresPro

Сложности с pg_dump/pg_restore

```
shardmanctl load --schema schema_description.yml
```

- из файла или из исходной БД
- создание таблиц, пользователей, перенос прав
- многопоточность
- передача данных сразу на нужный шард
- создание индексов

Итоги

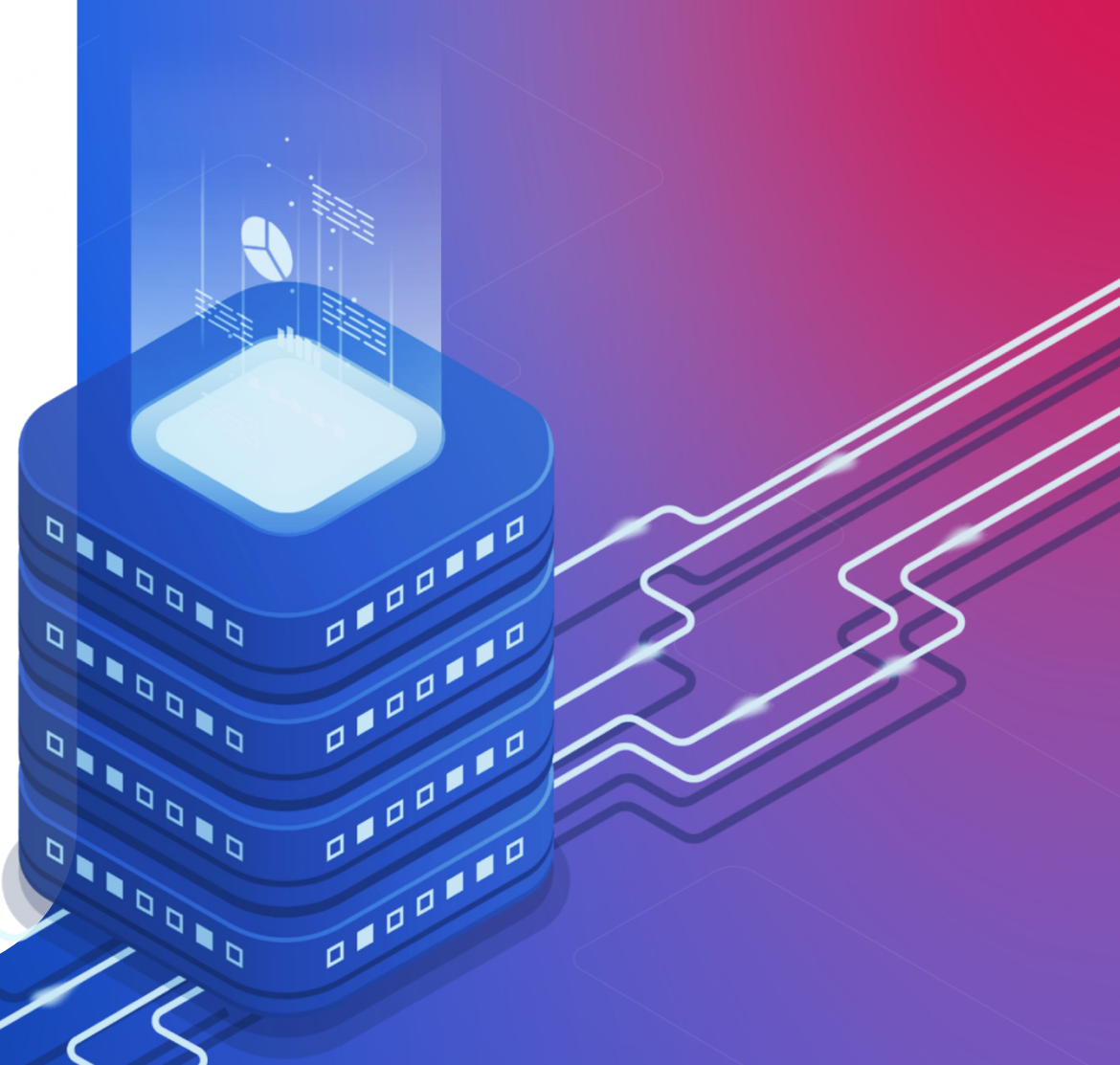
- Shardman – это про OLTP
- Как минимум – шардирование – это не скучно
- Как максимум – это больно
- Должна быть веская причина для этого
- Прозрачное шардирование – недостижимая мечта
- Придется адаптировать код приложения
- Придется изменять запросы
- Если есть много хранимок – то это проблема
- Придется мигрировать данные
- Много работы!

Документация доступна на сайте

- Документация
  https://postgrespro.ru/docs/shardman/14/index

PostgresPro

Спасибо за внимание!



PostgresPro

Q&A