PostgresPro

Шардирование больших баз данных

Забелин Андрей
a.zabelin@postgrespro.ru

27.05.2025

Shardman

PostgresPro

Распределенная СУБД Postgres Pro Enterprise 17

- Postgres Pro Enterprise
- Расширение Shardman
- Расширение postgres_fdw
- Расширение BiHA

Shardmanctl – утилита управление

Etcd – хранение состояния и конфигурации

BiHA – отказоустойчивость

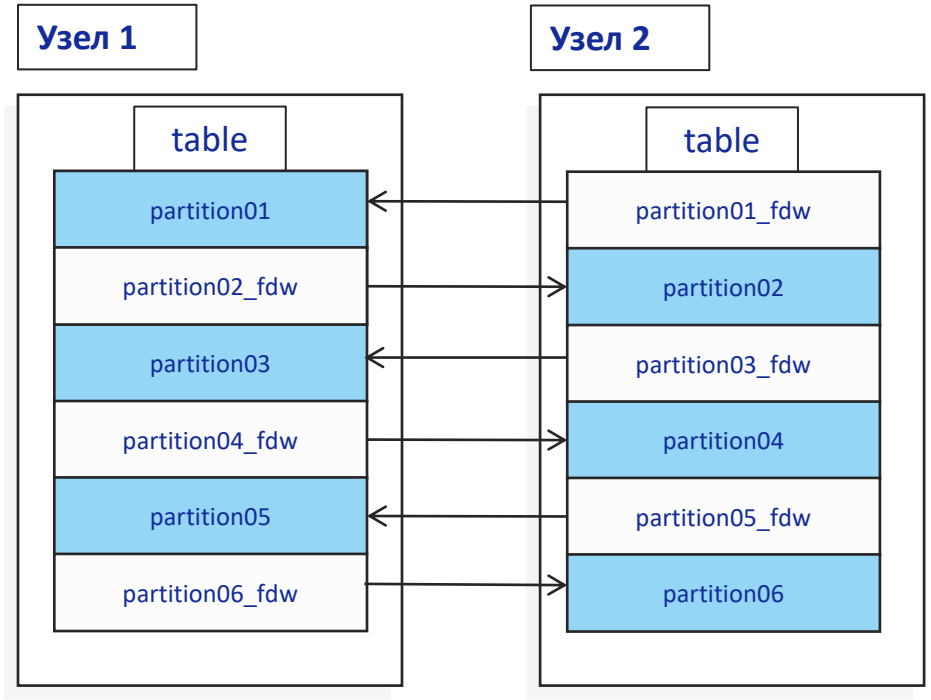
Shardman: преимущества

PostgresPro

- Прозрачное горизонтальное масштабирование
- OLTP нагрузка
- Доступ в кластер через любой узел
- Избыточность данных:
  встроенная отказоустойчивость
- Целостность данных:
  распределенные транзакции, строгие гарантии ACID

Shardman: распределённая БД

PostgresPro



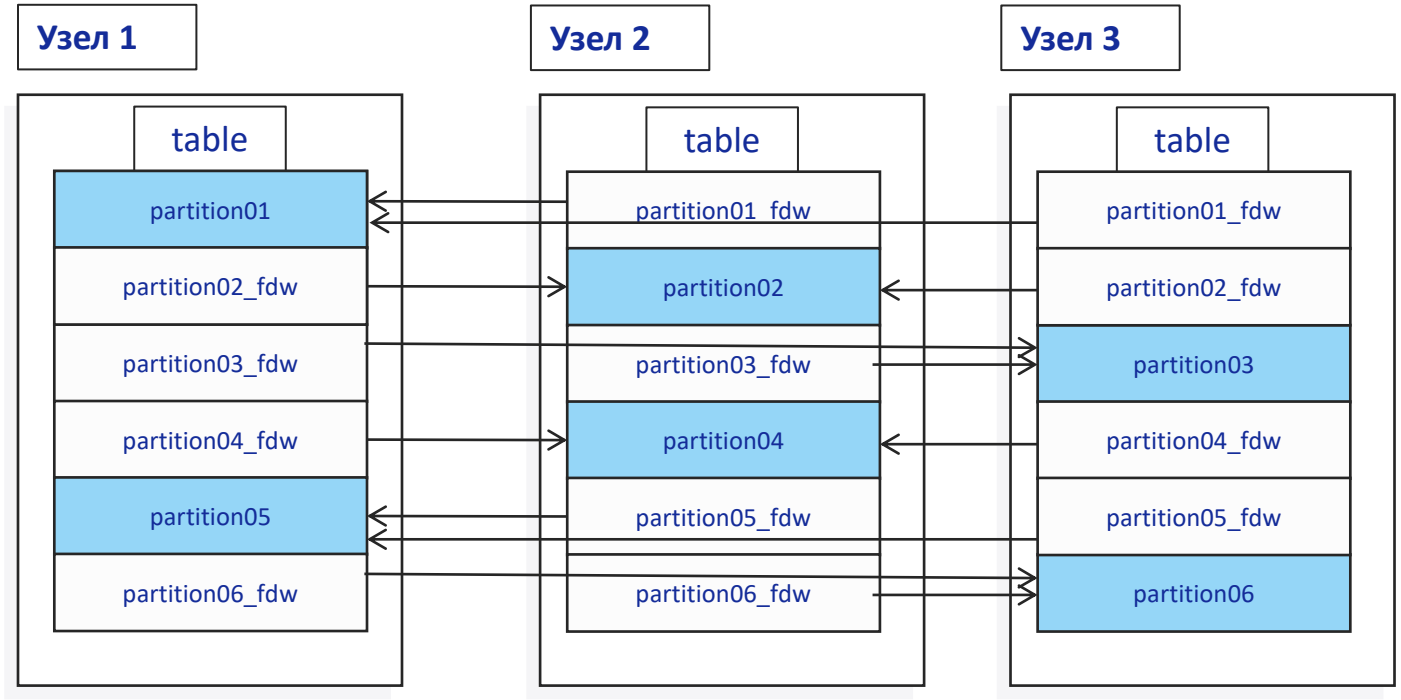
Каждый узел содержит только часть данных таблиц

Таблица состоит из набора секций, которые находятся физически на разных узлах кластера

Секции с других узлов доступны через fdw

Число секций фиксируется при создании таблицы

Shardman: масштабируемость БД

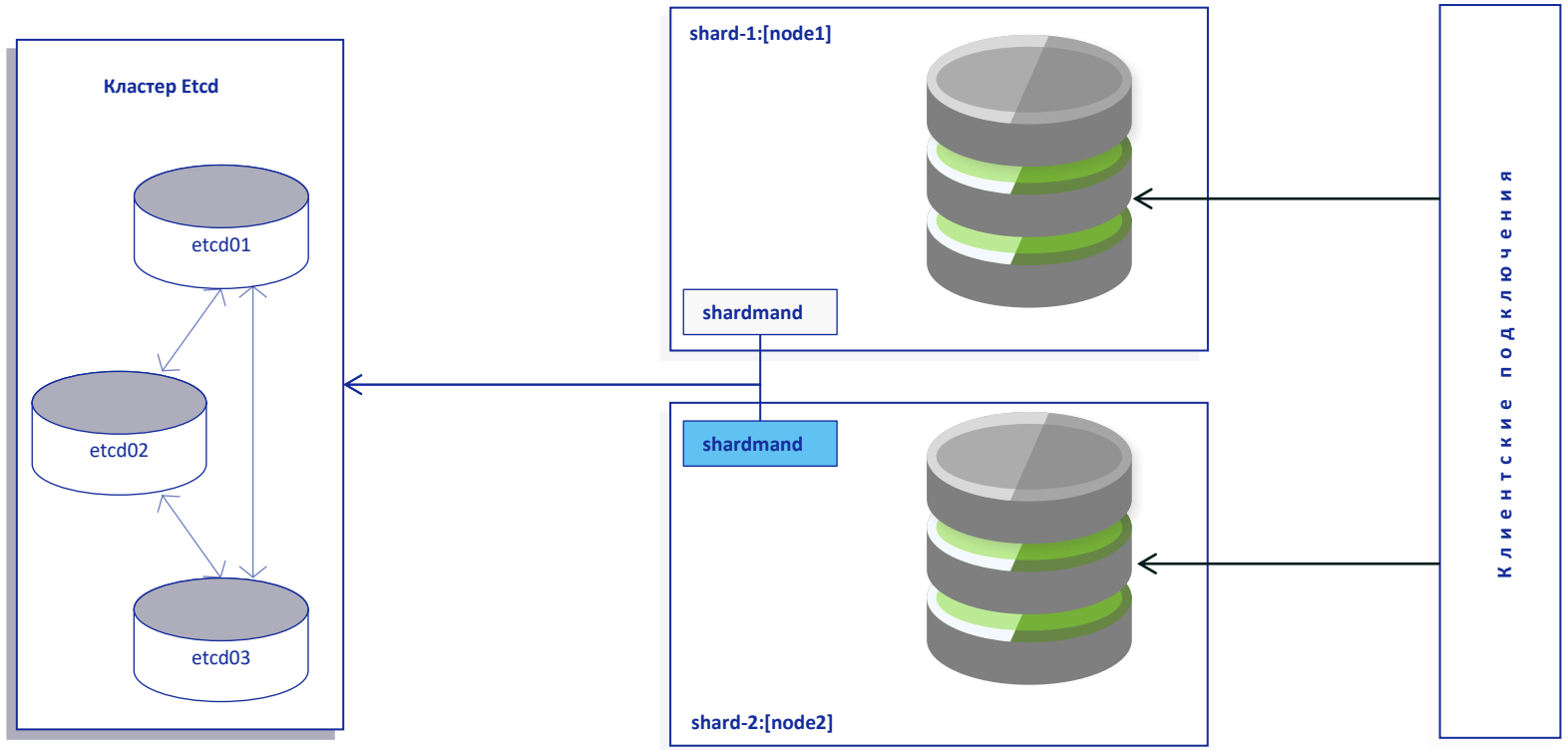


При добавлении узла часть партиций переезжает

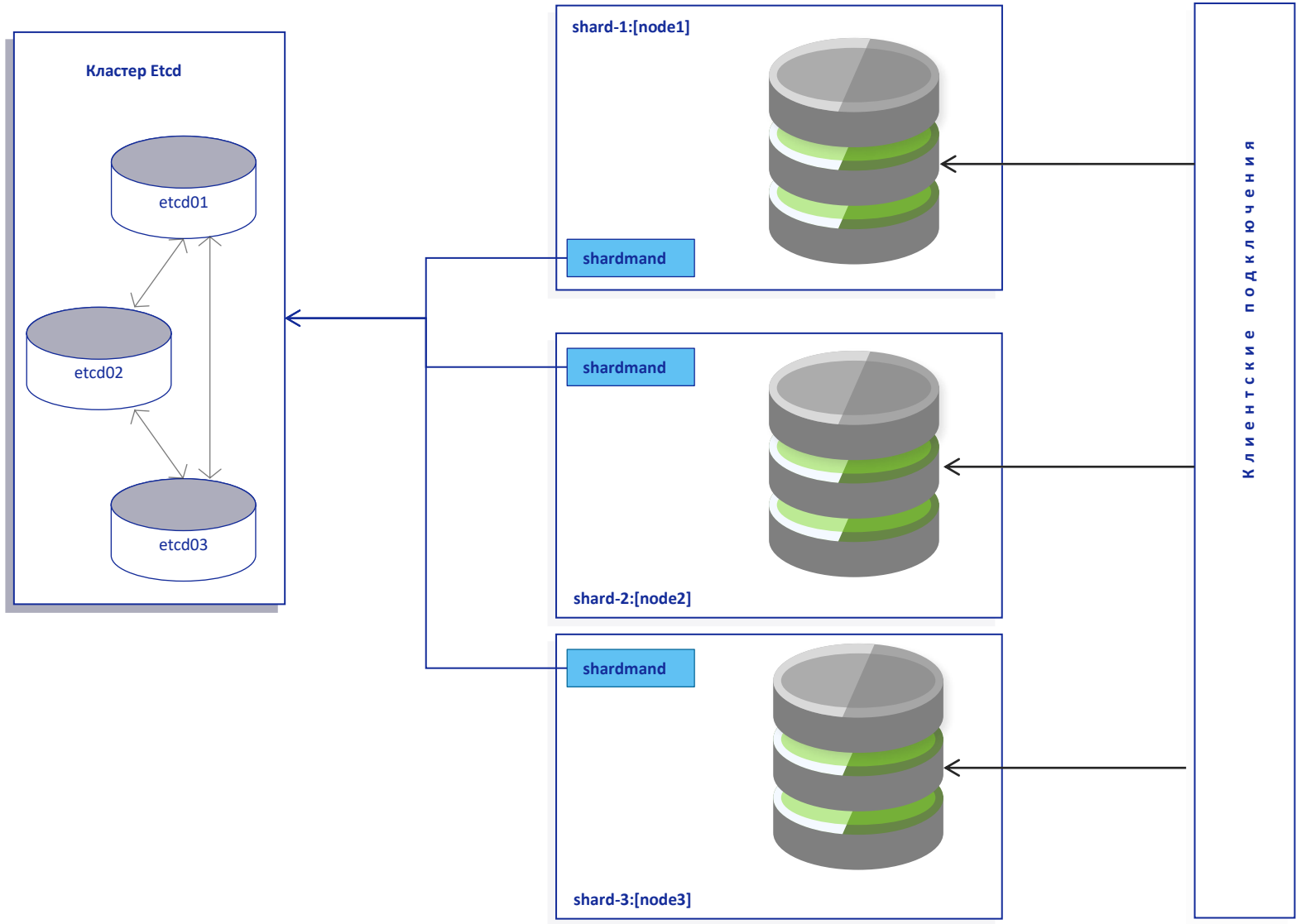
Каждый узел отвечает за обработку меньшего количества данных

Производительность линейно увеличивается

Архитектура Shardman



Масштабирование Shardman



PostgresPro

Shardman: виды таблиц

- Шардированная таблица — это секционированная таблица, где часть секций являются обычными локальными секциями таблицы, а остальные — внешними секциями таблицы через fdw.
  - Количество секций определяется при создании шардированной таблицы, оно должно быть больше или равно планируемого количества узлов в распределенной СУБД. Количество секций нельзя изменить.

- Несколько шардированных таблиц могут быть размещены совместно.
  - Секции совмещённых таблиц , соответствующие одному и тому же ключу шардирования, находятся на одном узле. Совместное размещение необходимо для того, чтобы операция соединения нескольких таблиц (JOIN) выполнялись на узле, где находятся фактические данные.

- Глобальные таблицы дублируются на все узлы и используются для редко обновляемых данных
  - Например наиболее подходящими для этого являются таблицы справочники. Когда выполняется соединение шардированной таблицы с глобальной таблицей, оно выполняется на узле, где находятся данные. При изменении данных в глобальной таблице, изменения данных одновременно происходят на всех узлах кластера.

PostgresPro

Shardman: схемы данных

Исходная схема данных:

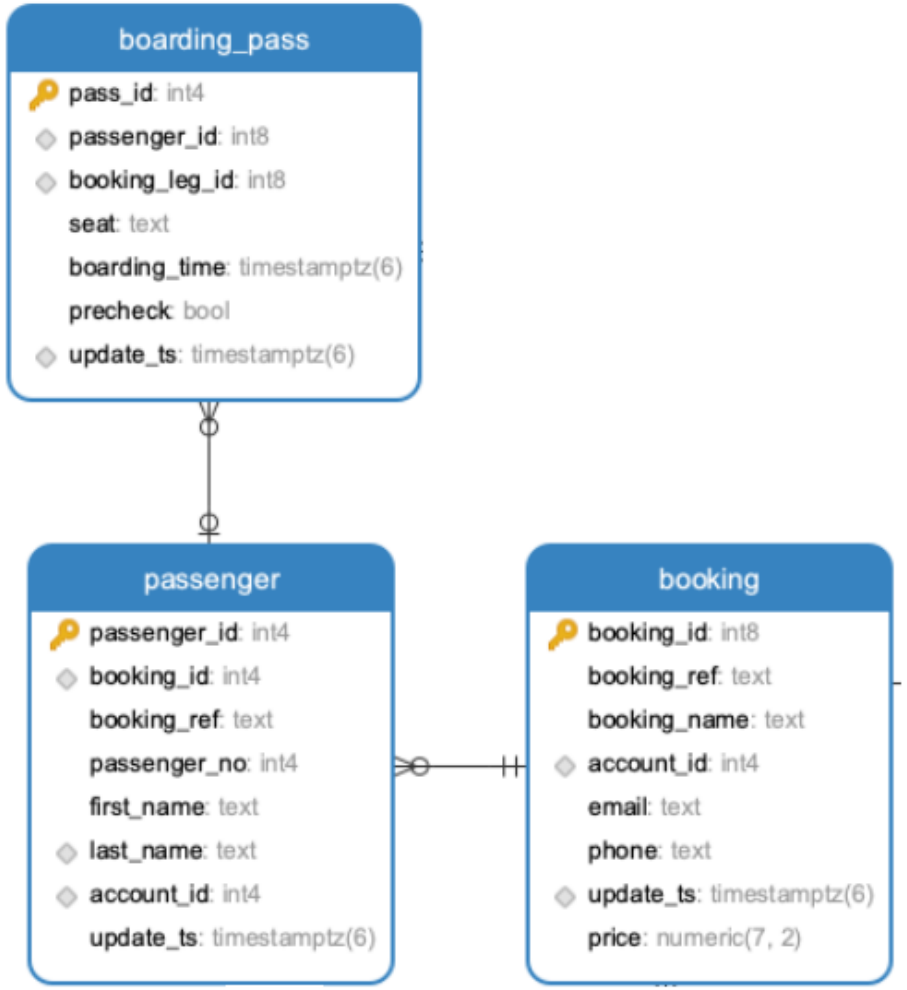


Схема данных для Shardman:

```
CREATE TABLE booking
        (booking_id bigint NOT NULL, … )
        with (distributed_by ='booking_id');
alter table booking add constraint booking_pkey
        primary key (booking_id);

CREATE TABLE passenger
        (passenger_id integer NOT NULL,
         booking_id bigint NOT NULL, …)
        with (distributed_by ='booking_id',
                colocate_with = 'booking');
alter table passenger add constraint passenger_pkey
        primary key (passenger_id, booking_id);

CREATE TABLE boarding_pass
        (pass_id integer NOT NULL,
         booking_id bigint NOT NULL, …)
        with (distributed_by ='booking_id',
                colocate_with = 'booking');
alter table boarding_pass  add constraint boarding_pass_pkey
        primary key (pass_id, booking_id);
```

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id and bp.booking_id = p.booking_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=0.99..1.00 rows=1 width=185) (actual time=3.854..3.855 rows=12 loops=1)
   Relations: ((booking_3_fdw b) INNER JOIN (passenger_3_fdw p)) INNER JOIN (boarding_pass_3_fdw bp)
   Foreign EXPLAIN:
     Nested Loop  (cost=1.29..270.02 rows=1 width=185)
       ->  Nested Loop  (cost=0.86..261.56 rows=1 width=99)
             ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                   Index Cond: (booking_id = 1111)
             ->  Index Scan using boarding_pass_3_passenger_id_idx on boarding_pass_3 r8  (cost=0.43..20.51 rows=1 width=48)
                   Index Cond: (passenger_id = r7.passenger_id)
                   Filter: (booking_id = 1111)
       ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
             Index Cond: (booking_id = 1111)
 Planning Time: 0.600 ms
 Execution Time: 3.887 ms
(14 rows)
```

Shardman: запрос с ключом шардирования

PostgresPro

Ключ шардирования

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id and bp.booking_id = p.booking_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=0.99..1.00 rows=1 width=185) (actual time=3.854..3.855 rows=12 loops=1)
   Relations: ((booking_3_fdw b) INNER JOIN (passenger_3_fdw p)) INNER JOIN (boarding_pass_3_fdw bp)
   Foreign EXPLAIN:
     Nested Loop  (cost=1.29..270.02 rows=1 width=185)
       ->  Nested Loop  (cost=0.86..261.56 rows=1 width=99)
             ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                   Index Cond: (booking_id = 1111)
             ->  Index Scan using boarding_pass_3_passenger_id_idx on boarding_pass_3 r8  (cost=0.43..20.51 rows=1 width=48)
                   Index Cond: (passenger_id = r7.passenger_id)
                   Filter: (booking_id = 1111)
       ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
             Index Cond: (booking_id = 1111)
 Planning Time: 0.600 ms
 Execution Time: 3.887 ms
(14 rows)
```

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id and bp.booking_id = p.booking_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Foreign Scan  (cost=0.99..1.00 rows=1 width=185) (actual time=3.854..3.855 rows=12 loops=1)
   Relations: ((booking_3_fdw b) INNER JOIN (passenger_3_fdw p)) INNER JOIN (boarding_pass_3_fdw bp)
   Foreign EXPLAIN:
     Nested Loop  (cost=1.29..270.02 rows=1 width=185)
       ->  Nested Loop  (cost=0.86..261.56 rows=1 width=99)
             ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                   Index Cond: (booking_id = 1111)
             ->  Index Scan using boarding_pass_3_passenger_id_idx on boarding_pass_3 r8  (cost=0.43..20.51 rows=1 width=48)
                   Index Cond: (passenger_id = r7.passenger_id)
                   Filter: (booking_id = 1111)
       ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
             Index Cond: (booking_id = 1111)
 Planning Time: 0.600 ms
 Execution Time: 3.887 ms
(14 rows)
```

Ключ шардирования

Выполнение JOIN на другом шарде

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id and bp.booking_id = p.booking_id
where b.booking_id = 1114;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.29..292.87 rows=1 width=185) (actual time=1.183..3.233 rows=12 loops=1)
   ->  Nested Loop  (cost=0.86..284.42 rows=1 width=99) (actual time=1.169..3.178 rows=12 loops=1)
         ->  Index Scan using passenger_0_booking_id_idx on passenger_0 p  (cost=0.43..38.15 rows=12 width=51) (actual time=0.704..0.805 rows=3 loops=1)
               Index Cond: (booking_id = 1114)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp  (cost=0.43..20.51 rows=1 width=48) (actual time=0.328..0.788 rows=4 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
               Filter: (booking_id = 1114)
   ->  Index Scan using booking_0_pkey on booking_0 b  (cost=0.43..8.45 rows=1 width=86) (actual time=0.003..0.004 rows=1 loops=12)
         Index Cond: (booking_id = 1114)
 Planning Time: 0.441 ms
 Execution Time: 3.262 ms
(11 rows)
```

Ключ шардирования

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id and bp.booking_id = p.booking_id
where b.booking_id = 1114;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.29..292.87 rows=1 width=185) (actual time=1.183..3.233 rows=12 loops=1)
   ->  Nested Loop  (cost=0.86..284.42 rows=1 width=99) (actual time=1.169..3.178 rows=12 loops=1)
         ->  Index Scan using passenger_0_booking_id_idx on passenger_0 p  (cost=0.43..38.15 rows=12 width=51) (actual time=0.704..0.805 rows=3 loops=1)
               Index Cond: (booking_id = 1114)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp  (cost=0.43..20.51 rows=1 width=48) (actual time=0.328..0.788 rows=4 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
               Filter: (booking_id = 1114)
   ->  Index Scan using booking_0_pkey on booking_0 b  (cost=0.43..8.45 rows=1 width=86) (actual time=0.003..0.004 rows=1 loops=12)
         Index Cond: (booking_id = 1114)
 Planning Time: 0.441 ms
 Execution Time: 3.262 ms
(11 rows)
```

Ключ шардирования

Выполнение на локальном шарде

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1114;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.29..4821.72 rows=1815972 width=185) (actual time=1.821..3.743 rows=12 loops=1)
   ->  Nested Loop  (cost=0.86..46.71 rows=12 width=137) (actual time=1.225..1.418 rows=3 loops=1)
         ->  Index Scan using booking_0_pkey on booking_0 b  (cost=0.43..8.45 rows=1 width=86) (actual time=0.728..0.729 rows=1 loops=1)
               Index Cond: (booking_id = 1114)
         ->  Index Scan using passenger_0_booking_id_idx on passenger_0 p  (cost=0.43..38.15 rows=12 width=51) (actual time=0.491..0.682 rows=3 loops=1)
               Index Cond: (booking_id = 1114)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=0.346..0.770 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.037..0.073 rows=4 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.190..0.190 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.060..0.060 rows=0 loops=3)
 Planning Time: 2.493 ms
 Execution Time: 3.945 ms
(17 rows)
```

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1114;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.29..4821.72 rows=1815972 width=185) (actual time=1.821..3.743 rows=12 loops=1)
   ->  Nested Loop  (cost=0.86..46.71 rows=12 width=137) (actual time=1.225..1.418 rows=3 loops=1)
         ->  Index Scan using booking_0_pkey on booking_0 b  (cost=0.43..8.45 rows=1 width=86) (actual time=0.728..0.729 rows=1 loops=1)
               Index Cond: (booking_id = 1114)
         ->  Index Scan using passenger_0_booking_id_idx on passenger_0 p  (cost=0.43..38.15 rows=12 width=51) (actual time=0.491..0.682 rows=3 loops=1)
               Index Cond: (booking_id = 1114)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=0.346..0.770 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.037..0.073 rows=4 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.190..0.190 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.060..0.060 rows=0 loops=3)
 Planning Time: 2.493 ms
 Execution Time: 3.945 ms
(17 rows)
```

Ключ шардирования

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1114;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.29..4821.72 rows=1815972 width=185) (actual time=1.821..3.743 rows=12 loops=1)
   ->  Nested Loop  (cost=0.86..46.71 rows=12 width=137) (actual time=1.225..1.418 rows=3 loops=1)
         ->  Index Scan using booking_0_pkey on booking_0 b  (cost=0.43..8.45 rows=1 width=86) (actual time=0.728..0.729 rows=1 loops=1)
               Index Cond: (booking_id = 1114)
         ->  Index Scan using passenger_0_booking_id_idx on passenger_0 p  (cost=0.43..38.15 rows=12 width=51) (actual time=0.491..0.682 rows=3 loops=1)
               Index Cond: (booking_id = 1114)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=0.346..0.770 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.037..0.073 rows=4 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.190..0.190 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.060..0.060 rows=0 loops=3)
 Planning Time: 2.493 ms
 Execution Time: 3.945 ms
(17 rows)
```

Ключ шардирования

Выполнение на локальном шарде

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1114;

QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.29..4821.72 rows=1815972 width=185) (actual time=1.821..3.743 rows=12 loops=1)
   ->  Nested Loop  (cost=0.86..46.71 rows=12 width=137) (actual time=1.225..1.418 rows=3 loops=1)
         ->  Index Scan using booking_0_pkey on booking_0 b  (cost=0.43..8.45 rows=1 width=86) (actual time=0.728..0.729 rows=1 loops=1)
               Index Cond: (booking_id = 1114)
         ->  Index Scan using passenger_0_booking_id_idx on passenger_0 p  (cost=0.43..38.15 rows=12 width=51) (actual time=0.491..0.682 rows=3 loops=1)
               Index Cond: (booking_id = 1114)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=0.346..0.770 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.037..0.073 rows=4 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.190..0.190 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.060..0.060 rows=0 loops=3)
 Planning Time: 2.493 ms
 Execution Time: 3.945 ms
(17 rows)
```

Ключ шардирования

Выполнение на локальном шарде

Выполнение на других шардах

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.42..4378.09 rows=1664641 width=185) (actual time=5.079..7.489 rows=12 loops=1)
   ->  Foreign Scan  (cost=0.99..1.00 rows=11 width=137) (actual time=4.117..4.118 rows=3 loops=1)
         Relations: (booking_3_fdw b) INNER JOIN (passenger_3_fdw p)
         Foreign EXPLAIN:
           Sort  (cost=44.56..44.59 rows=11 width=137)
             Sort Key: r7.passenger_id
             ->  Nested Loop  (cost=0.86..44.37 rows=11 width=137)
                   ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
                         Index Cond: (booking_id = 1111)
                   ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                         Index Cond: (booking_id = 1111)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=1.045..1.117 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.139..0.139 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.093..0.093 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.074..0.074 rows=4 loops=3)
 Planning Time: 3.933 ms
 Execution Time: 7.743 ms
(22 rows)
```

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.42..4378.09 rows=1664641 width=185) (actual time=5.079..7.489 rows=12 loops=1)
   ->  Foreign Scan  (cost=0.99..1.00 rows=11 width=137) (actual time=4.117..4.118 rows=3 loops=1)
         Relations: (booking_3_fdw b) INNER JOIN (passenger_3_fdw p)
         Foreign EXPLAIN:
           Sort  (cost=44.56..44.59 rows=11 width=137)
             Sort Key: r7.passenger_id
             ->  Nested Loop  (cost=0.86..44.37 rows=11 width=137)
                   ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
                         Index Cond: (booking_id = 1111)
                   ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                         Index Cond: (booking_id = 1111)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=1.045..1.117 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.139..0.139 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.093..0.093 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.074..0.074 rows=4 loops=3)
 Planning Time: 3.933 ms
 Execution Time: 7.743 ms
(22 rows)
```

Ключ шардирования

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.42..4378.09 rows=1664641 width=185) (actual time=5.079..7.489 rows=12 loops=1)
   ->  Foreign Scan  (cost=0.99..1.00 rows=11 width=137) (actual time=4.117..4.118 rows=3 loops=1)
         Relations: (booking_3_fdw b) INNER JOIN (passenger_3_fdw p)
         Foreign EXPLAIN:
           Sort  (cost=44.56..44.59 rows=11 width=137)
             Sort Key: r7.passenger_id
             ->  Nested Loop  (cost=0.86..44.37 rows=11 width=137)
                   ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
                         Index Cond: (booking_id = 1111)
                   ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                         Index Cond: (booking_id = 1111)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=1.045..1.117 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.139..0.139 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.093..0.093 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.074..0.074 rows=4 loops=3)
 Planning Time: 3.933 ms
 Execution Time: 7.743 ms
(22 rows)
```

Ключ шардирования

Выполнение JOIN на другом шарде

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.42..4378.09 rows=1664641 width=185) (actual time=5.079..7.489 rows=12 loops=1)
   ->  Foreign Scan  (cost=0.99..1.00 rows=11 width=137) (actual time=4.117..4.118 rows=3 loops=1)
         Relations: (booking_3_fdw b) INNER JOIN (passenger_3_fdw p)
         Foreign EXPLAIN:
           Sort  (cost=44.56..44.59 rows=11 width=137)
             Sort Key: r7.passenger_id
             ->  Nested Loop  (cost=0.86..44.37 rows=11 width=137)
                   ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
                         Index Cond: (booking_id = 1111)
                   ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                         Index Cond: (booking_id = 1111)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=1.045..1.117 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.139..0.139 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.093..0.093 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.074..0.074 rows=4 loops=3)
 Planning Time: 3.933 ms
 Execution Time: 7.743 ms
(22 rows)
```

Ключ шардирования

Выполнение JOIN на другом шарде

Выполнение на других шардах

Shardman: запрос с ключом шардирования

PostgresPro

```
postgres=# explain(analyse)
SELECT *
FROM booking b
join passenger p on p.booking_id = b.booking_id
join boarding_pass bp on bp.passenger_id = p.passenger_id
where b.booking_id = 1111;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------
 Nested Loop  (cost=1.42..4378.09 rows=1664641 width=185) (actual time=5.079..7.489 rows=12 loops=1)
   ->  Foreign Scan  (cost=0.99..1.00 rows=11 width=137) (actual time=4.117..4.118 rows=3 loops=1)
         Relations: (booking_3_fdw b) INNER JOIN (passenger_3_fdw p)
         Foreign EXPLAIN:
           Sort  (cost=44.56..44.59 rows=11 width=137)
             Sort Key: r7.passenger_id
             ->  Nested Loop  (cost=0.86..44.37 rows=11 width=137)
                   ->  Index Scan using booking_3_pkey on booking_3 r6  (cost=0.43..8.45 rows=1 width=86)
                         Index Cond: (booking_id = 1111)
                   ->  Index Scan using passenger_3_booking_id_idx on passenger_3 r7  (cost=0.43..35.81 rows=11 width=51)
                         Index Cond: (booking_id = 1111)
   ->  Append  (cost=0.43..397.76 rows=16 width=48) (actual time=1.045..1.117 rows=4 loops=3)
         ->  Index Scan using boarding_pass_0_passenger_id_idx on boarding_pass_0 bp_1  (cost=0.43..20.50 rows=4 width=48) (actual time=0.072..0.072 rows=0 loops=3)
               Index Cond: (passenger_id = p.passenger_id)
         ->  Async Foreign Scan on boarding_pass_1_fdw bp_2  (cost=104.46..125.73 rows=4 width=48) (actual time=0.139..0.139 rows=0 loops=3)
               Foreign EXPLAIN:
                 Index Scan using boarding_pass_1_passenger_id_idx on boarding_pass_1  (cost=0.43..20.50 rows=4 width=48)
                   Index Cond: (passenger_id = $1)
         ->  Async Foreign Scan on boarding_pass_2_fdw bp_3  (cost=104.46..125.73 rows=4 width=48) (actual time=0.093..0.093 rows=0 loops=3)
         ->  Async Foreign Scan on boarding_pass_3_fdw bp_4  (cost=104.46..125.73 rows=4 width=48) (actual time=0.074..0.074 rows=4 loops=3)
 Planning Time: 3.933 ms
 Execution Time: 7.743 ms
(22 rows)
```

Ключ шардирования

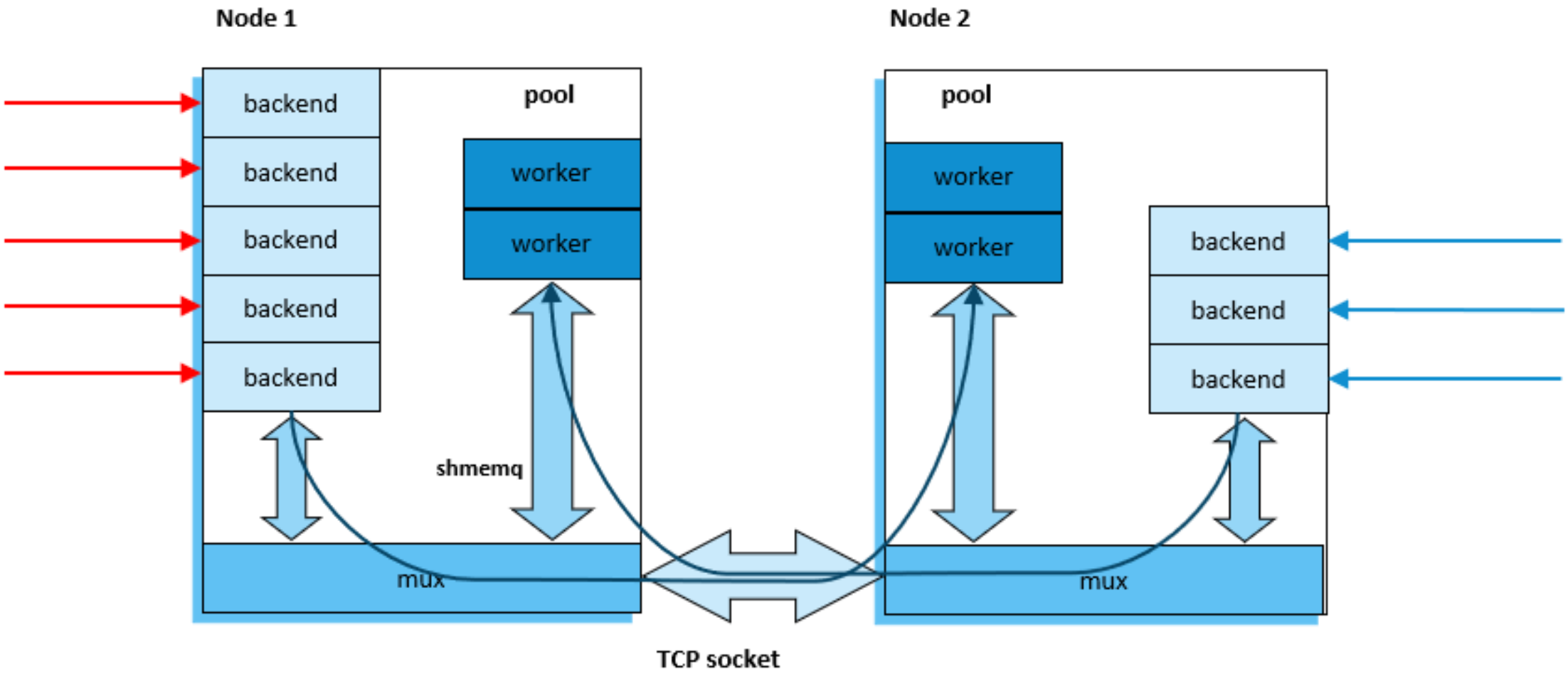
Выполнение JOIN на другом шарде

Выполнение на локальном шарде

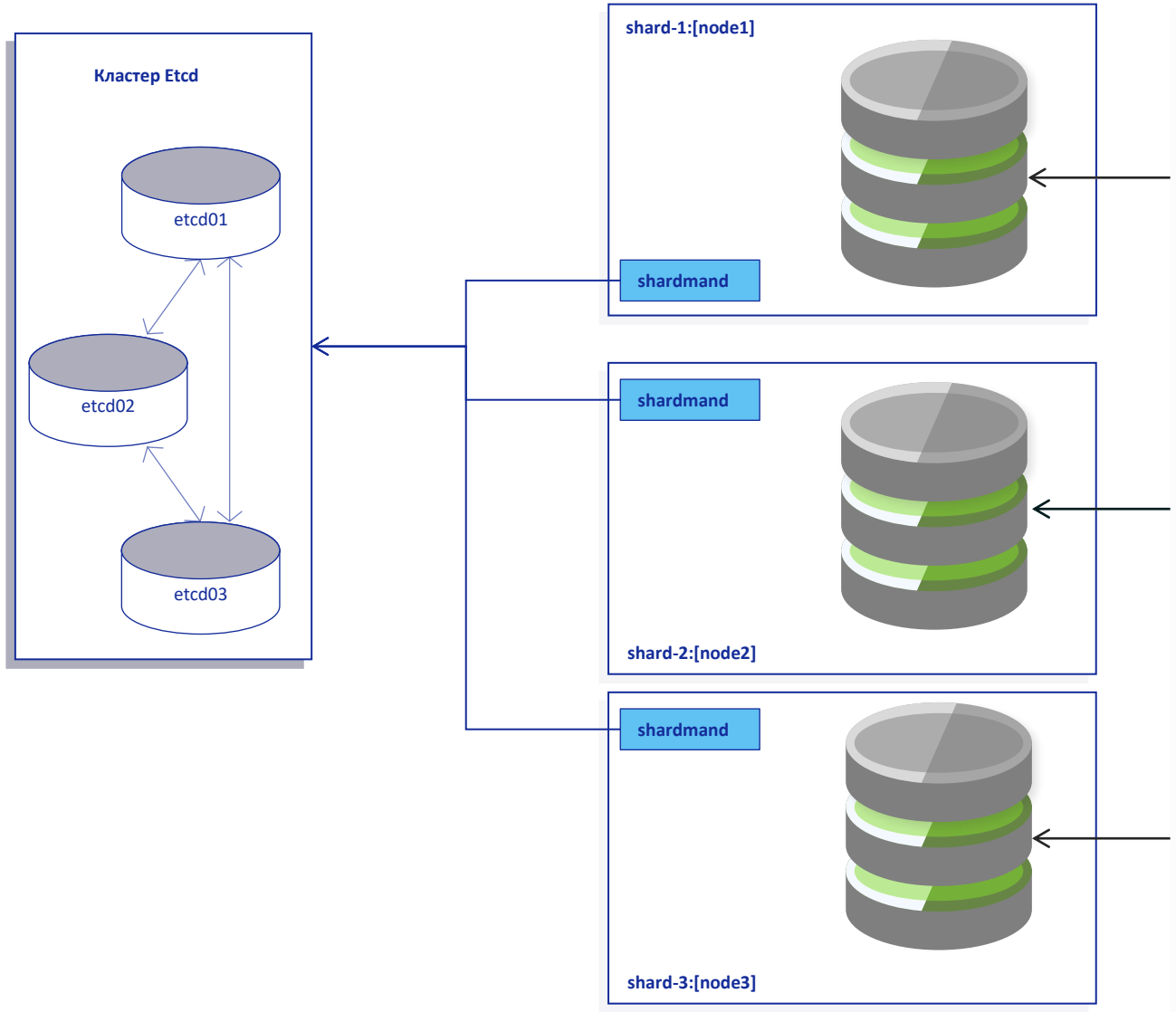
Выполнение на других шардах

Shardman: мультиплексирование fdw

PostgresPro



Загрузка данных в Shardman



shardmanctl load --schema load_schema.yaml

В **yaml** файле определён источник данных (CSV или другая БД) и некоторые параметры работы утилиты:
migrate.jobs количество параллельных процессов загрузки
migrate.batch задаёт количество строк в одной порции

migrate.schema определяет список схем
migrate.schemas.tables : список таблиц в схеме
два типа таблиц: global и sharded.

Для шардированной таблицы нужно указать:
distributedby - имя столбца, используемого для секционирования таблицы
partitions - количество секций, которые будут созданы для этой таблицы

Сначала загружаются глобальные (global) таблицы, затем шардированные (sharded) таблицы и в конце шардированные таблицы с параметром colocatedwith. Порядок загрузки таблиц одного типа можно изменить с помощью параметра priority

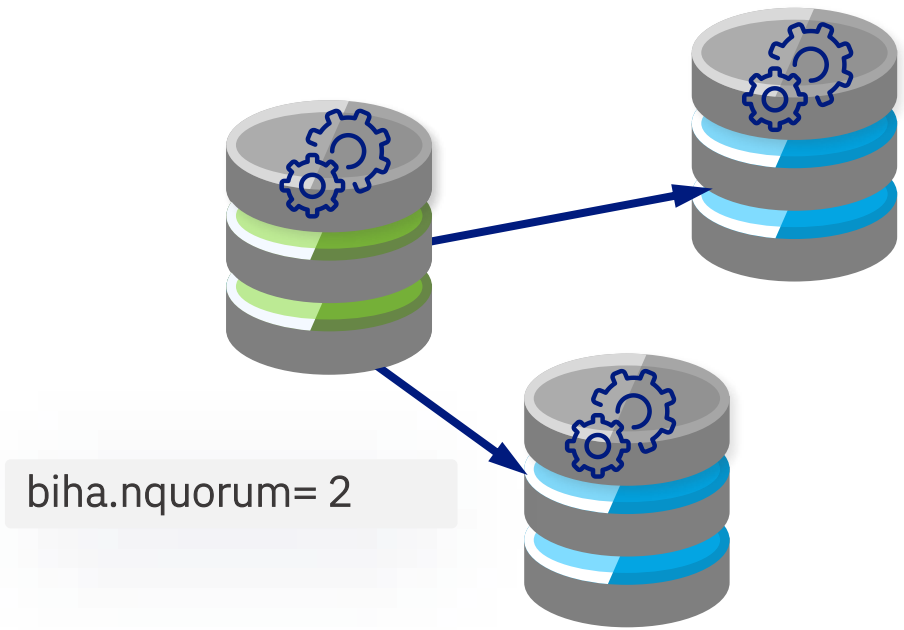
shardmanctl load умеет распределять данные при миграции данных сразу в тот шард, где они должны храниться

Shardman: возможности Postgres Pro Enterprise 17

PostgresPro

- механизм сжатия данных CFS,
- pg_probackup : инкрементальный бэкап на уровне страниц, используя PTRACK,
- расширение pgpro_stats : сбор статистики планирования и выполнения распределённых запросов, затрагивающих несколько узлов в кластере
- Встроенный отказоустойчивый кластер BiHA

BiHA: встроенный отказоустойчивый кластер



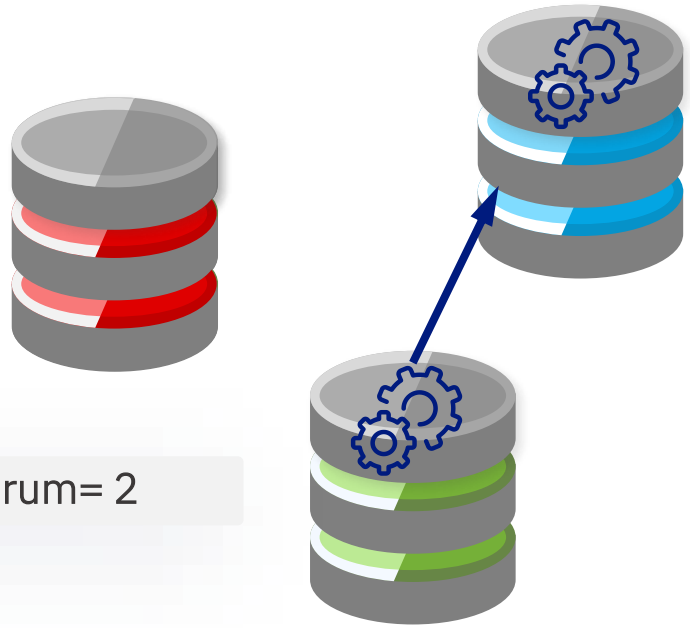
Встроен в ядро Postgres Pro Enterprise начиная с версии 16

Простая установка и конфигурирование

Не требуется установка дополнительного ПО

Лидер продолжает работать, если соблюдается кворум

BiHA: встроенный отказоустойчивый кластер



biha.nquorum= 2

Встроен в ядро Postgres Pro Enterprise начиная с версии 16

Простая установка и конфигурирование

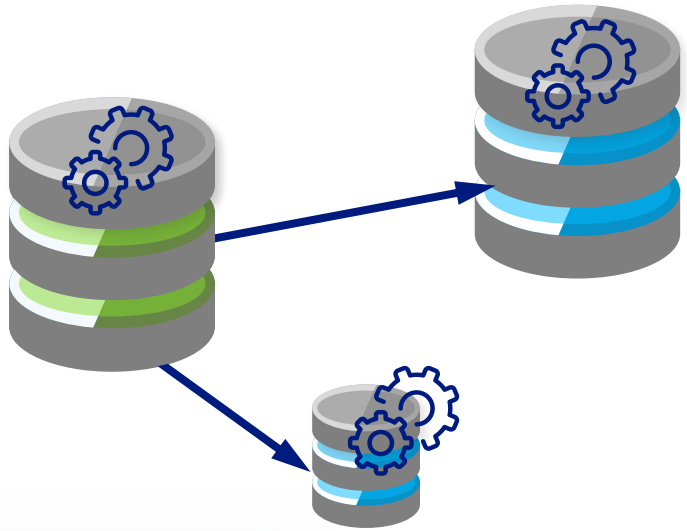
Не требуется установка дополнительного ПО

Лидер продолжает работать, если соблюдается кворум

В случае сбоя ведомые предлагают себя кандидатами и организуются выборы нового лидера

Защита от split-brain

BiHA: узел рефери



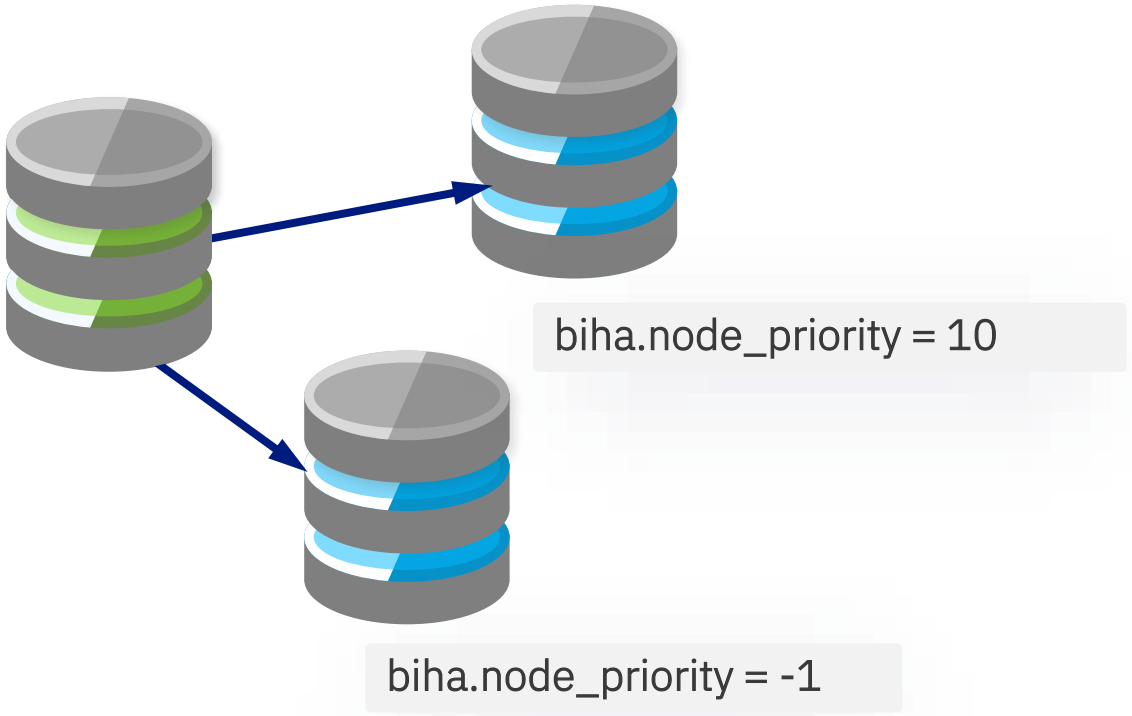
mode=
referee или
referee_with_wal

Рефери – легковесный экземпляр, который не содержит пользовательских данных, но является членом кластера BiHA:
сам кандидатом на звание нового лидера
не выступает, но участвует в голосовании

Рефери может работать в режиме referee_with_wal :

- Получает весь WAL и фильтрует его, применяются только системные записи WAL без пользовательских данных
- При сбое лидера может отправлять WAL кандидату на нового лидера,
если тот отстаёт от рефери

BiHA: приоритеты кандидатов в лидеры

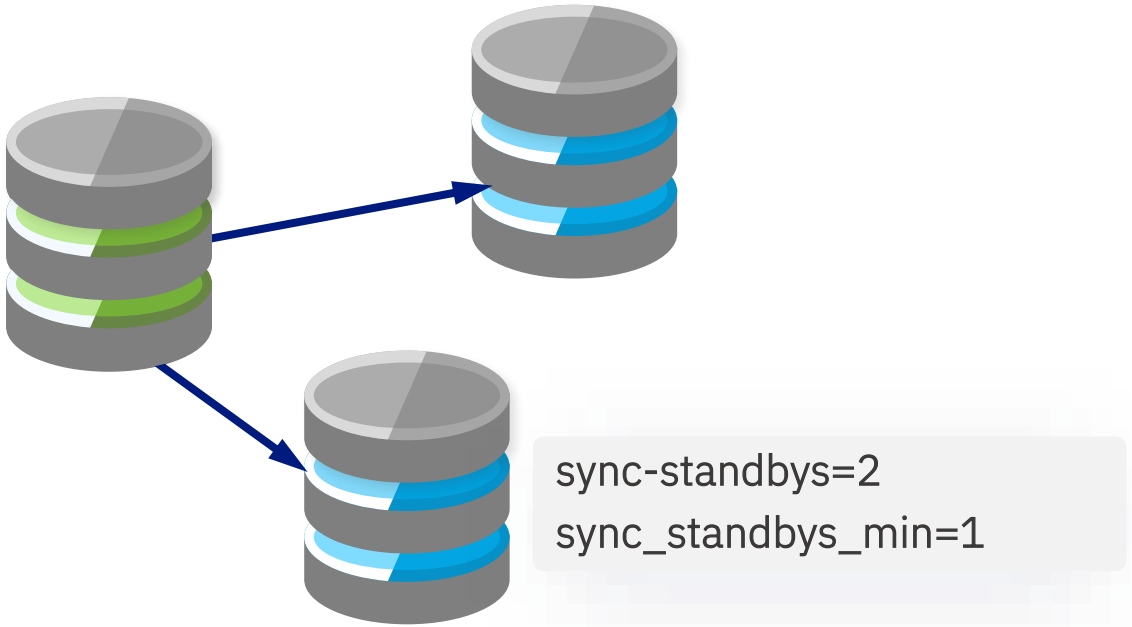


Можно установить приоритеты для выдвижения узла в качестве кандидата

Приоритет узла в секундах определяет тайм-аут, по достижении которого узел предложит себя в качестве кандидата на выборах.

Только в синхронном кластере

BiHA: синхронный режим



В Shardman отказоустойчивый кластер BiHA работает в синхронном режиме

Под капотом выполняется
bihactl init … --sync-standbys=
<число_синхронных_ведомых_серверов>

Кроме того, настраивается параметр sync_standbys_min - минимальное количество синхронных ведомых серверов, которые должны быть доступны, чтобы лидер продолжал работу.

Топология кластера Shardman

PostgresPro

Чтобы создать кластер

с необходимой топологией

отказоустойчивости BiHA,

нужно указать параметры

для каждого узла

в файле конфигурации

sdmspec.json

в разделе Topology

```
"Topology": {
  "shard-1": [
    {
      "host": "host1",
      "role": "leader",
      "pgPort": 5432,
      "silkPort": 8000,
      "bihaPort": 25432,
      "priority": 10
    },
    {
      "host": "host2",
      "role": "follower",
      "pgPort": 5433,
      "silkPort": 8001,
      "bihaPort": 25433,
      "priority": 1000
    },
    {
      "host": "host3",
      "role": "referee-with-wal",
      "pgPort": 5434,
      "silkPort": 8000,
      "bihaPort": 25434
    }
  ],
```

```
  "shard-2": [
    {
      "host": "host2",
      "role": "leader",
      "pgPort": 5432,
      "silkPort": 8000,
      "bihaPort": 25432,
      "priority": 10
    },
    {
      "host": "host1",
      "role": "follower",
      "pgPort": 5433,
      "silkPort": 8001,
      "bihaPort": 25433,
      "priority": 1000
    },
    {
      "host": "host3",
      "role": "referee-with-wal",
      "pgPort": 5434,
      "silkPort": 8000,
      "bihaPort": 25435
    }
  ]
}
```

Топология кластера Shardman

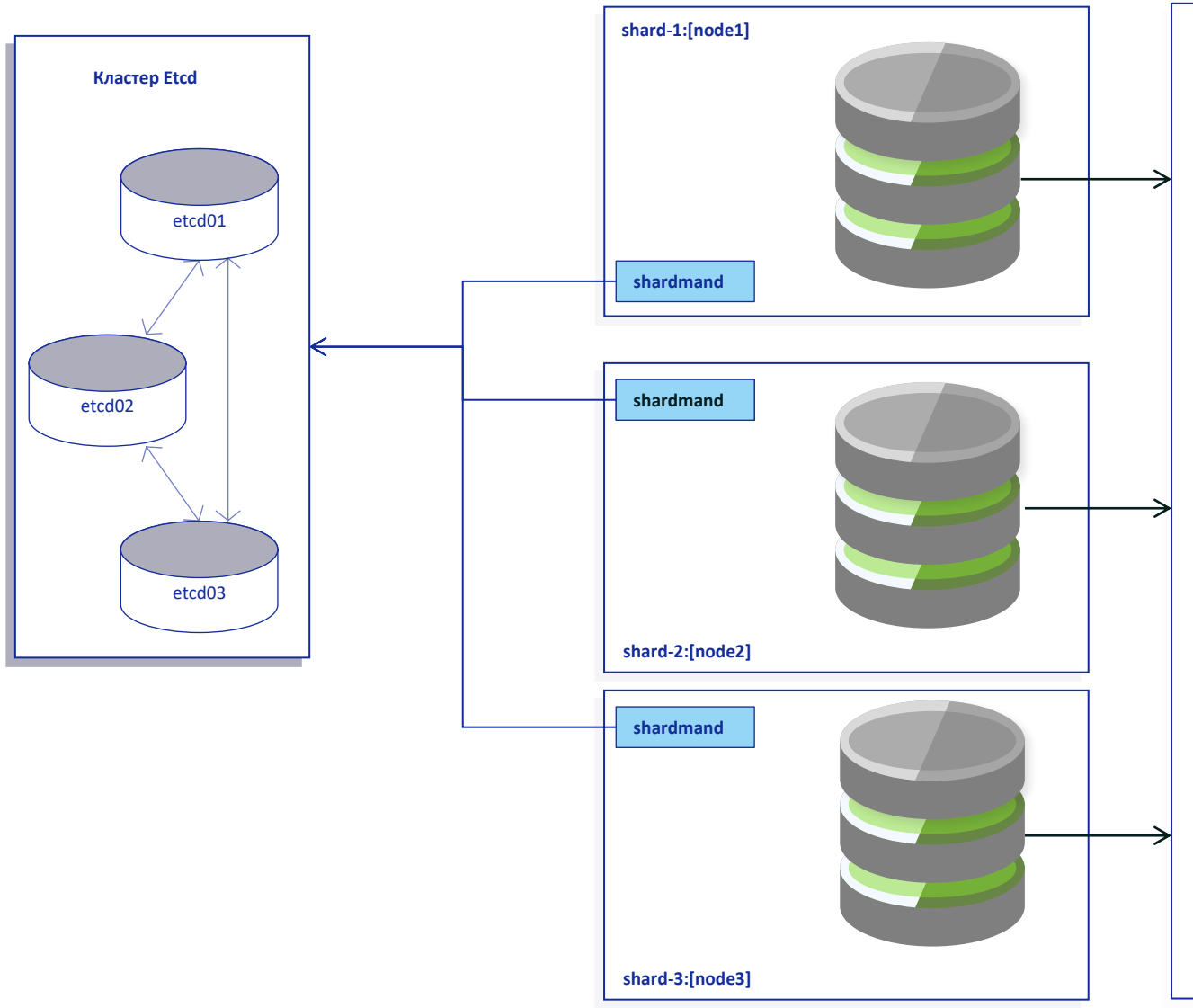
PostgresPro

```
$ shardmanctl cluster topology
```

== REPLICATION GROUP shard-1, RGID - 1 ==				
HOST	PORT	KEEPER	STATUS	PRIORITY
host1	5432	keeper_1	LEADER_RW	10
host2	5433	keeper_2	FOLLOWER	1000
host3	5434	keeper_3	REFEREE_WITH_WAL	NOT SET

== REPLICATION GROUP shard-2, RGID - 2 ==				
HOST	PORT	KEEPER	STATUS	PRIORITY
host2	5432	keeper_1	LEADER_RW	10
host3	5433	keeper_2	FOLLOWER	1000
host1	5434	keeper_3	REFEREE_WITH_WAL	NOT SET

Резервное копирование Shardman



shardmanctl probackup
[init|archive-command|backup|restore|show|validate]

Backup:
1. Блокировка метаданных Shardman, чтобы не допустить изменения во время резервного копирования.
2. Выгрузка метаданных из etcd в файл JSON в каталог резервного копирования.
3. Запуск pg_probackup в каждой группе репликации параллельно.
4. Создаётся точка синхронизации и определяется значения LSN для каждой группы репликации.
5. Затем pg_probackup arhive-push используется для отправки журналов WAL, созданных после завершения резервного копирования, и файла WAL, в котором для каждой группы репликации присутствуют LSN точки синхронизации.

Restore:
1 Восстанавление части метаданных etcd: конфигурацию кластера и части определений групп репликации.
2. Восстановление на LSN точки синхронизации каждой группы репликации с помощью pg_probackup .
3. Для каждой группы репликации файлы WAL, содержащие окончательное значение LSN для восстановления, будут автоматически запрашиваться pg_probackup archive-get.

PostgresPro

Дополнительная информация

- Описание
  https://postgrespro.ru/products/shardman
- Документация
  https://postgrespro.ru/docs/shardman/14
- Публикации
  https://habr.com/ru/companies/postgrespro/articles/811041/
- Презентации
  https://pgconf.ru/talk/1687651

PostgresPro

Спасибо за внимание!