Мамонов Антон 2ИСиП-19-1

Соединение хешированием

Напишите запрос, показывающий занятые места в салоне для всех рейсов. Какой способ соединения выбрал планировщик? Проверьте, хватило ли оперативной памяти для размещения хеш-таблиц. **Не хватило**

```
postgres=# EXPLAIN SELECT f.flight_id, bp.seat_no FROM flights f  JOIN boarding_passes bp ON bp.flight_id = f.flight_id;
                              QUERY PLAN
-----------------------------------------------------------------------
 Hash Join  (cost=19.23..38.34 rows=720 width=24)
   Hash Cond: (bp.flight_id = f.flight_id)
   ->  Seq Scan on boarding_passes bp  (cost=0.00..17.20 rows=720 width=24)
   ->  Hash  (cost=14.10..14.10 rows=410 width=4)
         ->  Seq Scan on flights f  (cost=0.00..14.10 rows=410 width=4)
(5 строк)


postgres=# EXPLAIN (COSTS OFF, ANALYZE) SELECT f.flight_id, bp.seat_no  FROM flights f  JOIN boarding_passes bp ON bp.flight_id = f.fli
ght_id;
                               QUERY PLAN
-------------------------------------------------------------------------
 Hash Join (actual time=0.021..0.023 rows=0 loops=1)
   Hash Cond: (bp.flight_id = f.flight_id)
   ->  Seq Scan on boarding_passes bp (actual time=0.019..0.020 rows=0 loops=1)
   ->  Hash (never executed)
         ->  Seq Scan on flights f (never executed)
 Planning Time: 0.479 ms
 Execution Time: 0.064 ms
(7 строк)
```

Измените запрос, чтобы он выводил только общее количество занятых мест. Как изменился план запроса? Почему планировщик не использовал аналогичный план для предыдущего запроса? **Планировщик использовал параллельный план. В предыдущем запросе это не было оправдано из-за высокой стоимости пересылки данных между процессами, а в данном случае передается только одно число.**

```
postgres=# EXPLAIN SELECT count(*) FROM flights f  JOIN boarding_passes bp ON bp.flight_id = f.flight_id;
                                 QUERY PLAN
----------------------------------------------------------------------------
 Aggregate  (cost=40.14..40.15 rows=1 width=8)
   ->  Hash Join  (cost=19.23..38.34 rows=720 width=0)
         Hash Cond: (bp.flight_id = f.flight_id)
         ->  Seq Scan on boarding_passes bp  (cost=0.00..17.20 rows=720 width=4)
         ->  Hash  (cost=14.10..14.10 rows=410 width=4)
               ->  Seq Scan on flights f  (cost=0.00..14.10 rows=410 width=4)
(6 строк)


postgres=# EXPLAIN (COSTS OFF, ANALYZE) SELECT count(*) FROM flights f JOIN boarding_passes bp ON bp.flight_id = f.flight_id;
                                    QUERY PLAN
-----------------------------------------------------------------------------------
 Aggregate (actual time=0.060..0.062 rows=1 loops=1)
   ->  Hash Join (actual time=0.052..0.054 rows=0 loops=1)
         Hash Cond: (bp.flight_id = f.flight_id)
         ->  Seq Scan on boarding_passes bp (actual time=0.051..0.051 rows=0 loops=1)
         ->  Hash (never executed)
               ->  Seq Scan on flights f (never executed)
 Planning Time: 0.823 ms
 Execution Time: 0.171 ms
(8 строк)
```

Соединение слиянием

```
postgres=# SET log_temp_files = 0;
SET
postgres=#  \timing on
Секундомер включён.
postgres=# SHOW maintenance_work_mem;
 maintenance_work_mem
----------------------
 64MB
(1 строка)


Время: 9,723 мс
postgres=# SHOW maintenance_work_mem;
 maintenance_work_mem
----------------------
 64MB
(1 строка)


Время: 0,826 мс
postgres=# CREATE INDEX ON tickets(passenger_name, passenger_id);
CREATE INDEX
Время: 161,657 мс
postgres=# \timing off
Секундомер выключен.
```

Создайте индекс по столбцам passenger_name и passenger id таблицы билетов (tickets). Потребовался ли временный файл для выполнения этой операции? **Да, потребовался**

Проверьте план выполнения запроса из демонстрации, показывающего все места в салонах в порядке кодов самолетов, но оформленного в виде курсора. Уменьшите значение параметра cursor_tuple_fraction в десять раз. Как при этом изменился план выполнения? **Планировщик выбирает другой план: его первая компонента стоимости меньше.**

```
postgres=# EXPLAIN DECLARE c CURSOR FOR SELECT *  FROM aircrafts a JOIN seats s ON a.aircraft_code = s.aircraft_code ORDER BY a.aircra
t_code;
                                    QUERY PLAN
-----------------------------------------------------------------------------------------
 Merge Join  (cost=0.30..132.15 rows=800 width=142)
   Merge Cond: (a.aircraft_code = s.aircraft_code)
   ->  Index Scan using aircrafts_pkey on aircrafts a  (cost=0.15..63.45 rows=1020 width=52)
   ->  Index Scan using seats_pkey on seats s  (cost=0.15..56.15 rows=800 width=74)
(4 строки)


postgres=# SHOW cursor_tuple_fraction;
 cursor_tuple_fraction
-----------------------
 0.1
(1 строка)


postgres=# SET cursor_tuple_fraction = 0.01;
SET
postgres=# EXPLAIN DECLARE c CURSOR FOR SELECT *  FROM aircrafts a JOIN seats s ON a.aircraft_code = s.aircraft_code ORDER BY a.aircra
t_code;
                                    QUERY PLAN
-----------------------------------------------------------------------------------------
 Merge Join  (cost=0.30..132.15 rows=800 width=142)
   Merge Cond: (a.aircraft_code = s.aircraft_code)
   ->  Index Scan using aircrafts_pkey on aircrafts a  (cost=0.15..63.45 rows=1020 width=52)
   ->  Index Scan using seats_pkey on seats s  (cost=0.15..56.15 rows=800 width=74)
(4 строки)
```