Мамонов Антон 2ИСиП-19-1

Приемы оптимизации

Отключение параллельного выполнения

```
1  SET max_parallel_workers_per_gather = 0;
2
3  EXPLAIN (ANALYZE, TIMING OFF) SELECT t.*
4  FROM  tickets t,
5        ticket_flights tf,
6        flights f
7  WHERE tf.ticket_no = t.ticket_no
8    AND f.flight_id = tf.flight_id
9    AND tf.fare_conditions = 'Business'
10   AND f.actual_departure > f.scheduled_dep
```

	QUERY PLAN text
1	Nested Loop (cost=0.15..34.76 rows=1 width=206) (actual rows=0 loops=1)
2	Join Filter: (tf.ticket_no = t.ticket_no)
3	-> Nested Loop (cost=0.15..33.71 rows=1 width=56) (actual rows=0 loops=1)
4	-> Seq Scan on ticket_flights tf (cost=0.00..17.12 rows=3 width=60) (actual rows=0 loops=1)
5	Filter: ((fare_conditions)::text = 'Business'::text)
6	-> Index Scan using flights_pkey on flights f (cost=0.15..5.50 rows=1 width=4) (never executed)
7	Index Cond: (flight_id = tf.flight_id)
8	Filter: (actual_departure > (scheduled_departure + '05:00:00'::interval))
9	-> Seq Scan on tickets t (cost=0.00..1.02 rows=2 width=206) (never executed)
10	Planning Time: 0.936 ms
11	Execution Time: 0.084 ms

Постройка функционального индекса на разности двух столбцов и переписание условия

```
1  CREATE INDEX ON flights ((actual_departure - scheduled_departure));
```

Результат | План выполнения | Сообщения | Notifications

```
CREATE INDEX

Запрос завершён успешно, время выполнения: 96 msec.
```

```
1  ANALYZE flights;
2  EXPLAIN (ANALYZE, TIMING OFF) SELECT t.*
3  FROM  tickets t,
4        ticket_flights tf,
5        flights f
6  WHERE tf.ticket_no = t.ticket_no
7    AND f.flight_id = tf.flight_id
8    AND tf.fare_conditions = 'Business'
9    AND f.actual_departure - f.scheduled_departure > interval '5 hour';
```

Результат | План выполнения | Сообщения | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.15..34.76 rows=1 width=206) (actual rows=0 loops=1)
2	Join Filter: (tf.ticket_no = t.ticket_no)
3	-> Nested Loop (cost=0.15..33.71 rows=1 width=56) (actual rows=0 loops=1)
4	-> Seq Scan on ticket_flights tf (cost=0.00..17.12 rows=3 width=60) (actual rows=0 loops=1)
5	Filter: ((fare_conditions)::text = 'Business'::text)
6	-> Index Scan using flights_pkey on flights f (cost=0.15..5.50 rows=1 width=4) (never executed)
7	Index Cond: (flight_id = tf.flight_id)
8	Filter: ((actual_departure - scheduled_departure) > '05:00:00'::interval)
9	-> Seq Scan on tickets t (cost=0.00..1.02 rows=2 width=206) (never executed)
10	Planning Time: 0.845 ms
11	Execution Time: 0.086 ms

Теперь займемся таблицей ticket_flights, которая тоже сканируется полностью, хотя из нее читается незначительная часть строк.

Помог бы индекс по классам обслуживания fare_conditions, но лучше создать индекс по столбцу flight_id, что позволит эффективно выполнять соединение вложенным циклом с flights.

```
1  CREATE INDEX ON ticket_flights(flight_id);
2  EXPLAIN (ANALYZE, TIMING OFF) SELECT t.*
3  FROM  tickets t,
4        ticket_flights tf,
5        flights f
6  WHERE tf.ticket_no = t.ticket_no
7    AND f.flight_id = tf.flight_id
8    AND tf.fare_conditions = 'Business'
9    AND f.actual_departure - f.scheduled_departure > interval '5 hour';
```

	QUERY PLAN text
1	Nested Loop (cost=0.15..34.76 rows=1 width=206) (actual rows=0 loops=1)
2	Join Filter: (tf.ticket_no = t.ticket_no)
3	-> Nested Loop (cost=0.15..33.71 rows=1 width=56) (actual rows=0 loops=1)
4	-> Seq Scan on ticket_flights tf (cost=0.00..17.12 rows=3 width=60) (actual rows=0 loops=1)
5	Filter: ((fare_conditions)::text = 'Business'::text)
6	-> Index Scan using flights_pkey on flights f (cost=0.15..5.50 rows=1 width=4) (never executed)
7	Index Cond: (flight_id = tf.flight_id)
8	Filter: ((actual_departure - scheduled_departure) > '05:00:00'::interval)
9	-> Seq Scan on tickets t (cost=0.00..1.02 rows=2 width=206) (never executed)
10	Planning Time: 0.423 ms
11	Execution Time: 0.047 ms

Время планирования и выполнения уменьшилось.

Проверка на собственной БД

Отключение параллельного выполнения

project/postgres@PostgreSQL 13 (64bit)

Query Editor | История запросов

```
1  SET max_parallel_workers_per_gather = 0;
```

Результат | План выполнения | Сообщения | Notifications

```
SET

Запрос завершён успешно, время выполнения: 49 msec.
```

Query Editor | История запросов

```
1  EXPLAIN (ANALYZE, TIMING OFF) SELECT t.*
2  FROM  subjects t,
3        subj_teacher tf,
4        teachers f
5  WHERE tf.subject_id = t.subject_id
6    AND f.teacher_id = tf.teacher_id
7    AND tf.subject_id = 101
```

Результат | План выполнения | Сообщения | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.30..53.95 rows=10 width=36) (actual rows=1 loops=1)
2	-> Nested Loop (cost=0.15..38.77 rows=10 width=40) (actual rows=1 loops=1)
3	-> Index Scan using subjects_pkey on subjects t (cost=0.15..3.17 rows=1 width=36) (actual rows=1 loops=1)
4	Index Cond: (subject_id = 101)
5	-> Seq Scan on subj_teacher tf (cost=0.00..35.50 rows=10 width=8) (actual rows=1 loops=1)
6	Filter: (subject_id = 101)
7	Rows Removed by Filter: 3
8	-> Index Only Scan using teachers_pkey on teachers f (cost=0.15..1.52 rows=1 width=4) (actual rows=1 loops=1)
9	Index Cond: (teacher_id = tf.teacher_id)
10	Heap Fetches: 1
11	Planning Time: 2.288 ms
12	Execution Time: 0.515 ms

Можно построить функциональный индекс на разности двух столбцов и немного переписать условие

```
1  CREATE INDEX ON teachers (surname)
```

Результат | План выполнения | Сообщения | Notifications

```
CREATE INDEX

Запрос завершён успешно, время выполнения: 233 msec.
```

```
1  CREATE INDEX ON subj_teacher (teacher_id);
```

Результат | План выполнения | Сообщения | Notifications

```
CREATE INDEX

Запрос завершён успешно, время выполнения: 75 msec.
```

```
1  EXPLAIN (ANALYZE, TIMING OFF) SELECT t.*
2  FROM  subjects t,
3        subj_teacher tf,
4        teachers f
5  WHERE tf.subject_id = t.subject_id
6    AND f.teacher_id = tf.teacher_id
7    AND tf.subject_id = 101
```

Результат | План выполнения | Сообщения | Notifications

	QUERY PLAN text
1	Hash Join (cost=1.24..39.89 rows=10 width=36) (actual rows=1 loops=1)
2	Hash Cond: (tf.teacher_id = f.teacher_id)
3	-> Nested Loop (cost=0.15..38.77 rows=10 width=40) (actual rows=1 loops=1)
4	-> Index Scan using subjects_pkey on subjects t (cost=0.15..3.17 rows=1 width=36) (actual rows=1 loops=1)
5	Index Cond: (subject_id = 101)
6	-> Seq Scan on subj_teacher tf (cost=0.00..35.50 rows=10 width=8) (actual rows=1 loops=1)
7	Filter: (subject_id = 101)
8	Rows Removed by Filter: 3
9	-> Hash (cost=1.04..1.04 rows=4 width=4) (actual rows=4 loops=1)
10	Buckets: 1024 Batches: 1 Memory Usage: 9kB
11	-> Seq Scan on teachers f (cost=0.00..1.04 rows=4 width=4) (actual rows=4 loops=1)
12	Planning Time: 0.811 ms
13	Execution Time: 0.127 ms

```
1  EXPLAIN (ANALYZE, TIMING OFF) SELECT t.*
2  FROM  subjects t,
3        subj_teacher tf,
4        teachers f
5  WHERE tf.subject_id = t.subject_id
6    AND f.teacher_id = tf.teacher_id
7    AND tf.subject_id = 101
```

Результат | План выполнения | Сообщения | Notifications

	QUERY PLAN text
1	Nested Loop (cost=0.15..5.32 rows=1 width=36) (actual rows=1 loops=1)
2	Join Filter: (tf.teacher_id = f.teacher_id)
3	Rows Removed by Join Filter: 1
4	-> Nested Loop (cost=0.15..4.23 rows=1 width=40) (actual rows=1 loops=1)
5	-> Index Scan using subjects_pkey on subjects t (cost=0.15..3.17 rows=1 width=36) (actual rows=1 loops=1)
6	Index Cond: (subject_id = 101)
7	-> Seq Scan on subj_teacher tf (cost=0.00..1.05 rows=1 width=8) (actual rows=1 loops=1)
8	Filter: (subject_id = 101)
9	Rows Removed by Filter: 3
10	-> Seq Scan on teachers f (cost=0.00..1.04 rows=4 width=4) (actual rows=2 loops=1)
11	Planning Time: 0.195 ms
12	Execution Time: 0.053 ms

Время планирования и выполнения уменьшилось.