Оптимизација на прашалници

Анализа и оптимизација на `Performer_Events`

1. Без индекс

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Performer_Events" WHERE performer_id = 10;
```

QUERY PLAN
Nested Loop (cost=1.14..73.38 rows=7 width=68) (actual time=1505.862..3249.242 rows=4 loops=1)
-> Nested Loop (cost=0.86..70.92 rows=7 width=37) (actual time=1079.648..2259.465 rows=4 loops=1)
-> Index Scan using "Performer_pkey" on "Performer" p (cost=0.28..8.30 rows=1 width=21) (actual time=701.003..701.005 rows=1 loops=1)
Index Cond: (performer_id = 10)
-> Nested Loop (cost=0.57..62.55 rows=7 width=24) (actual time=378.637..1558.442 rows=4 loops=1)
-> Index Only Scan using uq_performer_at_time on "Event_Happening_Performer" ehp (cost=0.29..4.41 rows=7 width=16) (actual time=90.817..90.838 rows=4 loops=1)
Index Cond: (performer_id = 10)
Heap Fetches: 0
-> Index Scan using "Event_Happening_pkey" on "Event_Happening" eh (cost=0.29..8.30 rows=1 width=24) (actual time=366.888..366.889 rows=1 loops=4)
Index Cond: (event_happening_id = ehp.event_happening_id)
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..0.35 rows=1 width=39) (actual time=247.432..247.432 rows=1 loops=4)
Index Cond: (event_id = eh.event_id)
Planning Time: 1816.633 ms
Execution Time: 3249.314 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Event_Happening_Performer" (event_happening_id, performer_id)
    VALUES (1, 10);
```

QUERY PLAN
Insert on "Event_Happening_Performer" (cost=0.00..0.01 rows=0 width=0) (actual time=558.243..558.244 rows=0 loops=1)
-> Result (cost=0.00..0.01 rows=1 width=16) (actual time=0.001..0.002 rows=1 loops=1)
Planning Time: 0.042 ms
Trigger for constraint fk_ehp_event_happening: time=826.105 calls=1
Trigger for constraint fk_ehp_performer: time=40.075 calls=1
Execution Time: 1424.451 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Event_Happening_Performer"
    SET performer_id = 11
    WHERE event_happening_id = 1 AND performer_id = 10;
```

QUERY PLAN
Update on "Event_Happening_Performer" (cost=0.29..8.31 rows=0 width=0) (actual time=0.219..0.219 rows=0 loops=1)
-> Index Scan using uq_performer_at_time on "Event_Happening_Performer" (cost=0.29..8.31 rows=1 width=14) (actual time=0.120..0.121 rows=1 loops=1)
Index Cond: ((performer_id = 10) AND (event_happening_id = 1))
Planning Time: 0.139 ms
Trigger for constraint fk_ehp_performer: time=0.279 calls=1
Execution Time: 23.621 ms

Погледот е бавен поради комплексни врски помеѓи четири табели што резултира со време од 3.2s. Индексите на `performer_id` и `event_id` го елиминираат целосното скенирање на табелите и овозможуваат инстантно поврзување на изведувачите со нивните настани.

```
-- indexes for linking performers with specific event occurrences
(M:N relationship)
CREATE INDEX idx_ehp_performer_id ON
"Event_Happening_Performer"(performer_id);
CREATE INDEX idx_ehp_happening_id ON
"Event_Happening_Performer"(event_happening_id);

-- index for optimizing event lookups within scheduled event
happenings
CREATE INDEX idx_event_happening_event_id ON
"Event_Happening"(event_id);
```

2. Со индекс

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Performer_Events" WHERE performer_id = 10;
```

QUERY PLAN
Nested Loop (cost=1.14..77.38 rows=7 width=68) (actual time=0.251..0.479 rows=4 loops=1)
-> Nested Loop (cost=0.86..74.92 rows=7 width=37) (actual time=0.201..0.306 rows=4 loops=1)
-> Index Scan using "Performer_pkey" on "Performer" p (cost=0.28..8.30 rows=1 width=21) (actual time=0.093..0.094 rows=1 loops=1)
Index Cond: (performer_id = 10)
-> Nested Loop (cost=0.57..66.55 rows=7 width=24) (actual time=0.105..0.207 rows=4 loops=1)
-> Index Only Scan using uq_performer_at_time on "Event_Happening_Performer" ehp (cost=0.29..8.41 rows=7 width=16) (actual time=0.062..0.065 rows=4 loops=1)
Index Cond: (performer_id = 10)
Heap Fetches: 1
-> Index Scan using "Event_Happening_pkey" on "Event_Happening" eh (cost=0.29..8.30 rows=1 width=24) (actual time=0.032..0.032 rows=1 loops=4)
Index Cond: (event_happening_id = ehp.event_happening_id)
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..0.35 rows=1 width=39) (actual time=0.041..0.041 rows=1 loops=4)
Index Cond: (event_id = eh.event_id)
Planning Time: 1.600 ms
Execution Time: 0.533 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Event_Happening_Performer" (event_happening_id,
performer_id)
    VALUES (2, 10);
```

QUERY PLAN
Insert on "Event_Happening_Performer" (cost=0.00..0.01 rows=0 width=0) (actual time=0.307..0.308 rows=0 loops=1)
-> Result (cost=0.00..0.01 rows=1 width=16) (actual time=0.001..0.001 rows=1 loops=1)
Planning Time: 0.046 ms
Trigger for constraint fk_ehp_event_happening: time=0.332 calls=1
Trigger for constraint fk_ehp_performer: time=0.169 calls=1
Execution Time: 0.833 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Event_Happening_Performer"
    SET performer_id = 12
    WHERE event_happening_id = 2 AND performer_id = 10;
```

QUERY PLAN
Update on "Event_Happening_Performer" (cost=0.29..8.31 rows=0 width=0) (actual time=0.437..0.438 rows=0 loops=1)
-> Index Scan using idx_ehp_happening_id on "Event_Happening_Performer" (cost=0.29..8.31 rows=1 width=14) (actual time=0.174..0.175 rows=1 loops=1)
Index Cond: (event_happening_id = 2)
Filter: (performer_id = 10)
Rows Removed by Filter: 1
Planning Time: 0.161 ms
Trigger for constraint fk_ehp_performer: time=0.271 calls=1
Execution Time: 0.763 ms

Анализа и оптимизација на `Venue_Layout`

1. **Без индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Venue_Layout" WHERE venue_id = 1;
```

QUERY PLAN
Nested Loop (cost=1001.14..10398.19 rows=2264 width=55) (actual time=447.576..2164.258 rows=775 loops=1)
-> Index Scan using "Venue_pkey" on "Venue" v (cost=0.29..8.30 rows=1 width=28) (actual time=68.551..68.555 rows=1 loops=1)
Index Cond: (venue_id = 1)
-> Gather (cost=1000.85..10367.25 rows=2264 width=35) (actual time=379.017..2095.610 rows=775 loops=1)
Workers Planned: 1
Workers Launched: 1
-> Nested Loop (cost=0.85..9140.85 rows=1332 width=35) (actual time=932.279..1787.769 rows=388 loops=2)
-> Parallel Index Scan using "Section_pkey" on "Section" s (cost=0.29..1691.80 rows=4 width=23) (actual time=771.383..1395.534 rows=2 loops=2)
Filter: (venue_id = 1)
Rows Removed by Filter: 27502
-> Index Scan using uq_seat_section_number on "Seat" st (cost=0.56..1853.21 rows=905 width=20) (actual time=64.373..156.866 rows=155 loops=5)
Index Cond: (section_id = s.section_id)
Planning Time: 1874.048 ms
Execution Time: 2164.361 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Seat" (seat_id, section_id, seat_number)
    SELECT COALESCE(MAX(seat_id), 0) + 1, 1, 999999 FROM "Seat";
```

QUERY PLAN
Insert on "Seat" (cost=0.67..0.69 rows=0 width=0) (actual time=745.461..745.464 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.67..0.69 rows=1 width=20) (actual time=481.921..481.925 rows=1 loops=1)
-> Result (cost=0.67..0.69 rows=1 width=16) (actual time=481.918..481.920 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.56..0.67 rows=1 width=8) (actual time=481.909..481.910 rows=1 loops=1)
-> Index Only Scan Backward using "Seat_pkey" on "Seat" "Seat_1" (cost=0.56..2330912.81 rows=20753208 width=8) (actual time=481.907..481.908 rows=1 loops=1)
Heap Fetches: 0
Planning Time: 0.339 ms
Trigger for constraint fk_seat_section: time=0.533 calls=1
Execution Time: 746.039 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Seat"
    SET seat_number = 888888
    WHERE seat_number = 999999;
```

QUERY PLAN
Update on "Seat" (cost=0.00..391602.10 rows=0 width=0) (actual time=308799.033..308799.035 rows=0 loops=1)
-> Seq Scan on "Seat" (cost=0.00..391602.10 rows=11492 width=10) (actual time=308792.032..308792.034 rows=1 loops=1)
Filter: (seat_number = 999999)
Rows Removed by Filter: 20753209
Planning Time: 0.114 ms
JIT:
Functions: 4
Options: Inlining false, Optimization false, Expressions true, Deforming true
Timing: Generation 0.405 ms (Deform 0.071 ms), Inlining 0.000 ms, Optimization 0.595 ms, Emission 6.566 ms, Total 7.565 ms
Execution Time: 312239.424 ms

Времето за ажурирање од 312s е неприфатливо за интеракција со мапа на седишта во реално време. Со поставување индекси на `seat_number` и `venue_id`, пребарувањето и промената на статусот на седиштата се извршуваат за милисекунди наместо за неколку минути.

```sql
-- index for linking seats to their respective sections
CREATE INDEX idx_seat_section_id ON "Seat"(section_id);

-- index for linking sections to venues
CREATE INDEX idx_section_venue_id ON "Section"(venue_id);

-- index for optimizing search and update operations on specific seat
numbers
CREATE INDEX idx_seat_number ON "Seat"(seat_number);
```

2. **Со индекс**

- **SELECT**

```sql
EXPLAIN ANALYZE
    SELECT * FROM "Venue_Layout" WHERE venue_id = 1;
```

QUERY PLAN
Nested Loop (cost=1.01..3060.48 rows=2264 width=55) (actual time=0.184..0.512 rows=776 loops=1)
-> Nested Loop (cost=0.57..16.76 rows=6 width=43) (actual time=0.140..0.144 rows=5 loops=1)
-> Index Scan using "Venue_pkey" on "Venue" v (cost=0.29..8.30 rows=1 width=28) (actual time=0.077..0.078 rows=1 loops=1)
Index Cond: (venue_id = 1)
-> Index Scan using idx_section_venue_id on "Section" s (cost=0.29..8.39 rows=6 width=23) (actual time=0.058..0.060 rows=5 loops=1)
Index Cond: (venue_id = 1)
-> Index Scan using idx_seat_section_id on "Seat" st (cost=0.44..498.24 rows=905 width=20) (actual time=0.012..0.053 rows=155 loops=5)
Index Cond: (section_id = s.section_id)
Planning Time: 1.235 ms
Execution Time: 0.572 ms

- **INSERT**

```sql
EXPLAIN ANALYZE
    INSERT INTO "Seat" (seat_id, section_id, seat_number)
    SELECT COALESCE(MAX(seat_id), 0) + 1, 1, 111222 FROM "Seat";
```

QUERY PLAN
Insert on "Seat" (cost=0.67..0.69 rows=0 width=0) (actual time=0.493..0.494 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.67..0.69 rows=1 width=20) (actual time=0.169..0.171 rows=1 loops=1)
-> Result (cost=0.67..0.69 rows=1 width=16) (actual time=0.167..0.168 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.56..0.67 rows=1 width=8) (actual time=0.162..0.163 rows=1 loops=1)
-> Index Only Scan Backward using "Seat_pkey" on "Seat" "Seat_1" (cost=0.56..2330912.84 rows=20753210 width=8) (actual time=0.161..0.161 rows=1 loops=1)
Heap Fetches: 1
Planning Time: 0.219 ms
Trigger for constraint fk_seat_section: time=0.300 calls=1
Execution Time: 0.839 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Seat"
    SET seat_number = 333444
    WHERE seat_number = 111222;
```

QUERY PLAN
Update on "Seat" (cost=0.44..44297.09 rows=0 width=0) (actual time=0.395..0.396 rows=0 loops=1)
-> Index Scan using idx_seat_number on "Seat" (cost=0.44..44297.09 rows=11492 width=10) (actual time=0.193..0.195 rows=1 loops=1)
Index Cond: (seat_number = 111222)
Planning Time: 0.141 ms
Execution Time: 0.461 ms

Анализа и оптимизација на `User_Tickets`

1. **Без индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "User_Tickets" WHERE user_id = 1;
```

QUERY PLAN
Nested Loop Left Join (cost=1001.42..266130.92 rows=1 width=76) (actual time=251932.834..251942.957 rows=0 loops=1)
-> Nested Loop (cost=1000.99..266122.47 rows=1 width=60) (actual time=251932.833..251942.955 rows=0 loops=1)
-> Nested Loop (cost=1000.43..266113.89 rows=1 width=60) (actual time=251932.833..251942.953 rows=0 loops=1)
-> Index Scan using "User_pkey" on "User" u (cost=0.43..8.45 rows=1 width=26) (actual time=374.404..374.410 rows=1 loops=1)
Index Cond: (user_id = 1)
-> Gather (cost=1000.00..266105.43 rows=1 width=42) (actual time=251558.416..251568.535 rows=0 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Parallel Seq Scan on "Ticket_Purchase" tp (cost=0.00..265105.33 rows=1 width=42) (actual time=251075.725..251075.726 rows=0 loops=3)
Filter: (user_id = 1)
Rows Removed by Filter: 5333333
-> Index Only Scan using "Ticket_pkey" on "Ticket" t (cost=0.56..8.58 rows=1 width=8) (never executed)
Index Cond: (ticket_id = tp.ticket_id)
Heap Fetches: 0
-> Index Scan using "Ticket_Refund_purchase_id_key" on "Ticket_Refund" tr (cost=0.43..8.45 rows=1 width=24) (never executed)
Index Cond: (purchase_id = tp.purchase_id)
Planning Time: 6624.478 ms
JIT:
Functions: 24
Options: Inlining false, Optimization false, Expressions true, Deforming true
Timing: Generation 2.284 ms (Deform 0.859 ms), Inlining 0.000 ms, Optimization 1.843 ms, Emission 23.281 ms, Total 27.408 ms
Execution Time: 251944.476 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Ticket_Purchase" (purchase_id, ticket_id, user_id, qr_code, purchase_amount)
    SELECT COALESCE(MAX(purchase_id), 0) + 1, 1, 1, 'QR-TEST-CODE-001', 1200.00
    FROM "Ticket_Purchase";
```

QUERY PLAN
Insert on "Ticket_Purchase" (cost=0.60..0.63 rows=0 width=0) (actual time=106.723..106.725 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.60..0.63 rows=1 width=552) (actual time=106.587..106.590 rows=1 loops=1)
-> Result (cost=0.60..0.61 rows=1 width=80) (actual time=106.570..106.571 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.43..0.60 rows=1 width=8) (actual time=106.562..106.563 rows=1 loops=1)
-> Index Only Scan Backward using "Ticket_Purchase_pkey" on "Ticket_Purchase" "Ticket_Purchase_1" (cost=0.43..2592572.76 rows=16000000 width=8) (actual time=106.560..106.561 rows=1 loops=1)
Heap Fetches: 1
Planning Time: 0.194 ms
Trigger for constraint fk_purchase_ticket: time=0.320 calls=1
Trigger for constraint fk_purchase_user: time=0.249 calls=1
Execution Time: 107.346 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Ticket_Purchase"
    SET purchase_amount = 1500.00
    WHERE user_id = 1 AND qr_code = 'QR-TEST-CODE-001';
```

QUERY PLAN
Update on "Ticket_Purchase" (cost=0.56..8.58 rows=0 width=0) (actual time=0.223..0.224 rows=0 loops=1)
-> Index Scan using "Ticket_Purchase_qr_code_key" on "Ticket_Purchase" (cost=0.56..8.58 rows=1 width=10) (actual time=0.181..0.183 rows=1 loops=1)
Index Cond: ((qr_code)::text = 'QR-TEST-CODE-001'::text)
Filter: (user_id = 1)
Planning Time: 0.139 ms
Execution Time: 0.251 ms

Приказот на историјата на билети трае предолги 251.9s, што го блокира корисничкиот профил. Индексот на `user_id` овозможува базата веднаш да ги лоцира билетите на конкретниот корисник без да ги пребарува сите трансакции во системот.

```sql
-- index for linking ticket purchases to the specific tickets
CREATE INDEX idx_ticket_purchase_ticket_id ON
"Ticket_Purchase"(ticket_id);

-- index for linking purchases to users
CREATE INDEX idx_ticket_purchase_user_id ON
"Ticket_Purchase"(user_id);

-- index for the LEFT JOIN with refunds
CREATE INDEX idx_ticket_refund_purchase_id ON
"Ticket_Refund"(purchase_id);
```

2. **Со индекс**

- **SELECT**

```sql
EXPLAIN ANALYZE
    SELECT * FROM "User_Tickets" WHERE user_id = 1;
```

QUERY PLAN
Nested Loop Left Join (cost=1001.42..266130.93 rows=1 width=76) (actual time=650.104..658.718 rows=1 loops=1)
-> Nested Loop (cost=1000.99..266122.48 rows=1 width=60) (actual time=650.074..658.687 rows=1 loops=1)
-> Nested Loop (cost=1000.43..266113.90 rows=1 width=60) (actual time=650.026..658.637 rows=1 loops=1)
-> Index Scan using "User_pkey" on "User" u (cost=0.43..8.45 rows=1 width=26) (actual time=0.164..0.168 rows=1 loops=1)
Index Cond: (user_id = 1)
-> Gather (cost=1000.00..266105.44 rows=1 width=42) (actual time=649.850..658.456 rows=1 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Parallel Seq Scan on "Ticket_Purchase" tp (cost=0.00..265105.34 rows=1 width=42) (actual time=620.207..620.208 rows=0 loops=3)
Filter: (user_id = 1)
Rows Removed by Filter: 5333333
-> Index Only Scan using "Ticket_pkey" on "Ticket" t (cost=0.56..8.58 rows=1 width=8) (actual time=0.027..0.028 rows=1 loops=1)
Index Cond: (ticket_id = tp.ticket_id)
Heap Fetches: 0
-> Index Scan using idx_ticket_refund_purchase_id on "Ticket_Refund" tr (cost=0.43..8.45 rows=1 width=24) (actual time=0.012..0.012 rows=0 loops=1)
Index Cond: (purchase_id = tp.purchase_id)
Planning Time: 851.914 ms
JIT:
Functions: 24
Options: Inlining false, Optimization false, Expressions true, Deforming true
Timing: Generation 1.553 ms (Deform 0.658 ms), Inlining 0.000 ms, Optimization 1.275 ms, Emission 22.118 ms, Total 24.946 ms
Execution Time: 922.978 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Ticket_Purchase" (purchase_id, ticket_id, user_id,
qr_code, purchase_amount)
    SELECT COALESCE(MAX(purchase_id), 0) + 1, 1, 1, 'QR-TEST-CODE-
002', 1200.00
    FROM "Ticket_Purchase";
```

QUERY PLAN
Insert on "Ticket_Purchase" (cost=0.70..0.73 rows=0 width=0) (actual time=0.367..0.368 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.70..0.73 rows=1 width=552) (actual time=0.039..0.041 rows=1 loops=1)
-> Result (cost=0.70..0.71 rows=1 width=80) (actual time=0.027..0.028 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.43..0.70 rows=1 width=8) (actual time=0.023..0.024 rows=1 loops=1)
-> Index Only Scan Backward using "Ticket_Purchase_pkey" on "Ticket_Purchase" "Ticket_Purchase_1" (cost=0.43..4162966.51 rows=16000002 width=8) (actual time=0.022..0.022 rows=1 loops=1)
Heap Fetches: 2
Planning Time: 0.269 ms
Trigger for constraint fk_purchase_ticket: time=0.334 calls=1
Trigger for constraint fk_purchase_user: time=0.197 calls=1
Execution Time: 0.952 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Ticket_Purchase"
    SET purchase_amount = 1350.00
    WHERE user_id = 1 AND qr_code = 'QR-TEST-CODE-002';
```

QUERY PLAN
Update on "Ticket_Purchase" (cost=0.56..8.58 rows=0 width=0) (actual time=0.137..0.138 rows=0 loops=1)
-> Index Scan using "Ticket_Purchase_qr_code_key" on "Ticket_Purchase" (cost=0.56..8.58 rows=1 width=10) (actual time=0.078..0.079 rows=1 loops=1)
Index Cond: ((qr_code)::text = 'QR-TEST-CODE-002'::text)
Filter: (user_id = 1)
Planning Time: 0.150 ms
Execution Time: 0.174 ms

Анализа и оптимизација на `Event_User_Ratings`

1. **Без индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Event_User_Ratings"
    WHERE user_id = 1;
```

QUERY PLAN
Nested Loop (cost=1.29..21.42 rows=1 width=131) (actual time=0.104..0.105 rows=0 loops=1)
-> Nested Loop (cost=0.86..12.96 rows=1 width=113) (actual time=0.103..0.105 rows=0 loops=1)
-> Nested Loop (cost=0.57..12.61 rows=1 width=82) (actual time=0.103..0.104 rows=0 loops=1)
-> Index Scan using idx_ehr_user_id on "Event_Happening_Rating" ehr (cost=0.29..4.30 rows=1 width=74) (actual time=0.103..0.103 rows=0 loops=1)
Index Cond: (user_id = 1)
-> Index Scan using "Event_Happening_pkey" on "Event_Happening" eh (cost=0.29..8.30 rows=1 width=16) (never executed)
Index Cond: (event_happening_id = ehr.event_happening_id)
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..0.35 rows=1 width=39) (never executed)
Index Cond: (event_id = eh.event_id)
-> Index Scan using "User_pkey" on "User" u (cost=0.43..8.45 rows=1 width=26) (never executed)
Index Cond: (user_id = 1)
Planning Time: 464.594 ms
Execution Time: 0.187 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Event_Happening_Rating" (rating_id,
event_happening_id, user_id, rating, comment)
    SELECT COALESCE(MAX(rating_id), 0) + 1, 1, 1, 5, 'Test rating'
    FROM "Event_Happening_Rating";
```

QUERY PLAN
Insert on "Event_Happening_Rating" (cost=0.31..0.34 rows=0 width=0) (actual time=0.521..0.522 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.31..0.34 rows=1 width=60) (actual time=0.122..0.123 rows=1 loops=1)
-> Result (cost=0.31..0.33 rows=1 width=52) (actual time=0.119..0.120 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.29..0.31 rows=1 width=8) (actual time=0.114..0.115 rows=1 loops=1)
-> Index Only Scan Backward using "Event_Happening_Rating_pkey" on "Event_Happening_Rating" "Event_Happening_Rating_1" (cost=0.29..478.37 rows=17872 width=8) (actual time=0.113..0.113 rows=1 loops=1)
Heap Fetches: 6
Planning Time: 0.297 ms
Trigger for constraint fk_event_happening_rating_event_happening: time=0.490 calls=1
Trigger for constraint fk_event_happening_rating_user: time=0.370 calls=1
Execution Time: 1.444 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Event_Happening_Rating"
    SET rating = 4, comment = 'New test rating'
    WHERE event_happening_id = 1 AND user_id = 1;
```

QUERY PLAN
Update on "Event_Happening_Rating" (cost=0.29..4.31 rows=0 width=0) (actual time=0.176..0.177 rows=0 loops=1)
-> Index Scan using idx_ehr_user_id on "Event_Happening_Rating" (cost=0.29..4.31 rows=1 width=42) (actual time=0.039..0.042 rows=1 loops=1)
Index Cond: (user_id = 1)
Filter: (event_happening_id = 1)
Planning Time: 0.153 ms
Execution Time: 0.218 ms

Без индекси, секое пребарување на оценките по корисник предизвикува непотребно оптоварување на меморијата преку **Seq Scan**. Индексирањето на `user_id` и `event_happening_id` обезбедува брза филтрација и поврзување на рејтинзите со соодветните термини на настаните.

```
-- index for linking ratings to event happenings
CREATE INDEX idx_ehr_happening_id ON
"Event_Happening_Rating"(event_happening_id);

-- index for linking ratings to users
CREATE INDEX idx_ehr_user_id ON "Event_Happening_Rating"(user_id);
```

2. **Со индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Event_User_Ratings" WHERE user_id = 1;
```

QUERY PLAN
Nested Loop (cost=1.00..476.51 rows=1 width=131) (actual time=359.514..359.524 rows=1 loops=1)
-> Nested Loop (cost=0.57..468.06 rows=1 width=113) (actual time=359.389..359.396 rows=1 loops=1)
-> Nested Loop (cost=0.29..467.71 rows=1 width=82) (actual time=2.104..2.110 rows=1 loops=1)
-> Seq Scan on "Event_Happening_Rating" ehr (cost=0.00..459.40 rows=1 width=74) (actual time=2.038..2.041 rows=1 loops=1)
Filter: (user_id = 1)
Rows Removed by Filter: 17871
-> Index Scan using "Event_Happening_pkey" on "Event_Happening" eh (cost=0.29..8.30 rows=1 width=16) (actual time=0.061..0.062 rows=1 loops=1)
Index Cond: (event_happening_id = ehr.event_happening_id)
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..0.35 rows=1 width=39) (actual time=357.279..357.280 rows=1 loops=1)
Index Cond: (event_id = eh.event_id)
-> Index Scan using "User_pkey" on "User" u (cost=0.43..8.45 rows=1 width=26) (actual time=0.120..0.121 rows=1 loops=1)
Index Cond: (user_id = 1)
Planning Time: 31.049 ms
Execution Time: 359.600 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Event_Happening_Rating" (rating_id,
event_happening_id, user_id, rating, comment)
    SELECT COALESCE(MAX(rating_id), 0) + 1, 2, 1, 5, 'Test rating'
    FROM "Event_Happening_Rating";
```

QUERY PLAN
Insert on "Event_Happening_Rating" (cost=0.31..0.34 rows=0 width=0) (actual time=0.482..0.483 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.31..0.34 rows=1 width=60) (actual time=0.117..0.119 rows=1 loops=1)
-> Result (cost=0.31..0.33 rows=1 width=52) (actual time=0.115..0.115 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.29..0.31 rows=1 width=8) (actual time=0.110..0.111 rows=1 loops=1)
-> Index Only Scan Backward using "Event_Happening_Rating_pkey" on "Event_Happening_Rating" "Event_Happening_Rating_1" (cost=0.29..484.37 rows=17872 width=8) (actual time=0.108..0.108 rows=1 loops=1)
Heap Fetches: 4
Planning Time: 0.203 ms
Trigger for constraint fk_event_happening_rating_event_happening: time=0.197 calls=1
Trigger for constraint fk_event_happening_rating_user: time=0.114 calls=1
Execution Time: 0.842 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Event_Happening_Rating"
    SET rating = 6, comment = 'New test rating'
    WHERE event_happening_id = 1 AND user_id = 2;
```

QUERY PLAN
Update on "Event_Happening_Rating" (cost=0.29..8.31 rows=0 width=0) (actual time=0.021..0.022 rows=0 loops=1)
-> Index Scan using uq_rating_happening_user on "Event_Happening_Rating" (cost=0.29..8.31 rows=1 width=42) (actual time=0.020..0.021 rows=0 loops=1)
Index Cond: ((event_happening_id = 1) AND (user_id = 2))
Planning Time: 0.132 ms
Execution Time: 0.081 ms

Анализа и оптимизација на `Event_Overall_Ratings`

1. **Без индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Event_Overall_Ratings" WHERE event_id = 1;
```

QUERY PLAN
GroupAggregate (cost=72.25..72.32 rows=3 width=95) (actual time=1029.642..1029.647 rows=1 loops=1)
Group Key: eh.event_happening_id
-> Sort (cost=72.25..72.26 rows=3 width=67) (actual time=1029.557..1029.604 rows=1 loops=1)
Sort Key: eh.event_happening_id
Sort Method: quicksort Memory: 25kB
-> Nested Loop (cost=4.90..72.22 rows=3 width=67) (actual time=223.616..1029.548 rows=1 loops=1)
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..8.30 rows=1 width=39) (actual time=0.046..0.053 rows=1 loops=1)
Index Cond: (event_id = 1)
-> Nested Loop (cost=4.61..63.89 rows=3 width=36) (actual time=223.566..1029.488 rows=1 loops=1)
-> Bitmap Heap Scan on "Event_Happening" eh (cost=4.33..22.32 rows=5 width=24) (actual time=0.054..805.890 rows=5 loops=1)
Recheck Cond: (event_id = 1)
Heap Blocks: exact=5
-> Bitmap Index Scan on idx_event_happening_event_id (cost=0.00..4.32 rows=5 width=0) (actual time=0.029..0.029 rows=5 loops=1)
Index Cond: (event_id = 1)
-> Index Scan using uq_rating_happening_user on "Event_Happening_Rating" ehr (cost=0.29..8.30 rows=1 width=20) (actual time=44.710..44.711 rows=0 loops=5)
Index Cond: (event_happening_id = eh.event_happening_id)
Planning Time: 1.025 ms
Execution Time: 1029.756 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Event_Happening_Rating" (rating_id,
event_happening_id, user_id, rating, comment)
    SELECT COALESCE(MAX(rating_id), 0) + 1, 1, 15, 9, 'Test rating'
    FROM "Event_Happening_Rating";
```

QUERY PLAN
Insert on "Event_Happening_Rating" (cost=0.31..0.34 rows=0 width=0) (actual time=0.405..0.407 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.31..0.34 rows=1 width=60) (actual time=0.156..0.158 rows=1 loops=1)
-> Result (cost=0.31..0.33 rows=1 width=52) (actual time=0.154..0.155 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.29..0.31 rows=1 width=8) (actual time=0.149..0.149 rows=1 loops=1)
-> Index Only Scan Backward using "Event_Happening_Rating_pkey" on "Event_Happening_Rating" "Event_Happening_Rating_1" (cost=0.29..484.37 rows=17872 width=8) (actual time=0.148..0.148 rows=1 loops=1)
Heap Fetches: 2
Planning Time: 0.199 ms
Trigger for constraint fk_event_happening_rating_event_happening: time=0.232 calls=1
Trigger for constraint fk_event_happening_rating_user: time=0.234 calls=1
Execution Time: 0.918 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Event_Happening_Rating"
    SET rating = 8, comment = 'New test rating'
    WHERE event_happening_id = 1 AND user_id = 15;
```

QUERY PLAN
Update on "Event_Happening_Rating" (cost=0.29..8.31 rows=0 width=0) (actual time=0.458..0.459 rows=0 loops=1)
-> Index Scan using uq_rating_happening_user on "Event_Happening_Rating" (cost=0.29..8.31 rows=1 width=42) (actual time=0.131..0.133 rows=1 loops=1)
Index Cond: ((event_happening_id = 1) AND (user_id = 15))
Planning Time: 0.136 ms
Execution Time: 0.500 ms

Пресметката на просечни оценки бара постојано агрегирање на податоци, што е бавно при секој нов приказ. Композитен индекс на `(event_happening_id, rating)` овозможува математичките операции да се вршат директно врз индексот, забрзувајќи го приказот на почетната страна.

```
-- composite index to speed up grouping and aggregate calculations
(AVG, COUNT)
CREATE INDEX idx_ehr_happening_id_rating ON
"Event_Happening_Rating"(event_happening_id, rating);
```

2. **Со индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Event_Overall_Ratings" WHERE event_id = 1;
```

QUERY PLAN
GroupAggregate (cost=72.25..72.32 rows=3 width=95) (actual time=0.387..0.389 rows=1 loops=1)
Group Key: eh.event_happening_id
-> Sort (cost=72.25..72.26 rows=3 width=67) (actual time=0.374..0.375 rows=1 loops=1)
Sort Key: eh.event_happening_id
Sort Method: quicksort Memory: 25kB
-> Nested Loop (cost=4.90..72.22 rows=3 width=67) (actual time=0.271..0.367 rows=1 loops=1)
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..8.30 rows=1 width=39) (actual time=0.126..0.128 rows=1 loops=1)
Index Cond: (event_id = 1)
-> Nested Loop (cost=4.61..63.89 rows=3 width=36) (actual time=0.142..0.235 rows=1 loops=1)
-> Bitmap Heap Scan on "Event_Happening" eh (cost=4.33..22.32 rows=5 width=24) (actual time=0.067..0.144 rows=5 loops=1)
Recheck Cond: (event_id = 1)
Heap Blocks: exact=5
-> Bitmap Index Scan on idx_event_happening_event_id (cost=0.00..4.32 rows=5 width=0) (actual time=0.038..0.038 rows=5 loops=1)
Index Cond: (event_id = 1)
-> Index Scan using uq_rating_happening_user on "Event_Happening_Rating" ehr (cost=0.29..8.30 rows=1 width=20) (actual time=0.016..0.016 rows=0 loops=5)
Index Cond: (event_happening_id = eh.event_happening_id)
Planning Time: 1.134 ms
Execution Time: 0.459 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Event_Happening_Rating" (rating_id,
event_happening_id, user_id, rating, comment)
    SELECT COALESCE(MAX(rating_id), 0) + 1, 1, 20, 7, 'Test rating'
    FROM "Event_Happening_Rating";
```

QUERY PLAN
Insert on "Event_Happening_Rating" (cost=0.31..0.34 rows=0 width=0) (actual time=0.538..0.539 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.31..0.34 rows=1 width=60) (actual time=0.111..0.112 rows=1 loops=1)
-> Result (cost=0.31..0.33 rows=1 width=52) (actual time=0.109..0.109 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.29..0.31 rows=1 width=8) (actual time=0.104..0.105 rows=1 loops=1)
-> Index Only Scan Backward using "Event_Happening_Rating_pkey" on "Event_Happening_Rating" "Event_Happening_Rating_1" (cost=0.29..485.40 rows=17874 width=8) (actual time=0.102..0.103 rows=1 loops=1)
Heap Fetches: 1
Planning Time: 0.195 ms
Trigger for constraint fk_event_happening_rating_event_happening: time=0.223 calls=1
Trigger for constraint fk_event_happening_rating_user: time=0.172 calls=1
Execution Time: 0.980 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Event_Happening_Rating"
    SET rating = 8, comment = 'New test rating'
    WHERE event_happening_id = 1 AND user_id = 15;
```

QUERY PLAN
Update on "Event_Happening_Rating" (cost=0.29..8.31 rows=0 width=0) (actual time=0.482..0.483 rows=0 loops=1)
-> Index Scan using uq_rating_happening_user on "Event_Happening_Rating" (cost=0.29..8.31 rows=1 width=42) (actual time=0.096..0.098 rows=1 loops=1)
Index Cond: ((event_happening_id = 1) AND (user_id = 20))
Planning Time: 0.163 ms
Execution Time: 0.548 ms

Анализа и оптимизација на `Event_Financial_Summary`

1. **Без индекс**

- **SELECT**

```
EXPLAIN ANALYZE
    SELECT * FROM "Event_Financial_Summary" WHERE event_id = 1;
```

QUERY PLAN
Finalize GroupAggregate (cost=300491.02..300511.03 rows=5 width=71) (actual time=319308.288..319318.376 rows=1 loops=1)
Group Key: eh.event_happening_id
-> Gather Merge (cost=300491.02..300510.84 rows=10 width=75) (actual time=319308.204..319318.298 rows=1 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Partial GroupAggregate (cost=299491.00..299509.67 rows=5 width=75) (actual time=319135.885..319135.894 rows=0 loops=3)
Group Key: eh.event_happening_id
-> Sort (cost=299491.00..299493.66 rows=1064 width=79) (actual time=319135.736..319135.767 rows=388 loops=3)
Sort Key: eh.event_happening_id
Sort Method: quicksort Memory: 25kB
Worker 0: Sort Method: quicksort Memory: 25kB
Worker 1: Sort Method: quicksort Memory: 160kB
-> Nested Loop Left Join (cost=17569.44..299437.50 rows=1064 width=79) (actual time=257215.363..319135.341 rows=388 loops=3)
-> Parallel Hash Join (cost=17569.00..291013.94 rows=1064 width=67) (actual time=257187.356..319052.018 rows=388 loops=3)
Hash Cond: (tp.ticket_id = t.ticket_id)
-> Parallel Seq Scan on "Ticket_Purchase" tp (cost=0.00..248438.67 rows=6666668 width=20) (actual time=39.675..315154.157 rows=5333334 loops=3)
-> Parallel Hash (cost=17536.03..17536.03 rows=2638 width=63) (actual time=2787.389..2787.393 rows=1442 loops=3)
Buckets: 8192 Batches: 1 Memory Usage: 576kB
-> Nested Loop (cost=1.14..17536.03 rows=2638 width=63) (actual time=1483.997..2465.151 rows=1442 loops=3)
-> Nested Loop (cost=0.57..1029.33 rows=3 width=55) (actual time=632.603..753.210 rows=2 loops=3)
-> Parallel Index Scan using "Event_Happening_pkey" on "Event_Happening" eh (cost=0.29..1004.39 rows=3 width=24) (actual time=632.410..752.992 rows=2 loops=3)
Filter: (event_id = 1)
Rows Removed by Filter: 10444
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..8.30 rows=1 width=39) (actual time=0.094..0.100 rows=1 loops=5)
Index Cond: (event_id = 1)
-> Index Scan using uq_ticket_event_happening_seat on "Ticket" t (cost=0.56..5486.81 rows=1542 width=16) (actual time=652.636..1027.008 rows=865 loops=5)
Index Cond: (event_happening_id = eh.event_happening_id)
-> Index Scan using idx_ticket_refund_purchase_id on "Ticket_Refund" tr (cost=0.43..7.92 rows=1 width=20) (actual time=0.214..0.214 rows=0 loops=1163)
Index Cond: (purchase_id = tp.purchase_id)
Planning Time: 2770.897 ms
JIT:
Functions: 96

Options: Inlining false, Optimization false, Expressions true, Deforming true
Timing: Generation 9.017 ms (Deform 4.245 ms), Inlining 0.000 ms, Optimization 4.730 ms, Emission 79.393 ms, Total 93.140 ms
Execution Time: 319320.792 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Ticket_Purchase" (purchase_id, ticket_id, user_id,
qr_code, purchase_time, purchase_amount)
    SELECT COALESCE(MAX(purchase_id), 0) + 1, 1, 1, 'QR-TEST-CODE-
003', CURRENT_TIMESTAMP, 1500.00
    FROM "Ticket_Purchase";
```

QUERY PLAN
Insert on "Ticket_Purchase" (cost=0.70..0.73 rows=0 width=0) (actual time=1391.459..1391.461 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.70..0.73 rows=1 width=552) (actual time=0.132..0.137 rows=1 loops=1)
-> Result (cost=0.70..0.71 rows=1 width=88) (actual time=0.106..0.109 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.43..0.70 rows=1 width=8) (actual time=0.101..0.102 rows=1 loops=1)
-> Index Only Scan Backward using "Ticket_Purchase_pkey" on "Ticket_Purchase" "Ticket_Purchase_1" (cost=0.43..4162966.51 rows=16000002 width=8) (actual time=0.099..0.099 rows=1 loops=1)
Heap Fetches: 1
Planning Time: 0.232 ms
Trigger for constraint fk_purchase_ticket: time=1.278 calls=1
Trigger for constraint fk_purchase_user: time=0.358 calls=1
Execution Time: 1393.154 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Ticket_Purchase"
    SET purchase_amount = 1800.00
    WHERE qr_code = 'QR-TEST-CODE-003';
```

QUERY PLAN
Update on "Ticket_Purchase" (cost=0.56..8.58 rows=0 width=0) (actual time=0.266..0.267 rows=0 loops=1)
-> Index Scan using "Ticket_Purchase_qr_code_key" on "Ticket_Purchase" (cost=0.56..8.58 rows=1 width=10) (actual time=0.117..0.119 rows=1 loops=1)
Index Cond: ((qr_code)::text = 'QR-TEST-CODE-003'::text)
Planning Time: 0.188 ms
Execution Time: 0.364 ms

Овој поглед има критично време на извршување од над 5 минути поради обработка на милиони трансакции. Индексите на `ticket_id` го намалуваат времето за 99%, овозможувајќи моментален преглед на приходите и рефундациите за секој настан.

```sql
-- index for linking ticket purchases to the specific tickets
CREATE INDEX idx_ticket_purchase_ticket_id ON
"Ticket_Purchase"(ticket_id);

-- index for the LEFT JOIN with refunds to calculate net revenue
accurately
CREATE INDEX idx_ticket_refund_purchase_id ON
"Ticket_Refund"(purchase_id);

-- index for linking tickets to scheduled events
CREATE INDEX idx_ticket_event_happening_id ON
"Ticket"(event_happening_id);
```

2. **Со индекс**

- **SELECT**

```sql
EXPLAIN ANALYZE
    SELECT * FROM "Event_Financial_Summary" WHERE event_id = 1;
```

QUERY PLAN
Finalize GroupAggregate (cost=298955.06..298975.08 rows=5 width=71) (actual time=1499.323..1508.016 rows=1 loops=1)
Group Key: eh.event_happening_id
-> Gather Merge (cost=298955.06..298974.89 rows=10 width=75) (actual time=1499.269..1507.962 rows=1 loops=1)
Workers Planned: 2
Workers Launched: 2
-> Partial GroupAggregate (cost=297955.04..297973.71 rows=5 width=75) (actual time=1463.528..1463.535 rows=0 loops=3)
Group Key: eh.event_happening_id
-> Sort (cost=297955.04..297957.70 rows=1064 width=79) (actual time=1463.377..1463.407 rows=388 loops=3)
Sort Key: eh.event_happening_id
Sort Method: quicksort Memory: 25kB
Worker 0: Sort Method: quicksort Memory: 25kB
Worker 1: Sort Method: quicksort Memory: 160kB
-> Nested Loop Left Join (cost=16033.48..297901.55 rows=1064 width=79) (actual time=1201.975..1463.060 rows=388 loops=3)
-> Parallel Hash Join (cost=16033.05..289477.99 rows=1064 width=67) (actual time=1201.913..1460.100 rows=388 loops=3)
Hash Cond: (tp.ticket_id = t.ticket_id)
-> Parallel Seq Scan on "Ticket_Purchase" tp (cost=0.00..248438.67 rows=6666668 width=20) (actual time=0.091..663.287 rows=5333334 loops=3)
-> Parallel Hash (cost=16000.08..16000.08 rows=2638 width=63) (actual time=13.806..13.809 rows=1442 loops=3)
Buckets: 8192 Batches: 1 Memory Usage: 512kB
-> Nested Loop (cost=1.01..16000.08 rows=2638 width=63) (actual time=30.720..39.526 rows=4325 loops=1)
-> Nested Loop (cost=0.57..1029.33 rows=3 width=55) (actual time=30.641..37.392 rows=5 loops=1)
-> Parallel Index Scan using "Event_Happening_pkey" on "Event_Happening" eh (cost=0.29..1004.39 rows=3 width=24) (actual time=30.536..37.242 rows=5 loops=1)
Filter: (event_id = 1)
Rows Removed by Filter: 31332
-> Index Scan using "Event_pkey" on "Event" e (cost=0.29..8.30 rows=1 width=39) (actual time=0.021..0.023 rows=1 loops=5)
Index Cond: (event_id = 1)
-> Index Scan using idx_ticket_event_happening_id on "Ticket" t (cost=0.44..4974.83 rows=1542 width=16) (actual time=0.052..0.324 rows=865 loops=5)
Index Cond: (event_happening_id = eh.event_happening_id)
-> Index Scan using idx_ticket_refund_purchase_id on "Ticket_Refund" tr (cost=0.43..7.92 rows=1 width=20) (actual time=0.007..0.007 rows=0 loops=1163)
Index Cond: (purchase_id = tp.purchase_id)
Planning Time: 1.604 ms
JIT:
Functions: 96
Options: Inlining false, Optimization false, Expressions true, Deforming true

Timing: Generation 8.629 ms (Deform 3.959 ms), Inlining 0.000 ms, Optimization 3.603 ms, Emission 79.354 ms, Total 91.585 ms
Execution Time: 1510.555 ms

- **INSERT**

```
EXPLAIN ANALYZE
    INSERT INTO "Ticket_Purchase" (purchase_id, ticket_id, user_id,
qr_code, purchase_time, purchase_amount)
    SELECT COALESCE(MAX(purchase_id), 0) + 1, 1, 2, 'QR-TEST-CODE-
004', CURRENT_TIMESTAMP, 1400.00
    FROM "Ticket_Purchase";
```

QUERY PLAN
Insert on "Ticket_Purchase" (cost=0.70..0.73 rows=0 width=0) (actual time=0.351..0.352 rows=0 loops=1)
-> Subquery Scan on "*SELECT*" (cost=0.70..0.73 rows=1 width=552) (actual time=0.145..0.147 rows=1 loops=1)
-> Result (cost=0.70..0.71 rows=1 width=88) (actual time=0.132..0.133 rows=1 loops=1)
InitPlan 1
-> Limit (cost=0.43..0.70 rows=1 width=8) (actual time=0.126..0.127 rows=1 loops=1)
-> Index Only Scan Backward using "Ticket_Purchase_pkey" on "Ticket_Purchase" "Ticket_Purchase_1" (cost=0.43..4162966.51 rows=16000002 width=8) (actual time=0.124..0.124 rows=1 loops=1)
Heap Fetches: 2
Planning Time: 0.208 ms
Trigger for constraint fk_purchase_ticket: time=0.225 calls=1
Trigger for constraint fk_purchase_user: time=0.176 calls=1
Execution Time: 0.802 ms

- **UPDATE**

```
EXPLAIN ANALYZE
    UPDATE "Ticket_Purchase"
    SET purchase_amount = 1700.00
    WHERE qr_code = 'QR-TEST-CODE-004';
```

QUERY PLAN
Update on "Ticket_Purchase" (cost=0.56..8.58 rows=0 width=0) (actual time=0.161..0.162 rows=0 loops=1)
-> Index Scan using "Ticket_Purchase_qr_code_key" on "Ticket_Purchase" (cost=0.56..8.58 rows=1 width=10) (actual time=0.115..0.116 rows=1 loops=1)
Index Cond: ((qr_code)::text = 'QR-TEST-CODE-004'::text)
Planning Time: 0.134 ms
Execution Time: 0.195 ms