

## **Case2 – COUNT Query Bottleneck in Pagination**

**Verification Evidence PDF**


case2\_get\_posts.yml ×

```
1  config:
2    target: "http://localhost:8008"
3    phases:
4      - duration: 120          # 2분
5        arrivalRate: 4        # 초당 4명 시작(RPS를 올리기 위한 진입 부하)
6    defaults:
7      headers:
8        Content-Type: "application/json"
9
10   scenarios:
11     - name: "Case2 - GET /api/posts paging"
12       flow:
13         - get:
14             url: "/api/posts?page=0&size=20&sort=id,desc"
15             expect:
16               - statusCode: 200
```

Artillery Test Scenario: case2\_get\_posts.yml

## Query Definition

// 7) [case4-1] 목록 조회 전용 DTO ( N+1 제거 목적 )

@Query( 1개 사용 위치  cw01483-ly

```
value =
    "select new com.example.demo.domain.post.dto.PostListResponseDto(" +
        "    p.id, " +
        "    p.displayNumber, " +
        "    p.title, " +
        "    p.content, " +
        "    p.views, " +
        "    a.nickname, " +
        "    p.createdAt, " +
        "    p.updatedAt, " +
        "    coalesce(count(pl.id), 0L) " +
        ") " +
    "from Post p " +
    "join p.author a " +
    "left join com.example.demo.domain.post.entity.PostLike pl on pl.post = p " +
    "group by " +
    "    p.id, p.displayNumber, p.title, p.content, p.views, " +
    "    a.nickname, p.createdAt, p.updatedAt " +
    "order by p.id desc",
countQuery = "select count(distinct p.id) " +
    "from Post p left join com.example.demo.domain.post.entity.PostLike pl on pl.post = p"
)

Page<PostListResponseDto> findPostListWithLikeCount(Pageable pageable);
```

Paging Query and Associated COUNT Query  
used in Spring Data Pageable Retrieval

Database Index State (Before)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigation

SCHEMAS

Filter objects

demo

- Tables
  - comments
  - post\_likes
  - posts
  - users
- Views
- Stored Procedures
- Functions

- sys

Query 1

SQL File 3\*

SQL File 4\*

posts

Limit to 1000 rows

SHOW INDEX FROM posts;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	...
posts	0	PRIMARY	1	id	A	91079				BTREE			YES	NO
posts	1	idx_posts_user_id	1	user_id	A	4				BTREE			YES	NO
posts	1	idx_posts_title	1	title	A	87984				BTREE			YES	NO

Administration

Schemas

No object selected

Result 5

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	23:16:27	SHOW INDEX FROM posts	3 row(s) returned	0.000 sec / 0.000 sec

Index configuration of posts table before index creation

Execution Plan (Before)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

demo

- Tables
  - comments
  - post\_likes
  - posts
  - users
- Views
- Stored Procedures
- Functions

sys

Query 1

SQL File 3\* SQL File 4\* posts

Limit to 1000 rows

1 EXPLAIN  
2 SELECT  
3   COUNT(DISTINCT p1\_0.id)  
4 FROM posts p1\_0  
5 LEFT JOIN post\_likes pl1\_0  
6   ON pl1\_0.post\_id = p1\_0.id  
7 WHERE pl1\_0.is\_deleted = 0;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	p1_0	NULL	ALL	NULL	NULL	NULL	NULL	91079	50.00	Using where
	1	SIMPLE	pl1_0	NULL	ref	uk_post_likes_post_user,idx_post_likes_post_id	idx_post_likes_post_id	8	demo.p1_0.id	10	100.00	Using index

Administration

Schemas

No object selected

Result 1

Read Only

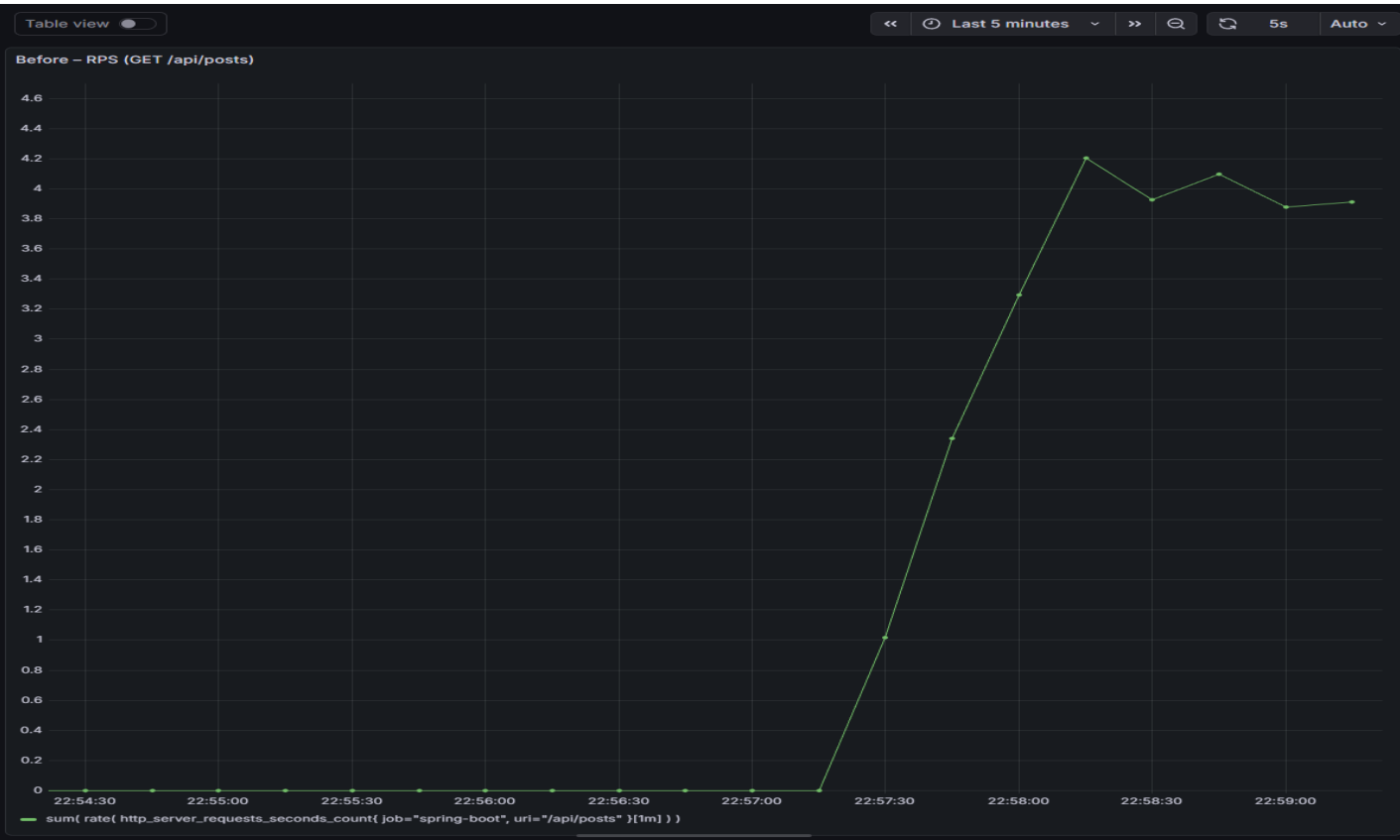
Output

Action Output

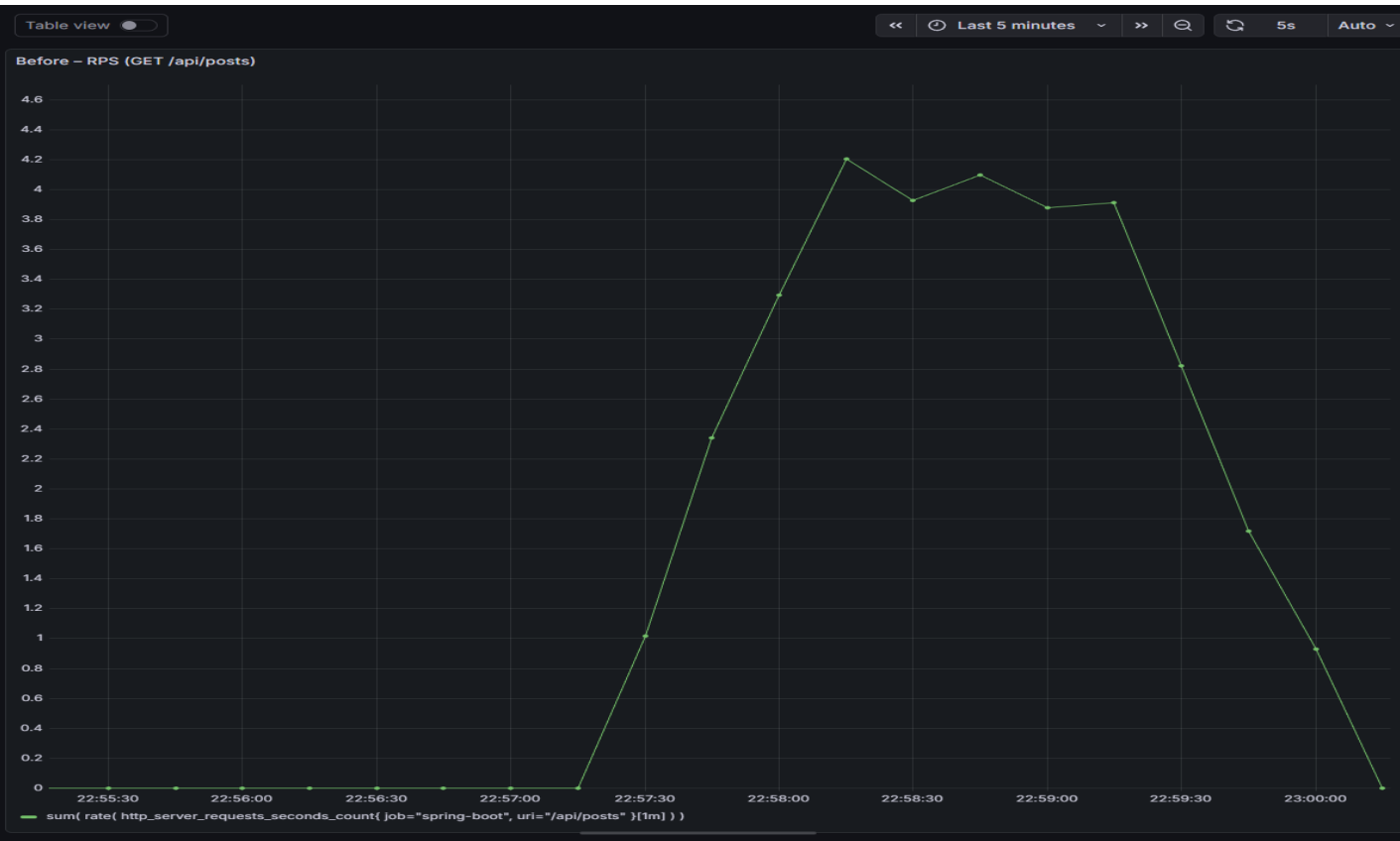
#	Time	Action	Message	Duration / Fetch
✓ 1	23:16:27	SHOW INDEX FROM posts	3 row(s) returned	0.000 sec / 0.000 sec
✓ 2	23:19:01	EXPLAIN SELECT   COUNT(DISTINCT p1_0.id) FROM posts p1_0 LEFT JOIN p...	2 row(s) returned	0.000 sec / 0.000 sec

Execution plan of COUNT query before index creation

# Performance Metrics (RPS) — Before



RPS During Load Test Execution (Before)



RPS After Test Completion (Before)

# Index Creation (Composite Index on posts)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

demo

Tables

comments

post\_likes

posts

users

Views

Stored Procedures

Functions

sys

Administration

Schemas

No object selected

Query 1 SQL File 3\* SQL File 4\* posts

Limit to 1000 rows

1 •

2

CREATE INDEX idx\_posts\_is\_deleted\_id ON posts (is\_deleted, id);

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	23:16:27	SHOW INDEX FROM posts	3 row(s) returned	0.000 sec / 0.000 sec
✓ 2	23:19:01	EXPLAIN SELECT COUNT(DISTINCT p1_0.id) FROM posts p1_0 LEFT JOIN p...	2 row(s) returned	0.000 sec / 0.000 sec
✓ 3	23:20:08	CREATE INDEX idx_posts_is_deleted_id ON posts (is_deleted, id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	3.375 sec

Creation of Composite Index (is\_deleted, id) on posts Table

Database Index State (After)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

demo

Tables

comments

post\_likes

posts

users

Views

Stored Procedures

Functions

sys

Administration Schemas

Information

No object selected

Query 1 SQL File 3\* SQL File 4\* posts

Limit to 1000 rows

1 SHOW INDEX FROM posts;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
posts	0	PRIMARY	1	id	A	91079				BTREE			YES
posts	1	idx_posts_user_id	1	user_id	A	4				BTREE			YES
posts	1	idx_posts_title	1	title	A	87984				BTREE			YES
posts	1	idx_posts_is_deleted_id	1	is_deleted	A	1				BTREE			YES
posts	1	idx_posts_is_deleted_id	2	id	A	91079				BTREE			YES

Result 6

Output

#	Time	Action	Message	Duration / Fetch
1	23:16:27	SHOW INDEX FROM posts	3 row(s) returned	0.000 sec / 0.000 sec
2	23:19:01	EXPLAIN SELECT COUNT(DISTINCT p1_0.id) FROM posts p1_0 LEFT JOIN p...	2 row(s) returned	0.000 sec / 0.000 sec
3	23:20:08	CREATE INDEX idx_posts_is_deleted_id ON posts (is_deleted, id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	3.375 sec
4	23:21:28	SHOW INDEX FROM posts	5 row(s) returned	0.000 sec / 0.000 sec

Index configuration of posts table after index creation

Execution Plan (After)

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

Filter objects

demo

Tables

comments

post\_likes

posts

users

Views

Stored Procedures

Functions

sys

Query 1

SQL File 3\* SQL File 4\* posts

Limit to 1000 rows

1

EXPLAIN

2

SELECT

3

COUNT(DISTINCT p1\_0.id)

4

FROM posts p1\_0

5

LEFT JOIN post\_likes p11\_0

6

ON p11\_0.post\_id = p1\_0.id

7

WHERE p1\_0.is\_deleted = 0;

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Result Grid

Form Editor

Field Types

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	p1_0	NULL	ref	idx_posts_is_deleted_id	idx_posts_is_deleted_id	1	const	45539	100.00	Using index
	1	SIMPLE	p11_0	NULL	ref	uk_post_likes_post_user,idx_post_likes_post_id	idx_post_likes_post_id	8	demo.p1_0.id	10	100.00	Using index

Administration Schemas

Information

No object selected

Result 2

Read Only

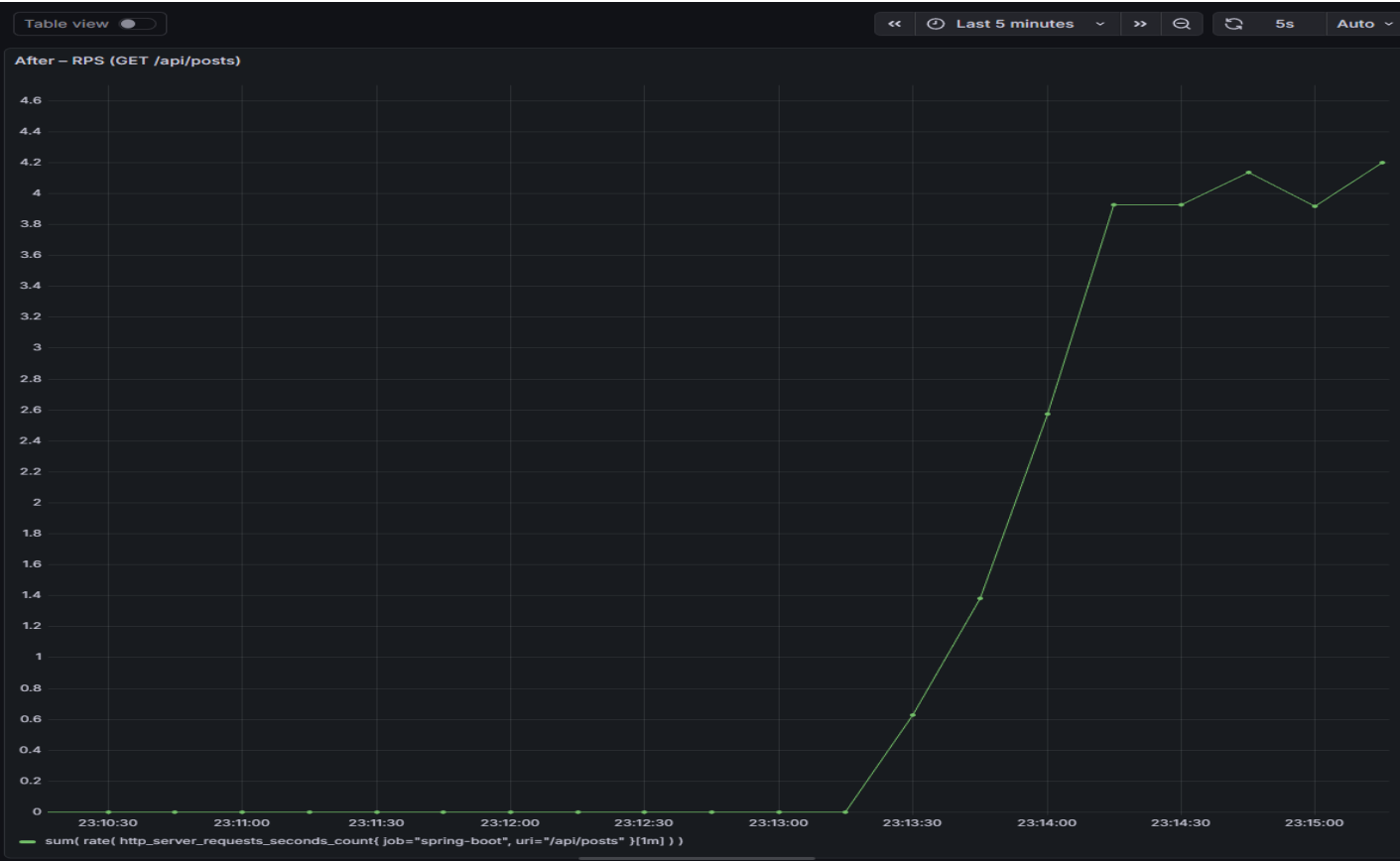
Output

Action Output

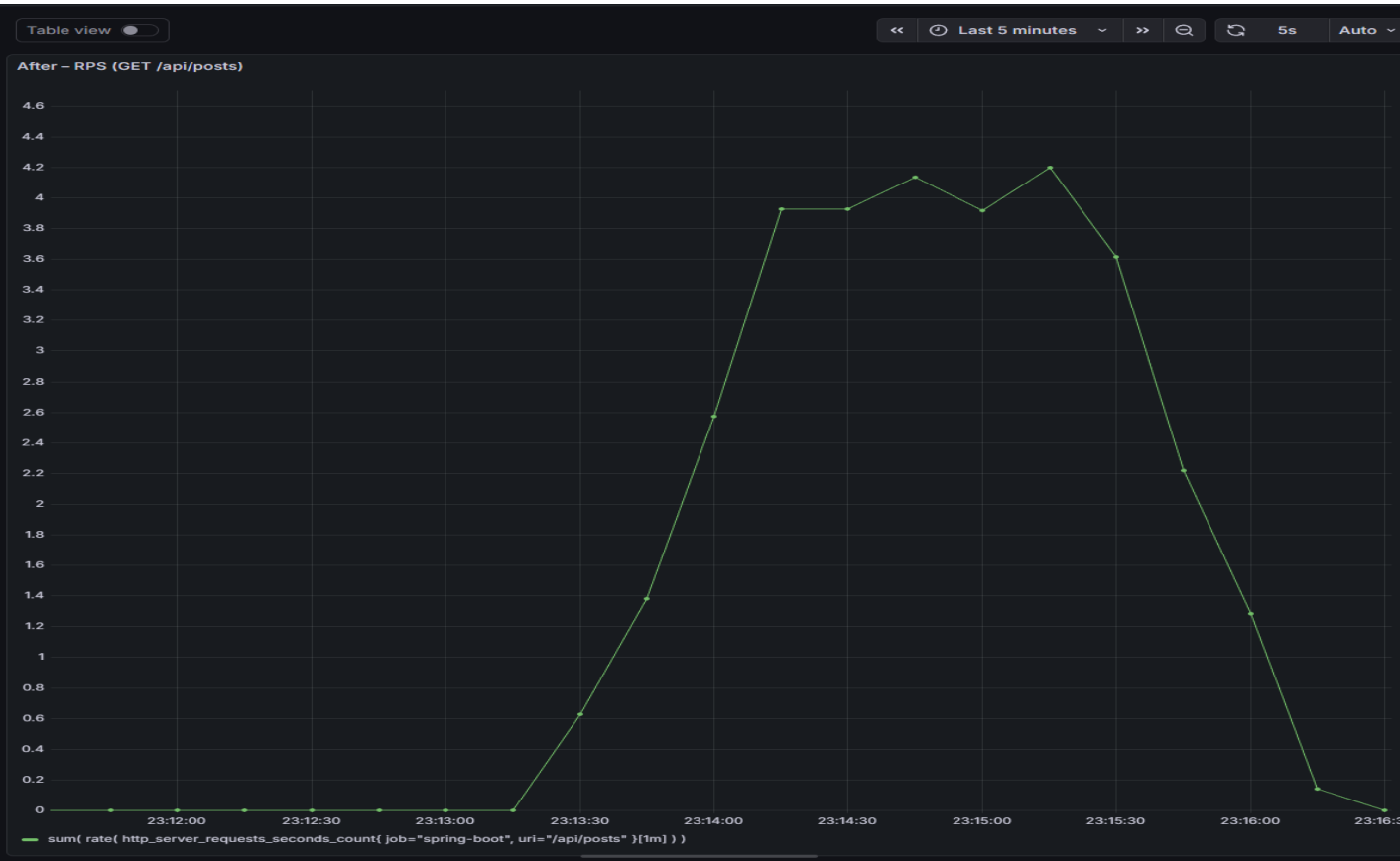
#	Time	Action	Message	Duration / Fetch
✓ 1	23:16:27	SHOW INDEX FROM posts	3 row(s) returned	0.000 sec / 0.000 sec
✓ 2	23:19:01	EXPLAIN SELECT COUNT(DISTINCT p1_0.id) FROM posts p1_0 LEFT JOIN p...	2 row(s) returned	0.000 sec / 0.000 sec
✓ 3	23:20:08	CREATE INDEX idx_posts_is_deleted_id ON posts (is_deleted, id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	3.375 sec
✓ 4	23:21:28	SHOW INDEX FROM posts	5 row(s) returned	0.000 sec / 0.000 sec
✓ 5	23:23:08	EXPLAIN SELECT COUNT(DISTINCT p1_0.id) FROM posts p1_0 LEFT JOIN p...	2 row(s) returned	0.000 sec / 0.000 sec

Execution plan of COUNT query after index creation

# Performance Metrics (RPS) — After



RPS During Load Test Execution (After)



RPS After Test Completion (After)

SQL Execution Log (Before / After)

```
Hibernate:
select
  p1_0.id,
  p1_0.display_number,
  p1_0.title,
  p1_0.content,
  p1_0.views,
  a1_0.nickname,
  p1_0.created_at,
  p1_0.updated_at,
  coalesce(count(pl1_0.id), 0)
from
  posts p1_0
join
  users a1_0
  on a1_0.id=p1_0.user_id
  and (a1_0.is_deleted = 0)
left join
  post_likes pl1_0
  on pl1_0.post_id=p1_0.id
where
  (
    p1_0.is_deleted = 0
  )
group by
  p1_0.id,
  p1_0.display_number,
  p1_0.title,
  p1_0.content,
  p1_0.views,
  a1_0.nickname,
  p1_0.created_at,
  p1_0.updated_at
order by
  p1_0.id desc
limit
  ?
2026-01-02T21:37:24.091+09:00 TRACE 22812 --- [nio-8008-exec-2] org.hibernate.orm.jdbc.bind : binding parameter (1:INTEGER) <- [20]
2026-01-02T21:37:24.891+09:00 DEBUG 22812 --- [nio-8008-exec-2] org.hibernate.SQL :
select
  count(distinct p1_0.id)
from
  posts p1_0
left join
  post_likes pl1_0
  on pl1_0.post_id=p1_0.id
where
  (
    p1_0.is_deleted = 0
  )
Hibernate:
select
  count(distinct p1_0.id)
from
  posts p1_0
```

COUNT query execution log before index creation

```
Hibernate:
select
  p1_0.id,
  p1_0.display_number,
  p1_0.title,
  p1_0.content,
  p1_0.views,
  a1_0.nickname,
  p1_0.created_at,
  p1_0.updated_at,
  coalesce(count(pl1_0.id), 0)
from
  posts p1_0
join
  users a1_0
  on a1_0.id=p1_0.user_id
  and (a1_0.is_deleted = 0)
left join
  post_likes pl1_0
  on pl1_0.post_id=p1_0.id
where
  (
    p1_0.is_deleted = 0
  )
group by
  p1_0.id,
  p1_0.display_number,
  p1_0.title,
  p1_0.content,
  p1_0.views,
  a1_0.nickname,
  p1_0.created_at,
  p1_0.updated_at
order by
  p1_0.id desc
limit
  ?
2026-01-04T21:31:11.347+09:00 TRACE 26916 --- [nio-8008-exec-5] org.hibernate.orm.jdbc.bind : binding parameter (1:INTEGER) <- [20]
2026-01-04T21:31:12.188+09:00 DEBUG 26916 --- [nio-8008-exec-5] org.hibernate.SQL :
select
  count(distinct p1_0.id)
from
  posts p1_0
left join
  post_likes pl1_0
  on pl1_0.post_id=p1_0.id
where
  (
    p1_0.is_deleted = 0
  )
Hibernate:
select
  count(distinct p1_0.id)
from
  posts p1_0
```

COUNT query execution log after index creation

## Artillery Result Comparison

Metric	Before (No Index)	After (is_deleted, id Index)	Reduction (%)
HTTP 200 Responses	480	480	0.00%
Requests	480	480	0.00%
Errors	0	0	0.00%
Min Response Time (ms)	1040.0	1019.0	+2.02%
Mean Response Time (ms)	1246.4	1179.8	+5.34%
Median Response Time (ms)	1224.4	1153.1	+5.82%
p95 Response Time (ms)	1436.8	1380.5	+3.92%
p99 Response Time (ms)	1556.5	1525.7	+1.98%
Max Response Time (ms)	1615.0	1606.0	+0.56%

Table: Artillery latency summary (http.response\_time)

```
All VUs finished. Total time: 2 minutes, 2 seconds

-----
Summary report @ 04:30:01(+0900)
-----

http.codes.200: ..... 480
http.downloaded_bytes: ..... 2463360
http.request_rate: ..... 4/sec
http.requests: ..... 480
http.response_time:
  min: ..... 1040
  max: ..... 1615
  mean: ..... 1246.4
  median: ..... 1224.4
  p95: ..... 1436.8
  p99: ..... 1556.5
http.response_time.2xx:
  min: ..... 1040
  max: ..... 1615
  mean: ..... 1246.4
  median: ..... 1224.4
  p95: ..... 1436.8
  p99: ..... 1556.5
http.responses: ..... 480
vusers.completed: ..... 480
vusers.created: ..... 480
vusers.created_by_name.Case2 - GET /api/posts paging: ..... 480
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 1043.2
  max: ..... 1617.7
  mean: ..... 1253.2
  median: ..... 1224.4
  p95: ..... 1465.9
  p99: ..... 1556.5
```

Artillery summary before index creation

```
All VUs finished. Total time: 2 minutes, 2 seconds

-----
Summary report @ 04:37:28(+0900)
-----

http.codes.200: ..... 480
http.downloaded_bytes: ..... 2463360
http.request_rate: ..... 4/sec
http.requests: ..... 480
http.response_time:
  min: ..... 1019
  max: ..... 1606
  mean: ..... 1179.8
  median: ..... 1153.1
  p95: ..... 1380.5
  p99: ..... 1525.7
http.response_time.2xx:
  min: ..... 1019
  max: ..... 1606
  mean: ..... 1179.8
  median: ..... 1153.1
  p95: ..... 1380.5
  p99: ..... 1525.7
http.responses: ..... 480
vusers.completed: ..... 480
vusers.created: ..... 480
vusers.created_by_name.Case2 - GET /api/posts paging: ..... 480
vusers.failed: ..... 0
vusers.session_length:
  min: ..... 1022
  max: ..... 1633
  mean: ..... 1185
  median: ..... 1153.1
  p95: ..... 1380.5
  p99: ..... 1525.7
```

Artillery summary after index creation

- **Result:**  
No significant, user-visible performance change.
- **Observation:**  
COUNT query remains the primary bottleneck.
- **Validation:**  
SQL Log / EXPLAIN / Artillery / Grafana

**Status: Limited Impact, Bottleneck Confirmed**