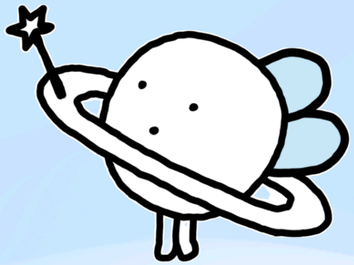
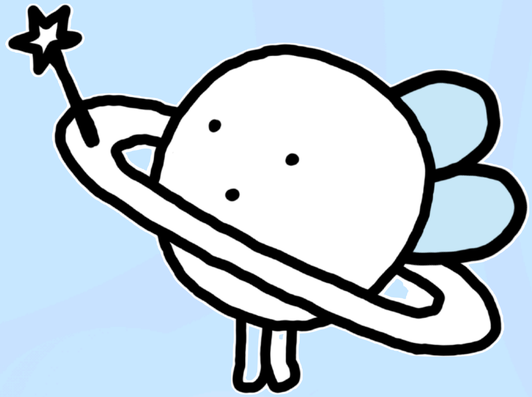
쿼리 튜닝으로
218배 빨라진 팬 점유율 API



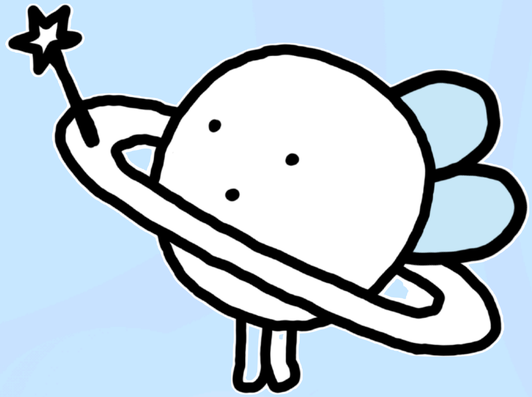
BE 7기 밍트

개요



1. 배경
2. 병목 쿼리 분석 & 개선
3. 결과
4. 마무리

개요



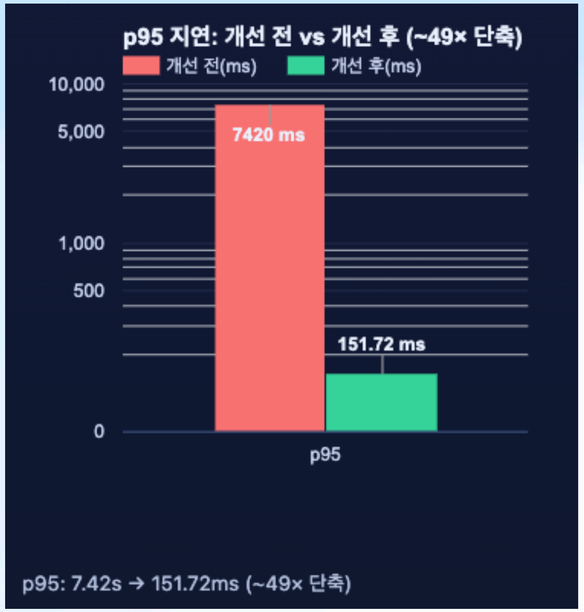
1. 배경
2. 병목 쿼리 분석 & 개선
3. 결과
4. 마무리

1. 배경

개선 전/후 성능 그래프 (응답속도, p95, p99, RPS)

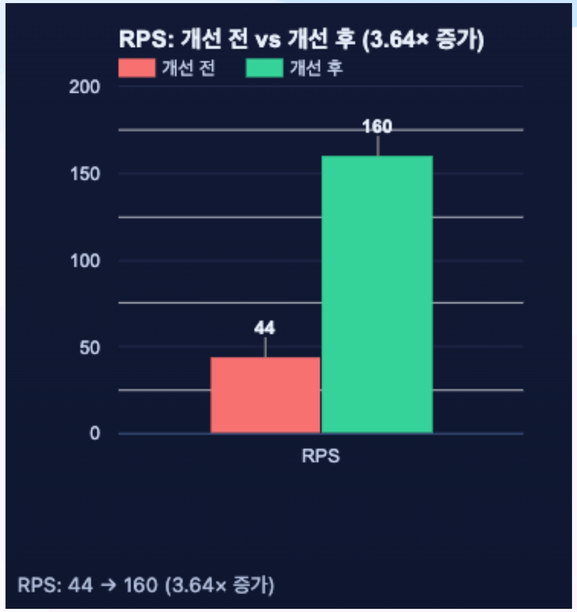
평균 응답 속도

P95



P99

RPS



1. 배경

팬 점유율 통계 성능 측정하기

테스트 데이터

→ 날짜 : 2013-01-01

→ 구장 : 기아 챔피언스 필드

→ 직관 인증 회원 수 : 30,031명

부하 측정 도구 : k6

→ RPS를 고정한 상태에서 P95, P99 측정 가능

1. 배경

팬 점유율 통계 성능 측정하기

Request per Second : 서버가 1초에 처리하는 요청 개수

목표 RPS : 160

→ 1초에 160명이 동시에 요청해도 버벅임없이 처리

Service Level Agreement : 서비스 성능 약속 기준

SLA : p95 < 300ms, p99 < 600ms

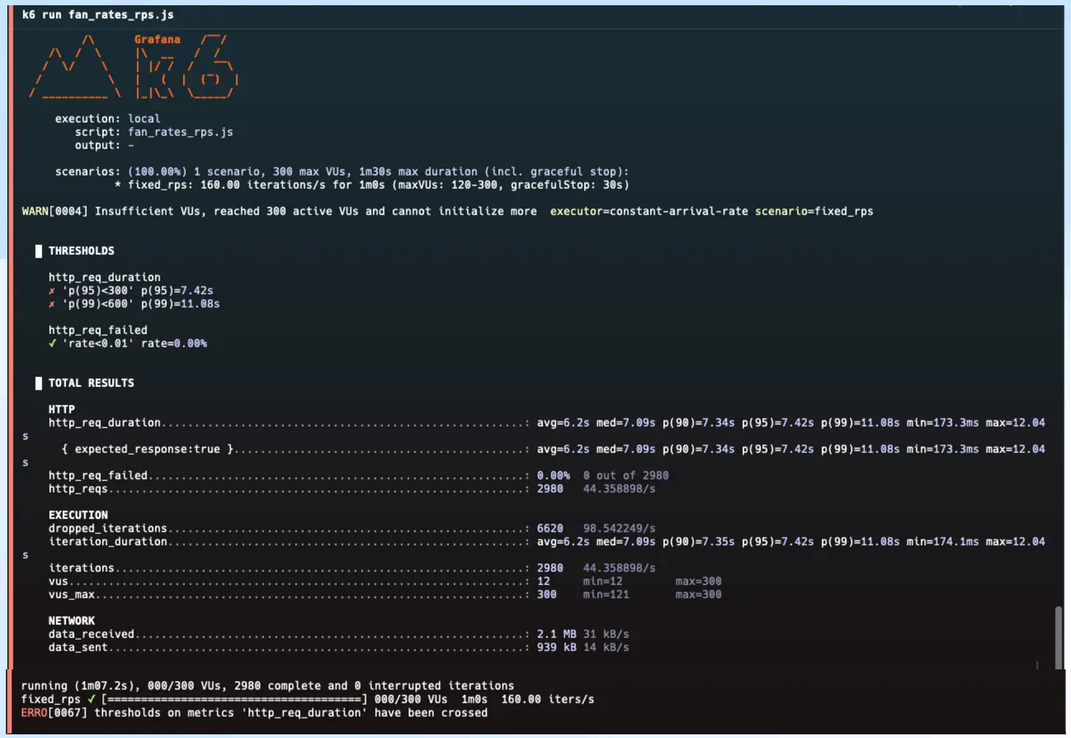
🎯 P95 : 전체 요청 중 가장 느린 5%를 제외한 나머지 95%가 걸린 시간

P99 : 가장 느린 1%를 제외한 나머지 99%가 걸린 시간

😀 UX 기준 즉시 반응 시간 : 0.1~0.3초

1. 배경

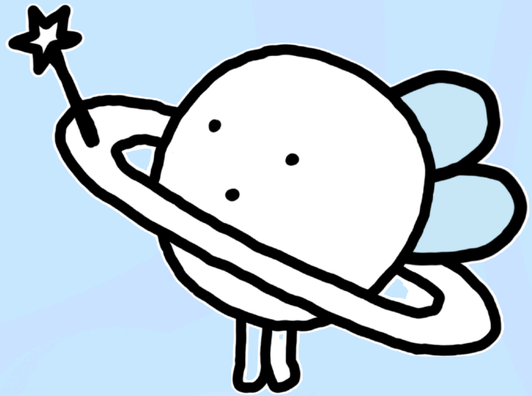
성능 측정 결과 : 목표 미충족



```
k6 run fan_rates_rps.js

     /\      Grafana   /‾/
  /\/  \     |\__  /  /
 /  \   \    |  |/ /  /
/ ‾‾‾‾‾‾\   |  ( |  (‾)
/_______\ \  |__|\__\ \___/

    execution: local
       script: fan_rates_rps.js
       output: -

    scenarios: (100.00%) 1 scenario, 300 max VUs, 1m30s max duration (incl. graceful stop):
             * fixed_rps: 160.00 iterations/s for 1m0s (maxVUs: 120-300, gracefulStop: 30s)

WARN[0004] Insufficient VUs, reached 300 active VUs and cannot initialize more  executor=constant-arrival-rate scenario=fixed_rps


  ▉ THRESHOLDS

    http_req_duration
    ✗ 'p(95)<300' p(95)=7.42s
    ✗ 'p(99)<600' p(99)=11.08s

    http_req_failed
    ✓ 'rate<0.01' rate=0.00%


  ▉ TOTAL RESULTS

    HTTP
    http_req_duration.....................................................: avg=6.2s med=7.09s p(90)=7.34s p(95)=7.42s p(99)=11.08s min=173.3ms max=12.04
s
      { expected_response:true }..........................................: avg=6.2s med=7.09s p(90)=7.34s p(95)=7.42s p(99)=11.08s min=173.3ms max=12.04
s
    http_req_failed.......................................................: 0.00%  0 out of 2980
    http_reqs.............................................................: 2980   44.358898/s

    EXECUTION
    dropped_iterations....................................................: 6620   98.542249/s
    iteration_duration....................................................: avg=6.2s med=7.09s p(90)=7.35s p(95)=7.42s p(99)=11.08s min=174.1ms max=12.04
s
    iterations............................................................: 2980   44.358898/s
    vus...................................................................: 12     min=12        max=300
    vus_max...............................................................: 300    min=121       max=300

    NETWORK
    data_received.........................................................: 2.1 MB 31 kB/s
    data_sent.............................................................: 939 kB 14 kB/s


running (1m07.2s), 000/300 VUs, 2980 complete and 0 interrupted iterations
fixed_rps ✓ [======================================] 000/300 VUs  1m0s  160.00 iters/s
ERRO[0067] thresholds on metrics 'http_req_duration' have been crossed
```

항목	목표치	실제 결과	상태
RPS	160	~44 RPS	목표 미달
p95 지연	< 300ms	7.42s	초과 (~25배)
p99 지연	< 600ms	11.08s	초과 (~18배)
dropped_iterations	0	6,620	발생
VU 사용량	-	300 (최대)	Insufficient VUs 경고
실패율	< 1%	0.00%	정상

개요



1. 배경
2. 병목 쿼리 분석 & 개선
3. 결과
4. 마무리

2. 병목 쿼리 분석

병목 쿼리 찾기

멤버 조회 → **날짜별 게임 목록 조회** → **팬 카운트 집계** → 팀 메타 조회 2건

Type	Text	Duration
SQL / User scrip	SELECT ROUND(TIMESTAMPDIFF(MICROSECOND, @t0, NOW(6))/1000, 2) AS total_ms_server¶LI	0.275s
SQL / User scrip	-- 7) 원정 팀 메타 조회¶SELECT t.team_id, t.`name`, t.short_name, t.team_code¶FROM teams t¶WH	0.142s
SQL / User scrip	-- 6) 홈 팀 메타 조회¶SELECT t.team_id, t.`name`, t.short_name, t.team_code¶FROM teams t¶WHEI	0.109s
SQL / User scrip	-- 5) 팬 카운트 집계¶explain SELECT¶¶COUNT(ci.check_ins_id) AS total_fans,¶¶COALESCE(SUM(C.	0.183s
SQL / User scrip	-- 4) 선택한 게임의 홈/원정 팀 ID 변수에 담기¶SELECT g.home_team_id, g.away_team_id¶INTO @home	0.000s
SQL / User scrip	-- 3) 날짜의 첫 번째 game_id 선택¶SET @gameId := (¶ SELECT g.game_id¶ FROM games g¶ WHER	0.001s
SQL / User scrip	-- 2) 해당 날짜 게임 목록¶SELECT¶ g.game_id,¶ g.away_score,¶ g.away_bases_on_balls,¶ g.a	0.220s
SQL / User scrip	-- 1) 멤버 단건 조회 (is_deleted = 0)¶SELECT¶ m.member_id,¶ m.email,¶ m.image_url,¶ m.is_	0.122s

2. 병목 쿼리 분석

A) 날짜별 게임 목록 조회 쿼리

나쁜 쿼리

```sql
SELECT g.game_id, g.home_team_id, g.away_team_id, g.date
FROM games g
WHERE g.date = @date
ORDER BY g.game_id;
```

PK 순서로 모든 행을 읽고 날짜로 필터링

→ 스캔 범위가 너무 큼

→ order by 에서 date 활용 불가

나쁜 실행 계획

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	g	[NULL]	index	[NULL]	PRIMARY	8	[NULL]	23,100	10	Using where

```
-> Filter: (g.`date` = ((@`date`)))  (cost=2350 rows=2310) (actual time=0.487..28.3 rows=5 loops=1)
    -> Index scan on g using PRIMARY  (cost=2350 rows=23100) (actual time=0.484..26.6 rows=22980 loops=1)
```

평균적으로 20~28ms대에서 마무리

2. 병목 쿼리 분석

A) 날짜별 게임 목록 조회 쿼리

인덱스 생성

```
CREATE INDEX idx_games_date ON games(date);
```

날짜 인덱스 생성

→ 해당 날짜의 행들만 스캔 가능

→ order by game_id 추가 정렬 필요

개선된 실행 계획

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	g	[NULL]	ref	idx_games_date	idx_games_date	3	const	5	100	[NULL]

```
-> Index lookup on g using idx_games_date (date=(@`date`))  (cost=1.75 rows=5)
   (actual time=0.0936..0.098 rows=5 loops=1)
```

평균적으로 0.09ms 에서 마무리 (약 222배 (99.6%) 성능 향상)

2. 병목 쿼리 분석

B) 구장의 팬 점유율 조회

나쁜 쿼리

```
SELECT
  COUNT(ci.check_ins_id) AS total_fans,
  SUM(CASE WHEN ci.team_id = @homeTeamId THEN 1 ELSE 0 END) AS home_fans,
  SUM(CASE WHEN ci.team_id = @awayTeamId THEN 1 ELSE 0 END) AS away_fans
FROM check_ins ci
WHERE ci.game_id = @gameId;
```

인덱스에 조회/집계에 필요한 컬럼이 없어서 테이블 조회

북마크 룩업 발생

→ 집계에 필요한 team_id 없음

→ 매 행마다 테이블 페이지 재조회 발생 (3만건)

나쁜 실행 계획

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	ci	[NULL]	ref	FKnisshsxfvs6ppe2qtoc1jy9g6	FKnisshsxfvs6ppe2qtoc1jy9g6	8	const	60,816	100	[NULL]

```
-> Aggregate: sum((case when (ci.team_id = (@awayTeamId)) then 1 else 0 end)),
              sum((case when (ci.team_id = (@homeTeamId)) then 1 else 0 end)),
              count(ci.check_ins_id)
              (cost=18243 rows=1) (actual time=48.8..48.8 rows=1 loops=1)
    -> Index lookup on ci using FKnisshsxfvs6ppe2qtoc1jy9g6 (game_id=(@gameId))
    (cost=12162 rows=60816) (actual time=7.24..42 rows=30041 loops=1)
```

평균적으로 48.8ms 에서 마무리

2. 병목 쿼리 분석

B) 구장의 팬 점유율 조회

커버링 인덱스 생성

```
SELECT
  COUNT(ci.check_ins_id) AS total_fans,
  SUM(CASE WHEN ci.team_id = @homeTeamId THEN 1 ELSE 0 END) AS home_fans,
  SUM(CASE WHEN ci.team_id = @awayTeamId THEN 1 ELSE 0 END) AS away_fans
FROM check_ins ci
WHERE ci.game_id = @gameId;
```

쿼리에서 필요로 하는 모든 컬럼이 인덱스에 존재

커버링 인덱스 사용

→ 필요한 데이터가 전부 인덱스에 존재

→ 테이블 접근이 사라져 시간 단축

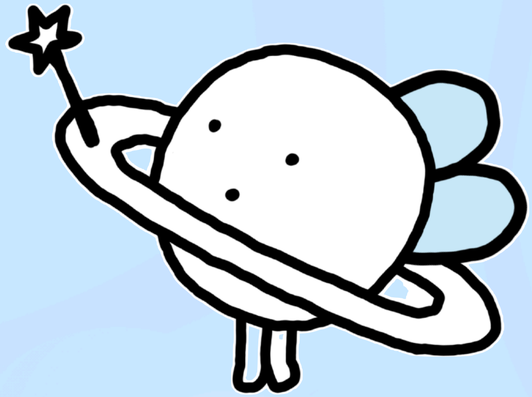
개선된 실행 계획

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	1	SIMPLE	ci	[NULL]	ref	idx_checkins_game_team	idx_checkins_game_team	8	const	61,904	100	Using index

```
-> Aggregate: count(0),
sum((case when (ci.team_id = (@homeTeamId)) then 1 else 0 end)),
sum((case when (ci.team_id = (@awayTeamId)) then 1 else 0 end))
  (cost=12563 rows=1) (actual time=26.3..26.3 rows=1 loops=1)
    -> Covering index lookup on ci using idx_checkins_game_team (game_id=(@gameId))
    (cost=6372 rows=61904) (actual time=7.18..17.8 rows=30041 loops=1)
```

평균적으로 26.3ms 에서 마무리 (약 1.8배, 46% 성능 향상)

개요



1. 배경
2. 병목 쿼리 분석 & 개선
3. **결과**
4. 마무리

3. 결과

쿼리 개선 결과

날짜별 게임 목록

구분	인덱스	스캔 행 수	실행 시간 (ms)	개선폭
개선 전	PK(전체 스캔)	23,100	0.487 ~ 28.3	-
개선 후	idx_games_date(date)	5	0.0936 ~ 0.098	~288배

문제 : PK 전체 스캔 → 23,100 행 읽음 → 28ms

개선 : date 인덱스 적용 → 5행만 읽음 → 0.09ms

효과 : ~288배 속도 개선 (풀 스캔 → 인덱스 룩업)

3. 결과

쿼리 개선 결과

구장의 팬 점유율 조회

구분	인덱스	스캔 행 수	실행 시간 (ms)	개선폭
개선 전	(game_id) 단일 인덱스	30,041	48.8	-
개선 후	(game_id, team_id) 커버링 + COUNT(*)	30,041	26.3	~1.86배 (46% 향상)

문제 : game_id 단일 인덱스 → team_id 조회시 북마크 룩업 발생 → 48.8ms 소요

개선 : game_id, team_id 커버링 인덱스 → 테이블 접근 제거 →26.6ms

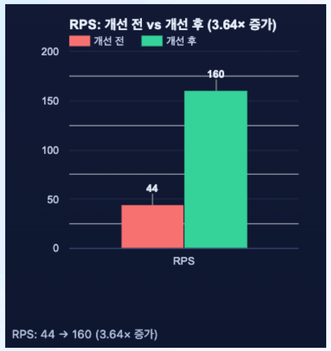
효과 : ~ 1.86배 속도 개선, 실행 시간 절반 단축

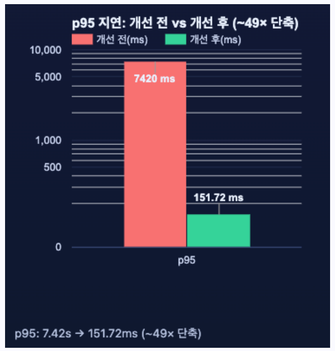
3. 결과

인덱스 적용 후 성능 개선 효과

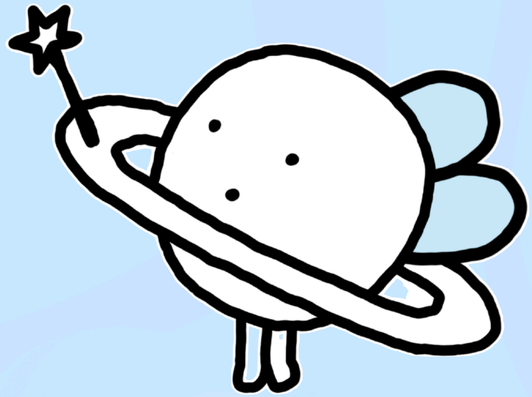
엔드투엔드

구분	인덱스 전	인덱스 후	변화
목표 RPS	160	160	동일
실제 처리량	~44 rps (2980/67.2s)	160 rps (9601/60s)	~3.64×↑
p95	7.42 s	151.72 ms	~49× 빨라짐
p99	11.08 s	301.83 ms	~37× 빨라짐
평균 지연	6.20 s	28.47 ms	~218× 빨라짐
실패율	0.00%	0.00%	동일 (정상)
dropped_iterations	6,620	0	드랍 소멸
VU 사용(실측)	최대 300, 경고 발생	최대 33 내외	자원 여유
임계치(p95<300, p99<600)	실패	모두 통과	✅





개요



1. 배경
2. 병목 쿼리 분석 & 개선
3. 결과
4. 마무리

결론

인덱스 최적화의 성능 개선 효과는 크다!

하지만 인덱스는 무조건 많이 만들면 안된다,,

성능 병목을 측정하고, 지속적으로 개선해나가자

꼿