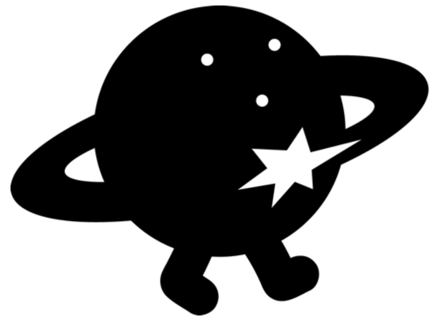
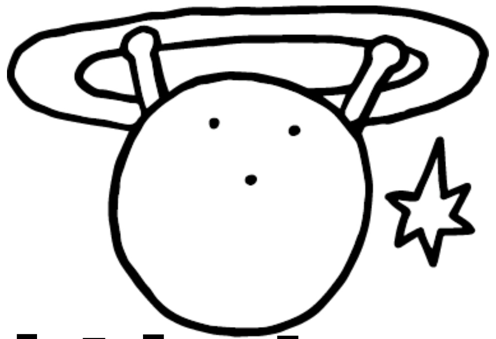
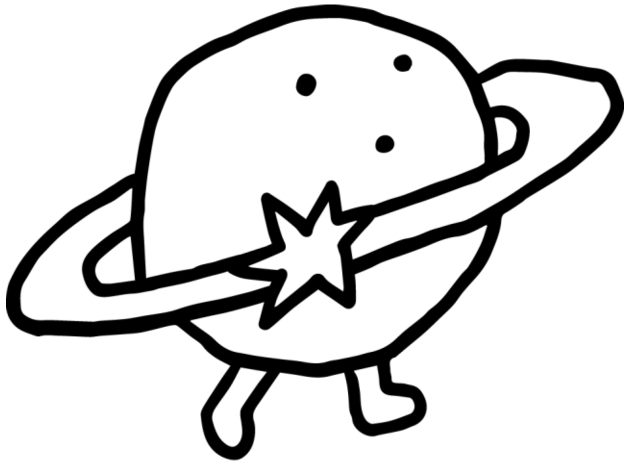
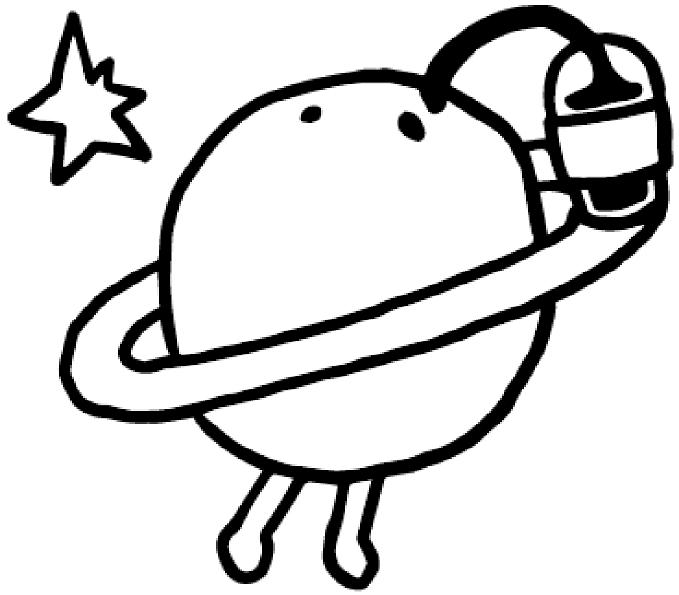



검색 기능 개선 실험하기

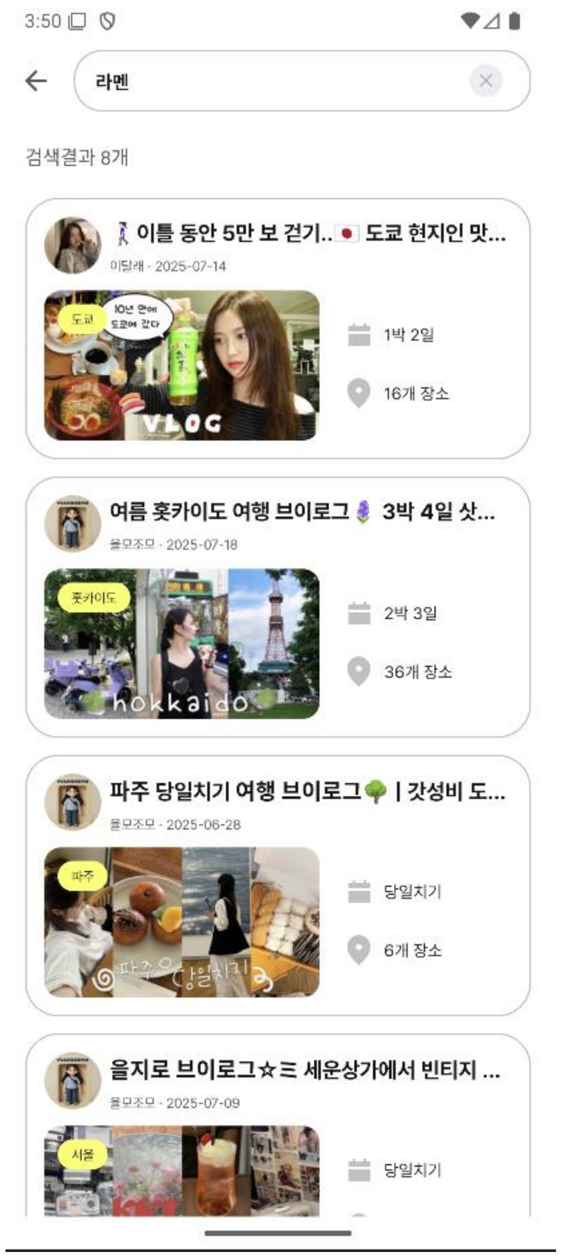
BE 7기 메이



현재 검색 기능
&
개선 필요한 부분



현재 검색 기능


3:50

라멘

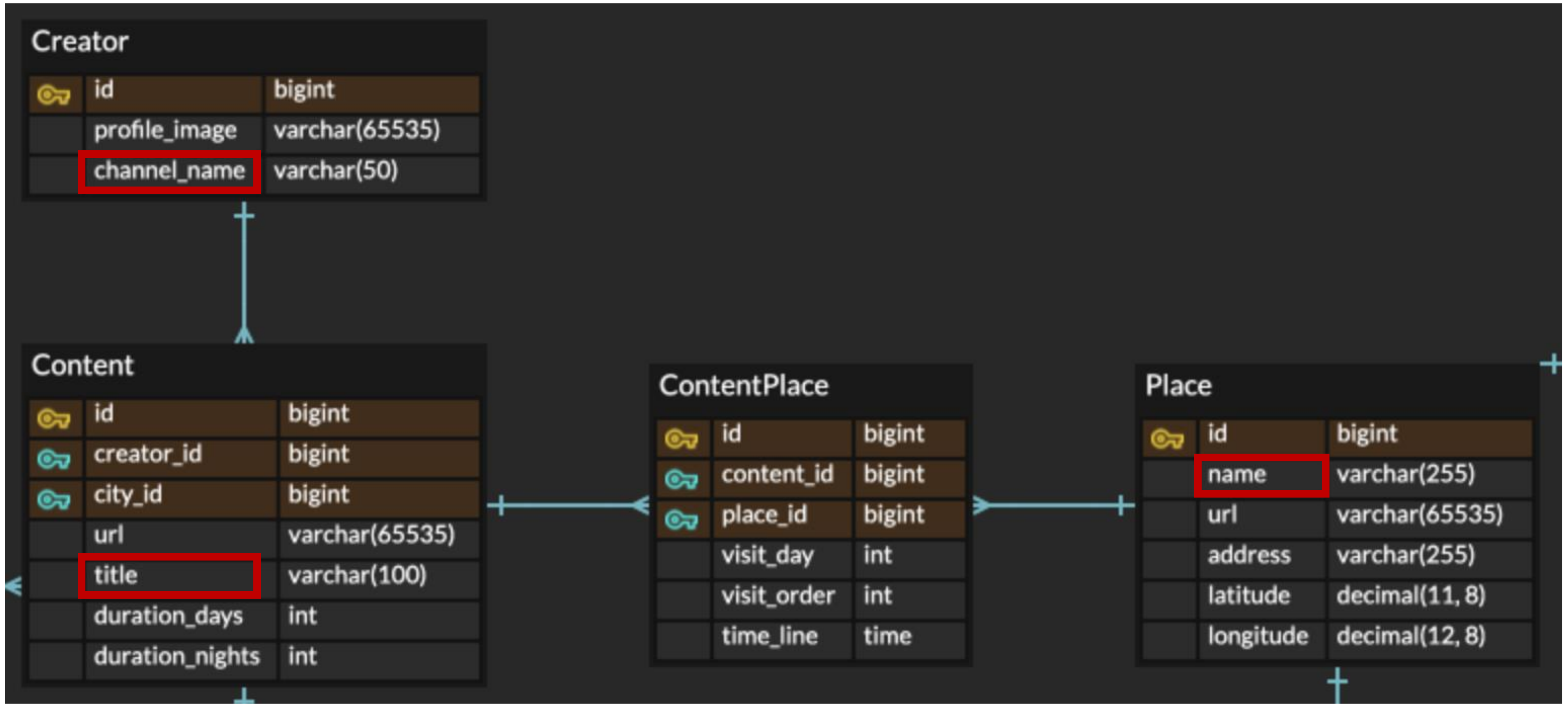
검색결과 8개

🚶 이틀 동안 5만 보 걷기..🇯🇵 도쿄 현지인 맛...
이달래 · 2025-07-14
1박 2일
16개 장소

여름 홋카이도 여행 브이로그 🪻 3박 4일 삿...
올모조모 · 2025-07-18
2박 3일
36개 장소

파주 당일치기 여행 브이로그🌳 | 갓성비 도...
올모조모 · 2025-06-28
당일치기
6개 장소

을지로 브이로그☆ミ 세운상가에서 빈티지 ...
올모조모 · 2025-07-09
당일치기


Creator

🔑	id	bigint
	profile_image	varchar(65535)
	channel_name	varchar(50)

Content

🔑	id	bigint
🔑	creator_id	bigint
🔑	city_id	bigint
	url	varchar(65535)
	title	varchar(100)
	duration_days	int
	duration_nights	int

ContentPlace

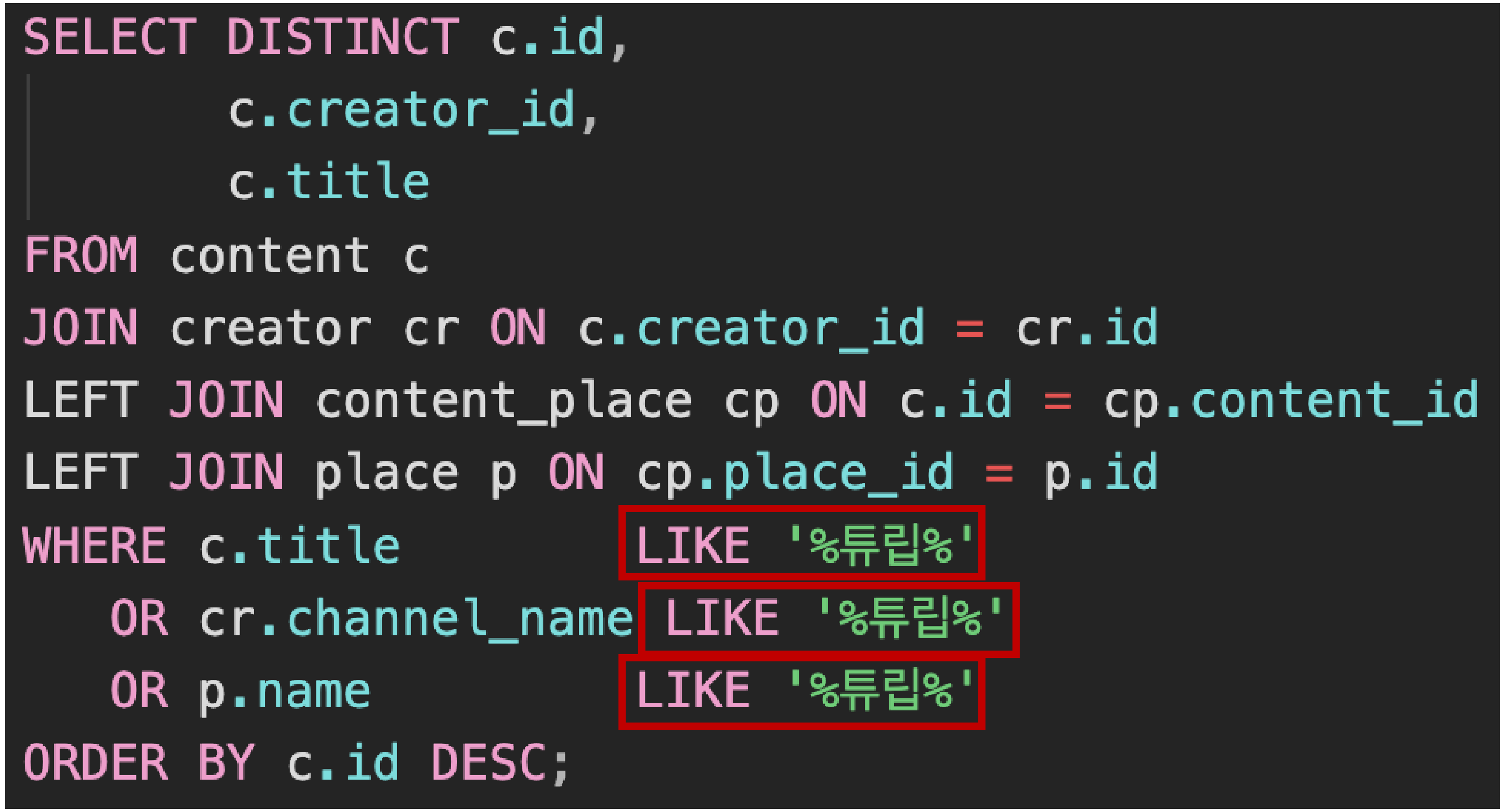
🔑	id	bigint
🔑	content_id	bigint
🔑	place_id	bigint
	visit_day	int
	visit_order	int
	time_line	time

Place

🔑	id	bigint
	name	varchar(255)
	url	varchar(65535)
	address	varchar(255)
	latitude	decimal(11, 8)
	longitude	decimal(12, 8)

- 컨텐츠(content) 테이블의 제목(title) 컬럼
- 크리에이터(creator) 테이블의 채널명(channel_name) 컬럼
- 장소(place) 테이블의 이름(name) 컬럼

현재 검색 기능


```sql
SELECT DISTINCT c.id,
       c.creator_id,
       c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
LEFT JOIN content_place cp ON c.id = cp.content_id
LEFT JOIN place p ON cp.place_id = p.id
WHERE c.title          LIKE '%튜립%'
   OR cr.channel_name LIKE '%튜립%'
   OR p.name           LIKE '%튜립%'
ORDER BY c.id DESC;
```

- LIKE %keyword% 쿼리 사용해서 필터링
- 단일 키워드 검색

개선이 필요한 부분

쿼리

현재 Full table scan

↓

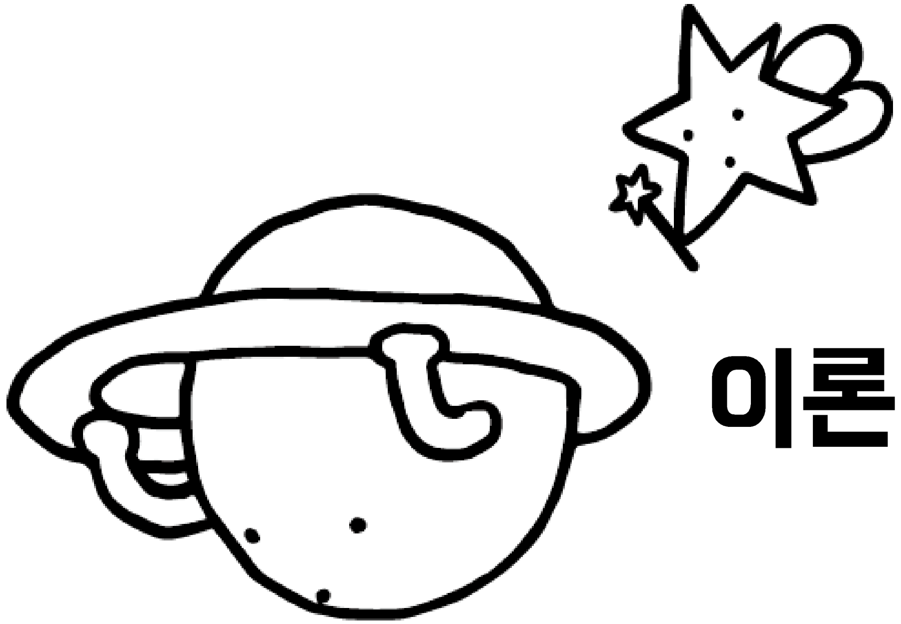
index

다중 키워드 검색 기능

”베트남 2박3일” -> “베트남”, “2박3일”과 일치하는 결과 원함

↓

fulltext index

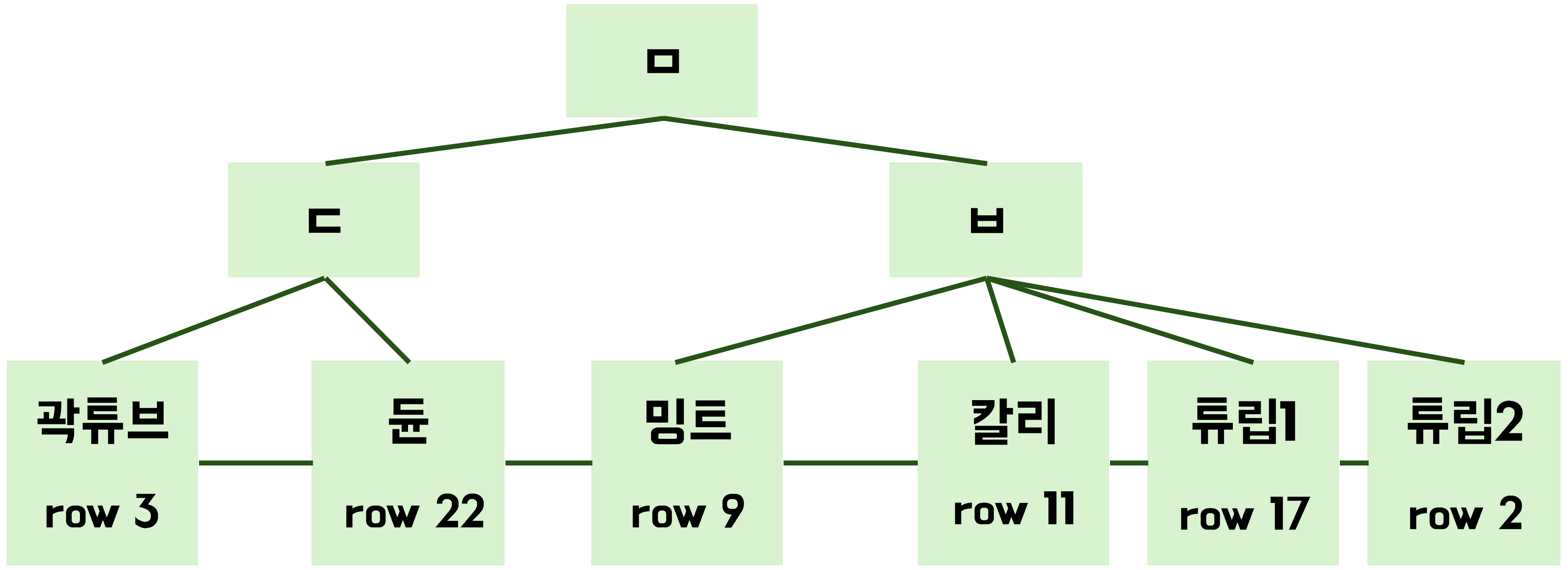


이론 공부

Index(MySQL)

- 특정 컬럼에 대한 탐색을 빠르게 하기 위한 도구
- B+Tree 구조로 저장
- 쓰기 성능이 저하될 수 있음

<Idx_creator_channel_name>



Fulltext Index(MySQL)

- 단어단위로 쪼개고, 그 단어들에 대한 역색인(Inverted index)을 만들어 관리
- WHERE MATCH … AGAINST …

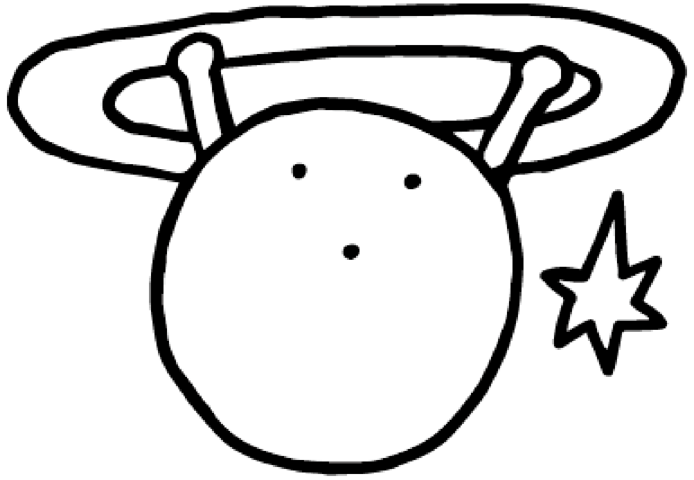
<ft_content_title>

"튜립 수원 행궁동 브이로그"가
3행에 저장

"튜립" -> [1행, 3행]
"수원" -> [3행, 5행]
"행궁동" -> [3행]
"브이로그" -> [1행, 2행, 3행, 5행]

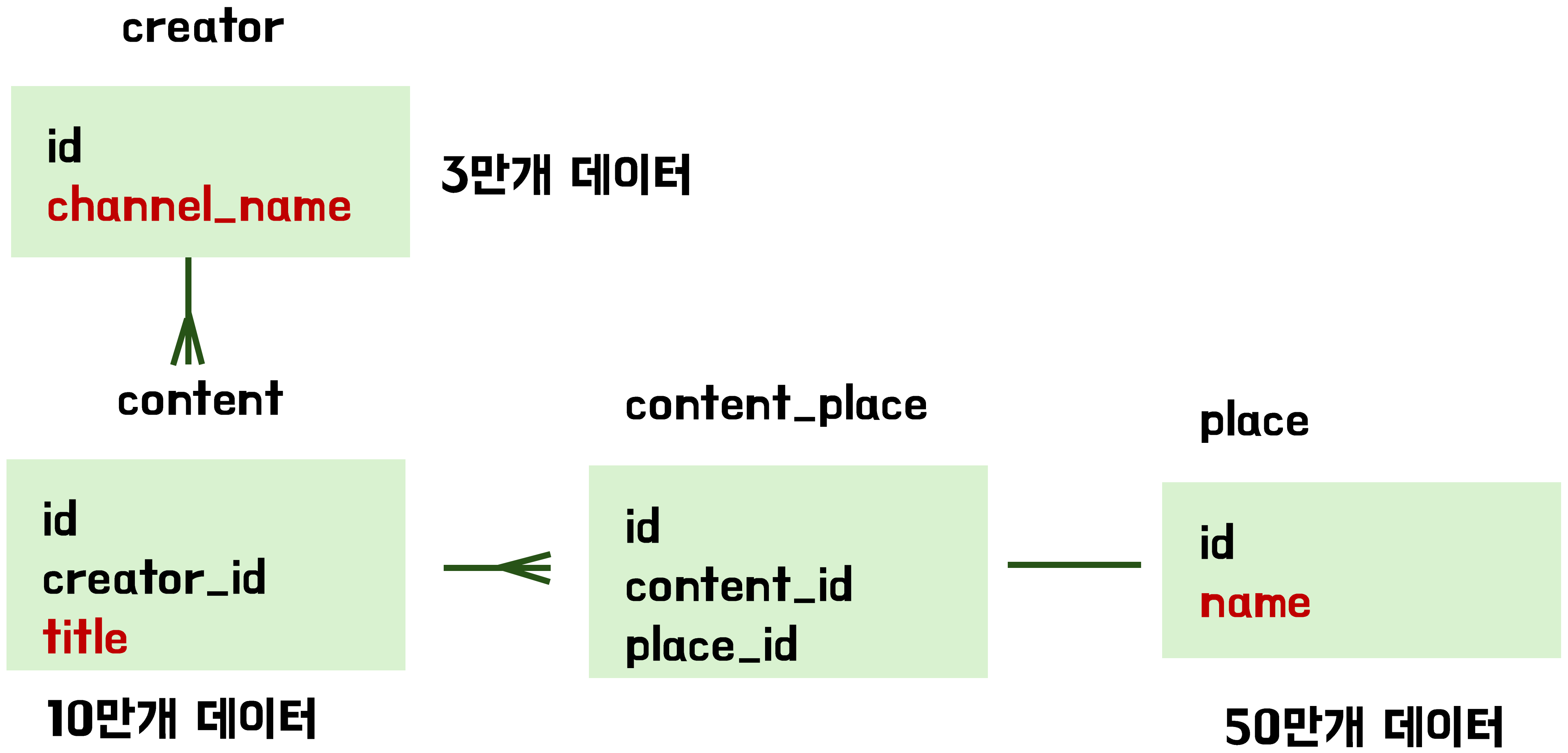
```
SELECT c.id, c.creator_id, c.title
FROM content c
WHERE MATCH(title) AGAINST('+수원 +브이로그' IN BOOLEAN MODE);
```

실험해보기

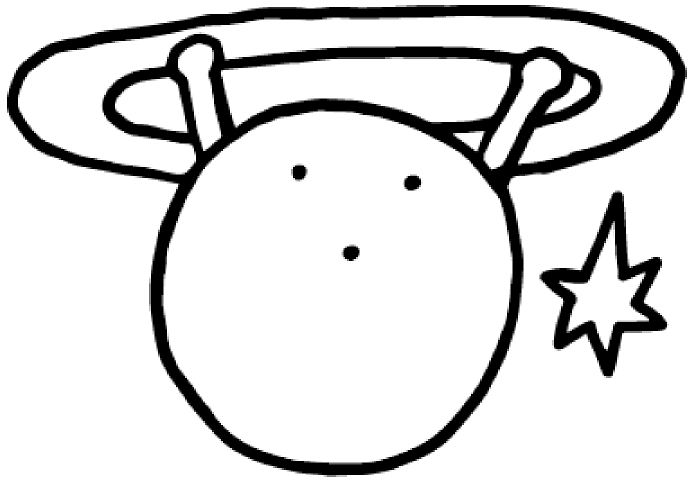


실험 환경

- 로컬 Docker container에 띄운 MySQL 사용
- 테스트용 DB에 검색 관련 테이블, 컬럼만 추가



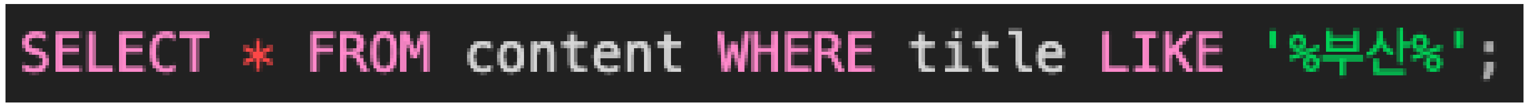
첫 번째 실험



인덱스 추가 - 탐색 속도 개선 실험

인덱스 추가했으니 LIKE %부산% 이 빨라지겠지?

Index 추가 전



```sql
SELECT * FROM content WHERE title LIKE '%부산%';
```

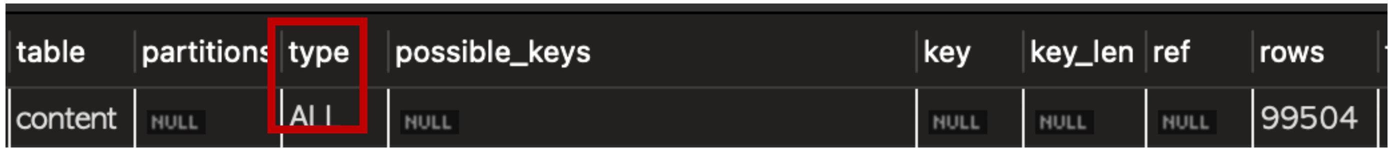
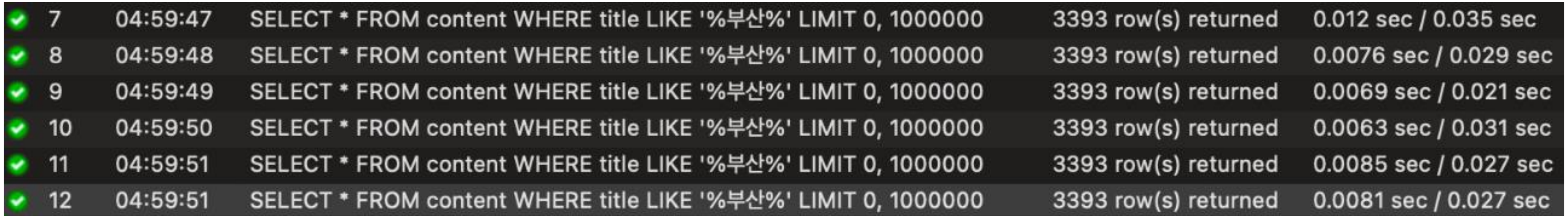


table	partitions	type	possible_keys	key	key_len	ref	rows
content	NULL	ALL	NULL	NULL	NULL	NULL	99504



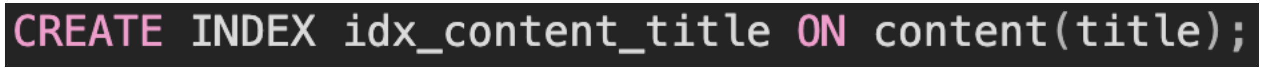
```
✓ 7    04:59:47   SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.012 sec / 0.035 sec
✓ 8    04:59:48   SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0076 sec / 0.029 sec
✓ 9    04:59:49   SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0069 sec / 0.021 sec
✓ 10   04:59:50   SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0063 sec / 0.031 sec
✓ 11   04:59:51   SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0085 sec / 0.027 sec
✓ 12   04:59:51   SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0081 sec / 0.027 sec
```

평균 7.48ms

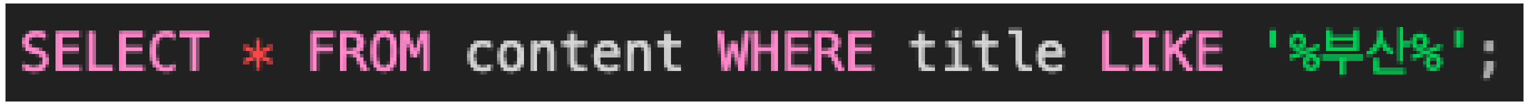
인덱스 추가 - 탐색 속도 개선 실험

인덱스 추가했으니 LIKE %부산% 이 빨라지겠지?

Index 추가 후



```
CREATE INDEX idx_content_title ON content(title);
```



```
SELECT * FROM content WHERE title LIKE '%부산%';
```

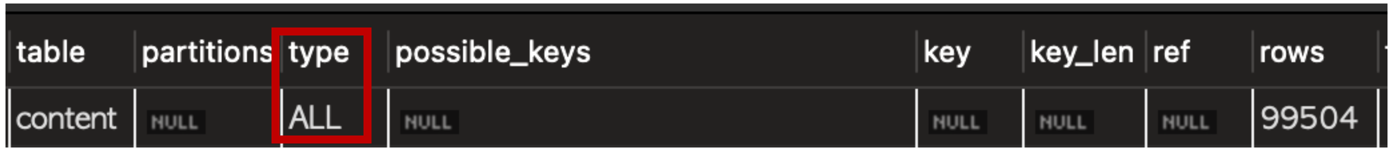
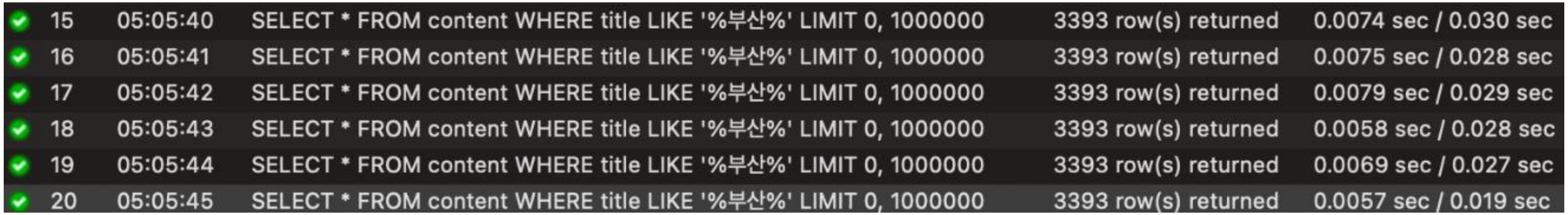


table	partitions	type	possible_keys	key	key_len	ref	rows
content	NULL	ALL	NULL	NULL	NULL	NULL	99504



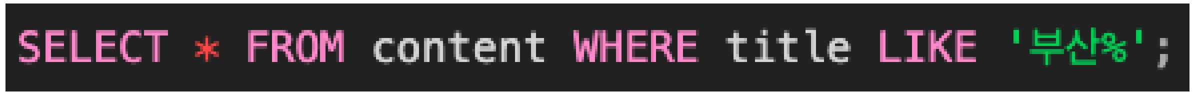
```
✓ 15  05:05:40  SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0074 sec / 0.030 sec
✓ 16  05:05:41  SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0075 sec / 0.028 sec
✓ 17  05:05:42  SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0079 sec / 0.029 sec
✓ 18  05:05:43  SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0058 sec / 0.028 sec
✓ 19  05:05:44  SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0069 sec / 0.027 sec
✓ 20  05:05:45  SELECT * FROM content WHERE title LIKE '%부산%' LIMIT 0, 1000000   3393 row(s) returned   0.0057 sec / 0.019 sec
```

평균 6.76ms
(인덱스를 활용하지 않음)

인덱스 추가 - 탐색 속도 개선 실험

인덱스 추가했을 때 LIKE 부산% 은 얼마나 빠를까?

Index 범위 탐색을 활용할 수 있는 경우



```sql
SELECT * FROM content WHERE title LIKE '부산%';
```

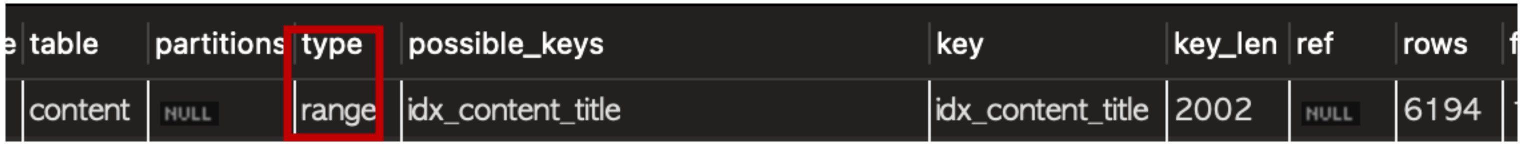
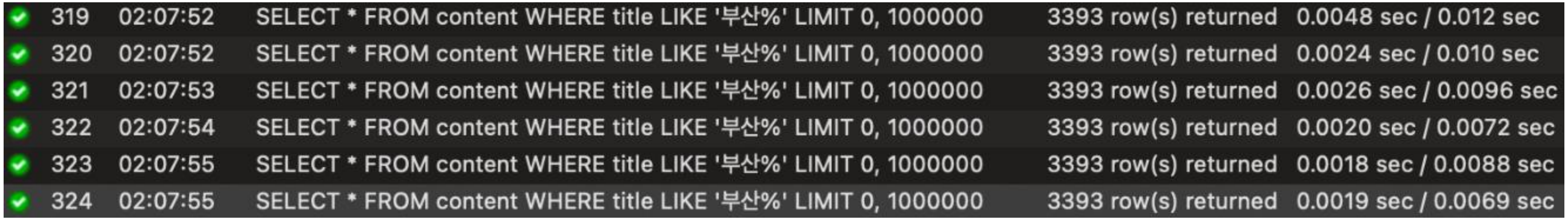


table	partitions	type	possible_keys	key	key_len	ref	rows
content	NULL	range	idx_content_title	idx_content_title	2002	NULL	6194



```
319  02:07:52  SELECT * FROM content WHERE title LIKE '부산%' LIMIT 0, 1000000  3393 row(s) returned  0.0048 sec / 0.012 sec
320  02:07:52  SELECT * FROM content WHERE title LIKE '부산%' LIMIT 0, 1000000  3393 row(s) returned  0.0024 sec / 0.010 sec
321  02:07:53  SELECT * FROM content WHERE title LIKE '부산%' LIMIT 0, 1000000  3393 row(s) returned  0.0026 sec / 0.0096 sec
322  02:07:54  SELECT * FROM content WHERE title LIKE '부산%' LIMIT 0, 1000000  3393 row(s) returned  0.0020 sec / 0.0072 sec
323  02:07:55  SELECT * FROM content WHERE title LIKE '부산%' LIMIT 0, 1000000  3393 row(s) returned  0.0018 sec / 0.0088 sec
324  02:07:55  SELECT * FROM content WHERE title LIKE '부산%' LIMIT 0, 1000000  3393 row(s) returned  0.0019 sec / 0.0069 sec
```

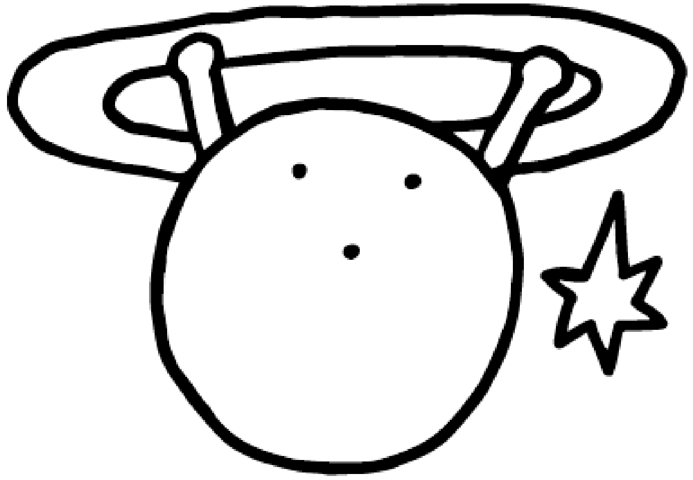
평균 2.14ms

인덱스 추가 - 탐색 속도 개선 실험 결과

LIKE %keyword% => 인덱스의 범위 탐색을 활용하지 못한다

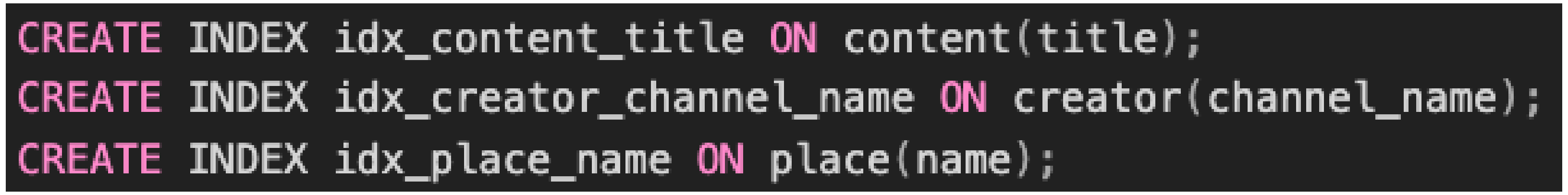
LIKE keyword% => 인덱스 범위 탐색을 활용할 수 있다 => 속도 차이 있다

두 번째 실험

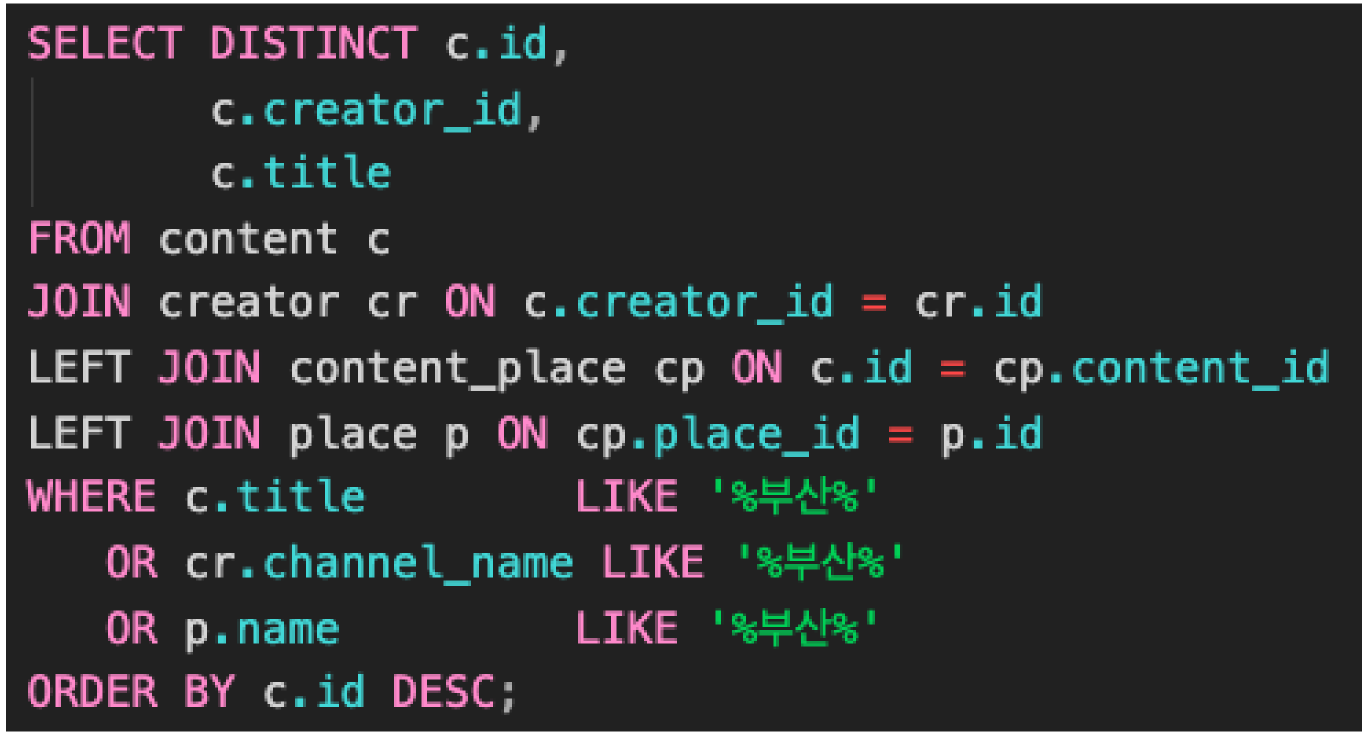


인덱스 추가 - 실제 쿼리 개선 실험

튜립 서비스 검색 쿼리도 인덱스 영향을 안 받을까?



```sql
CREATE INDEX idx_content_title ON content(title);
CREATE INDEX idx_creator_channel_name ON creator(channel_name);
CREATE INDEX idx_place_name ON place(name);
```

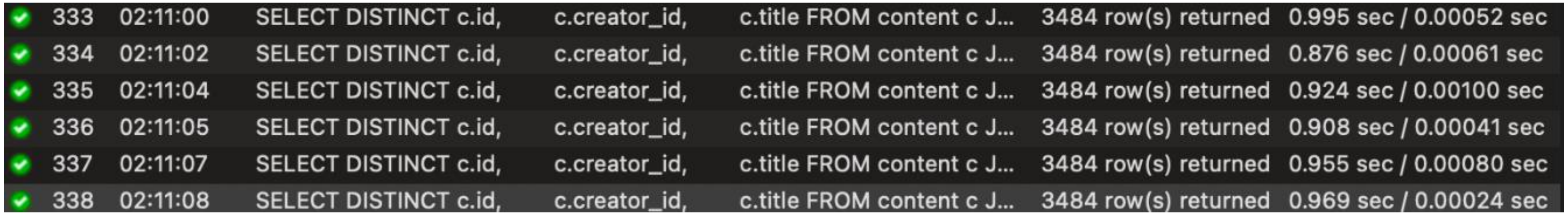


```sql
SELECT DISTINCT c.id,
       c.creator_id,
       c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
LEFT JOIN content_place cp ON c.id = cp.content_id
LEFT JOIN place p ON cp.place_id = p.id
WHERE c.title          LIKE '%부산%'
   OR cr.channel_name LIKE '%부산%'
   OR p.name          LIKE '%부산%'
ORDER BY c.id DESC;
```

인덱스 추가 - 실제 쿼리 개선 실험

튜립 서비스 검색 쿼리도 인덱스 영향을 안 받을까?

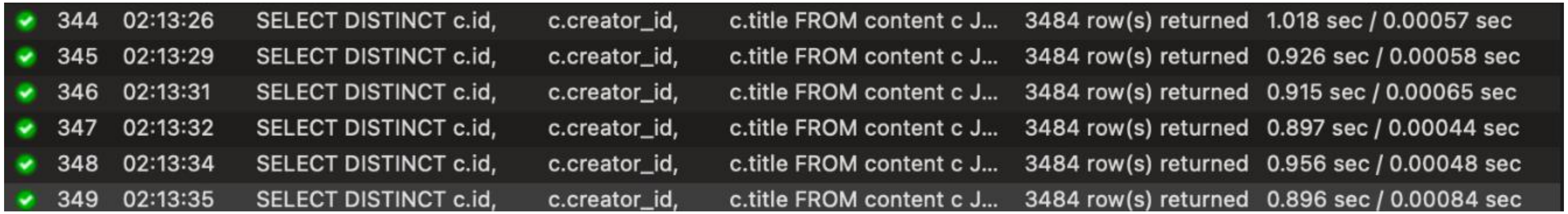
Index 추가 전



```
✓  333  02:11:00  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.995 sec / 0.00052 sec
✓  334  02:11:02  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.876 sec / 0.00061 sec
✓  335  02:11:04  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.924 sec / 0.00100 sec
✓  336  02:11:05  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.908 sec / 0.00041 sec
✓  337  02:11:07  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.955 sec / 0.00080 sec
✓  338  02:11:08  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.969 sec / 0.00024 sec
```

평균 0.9264sec

Index 추가 후

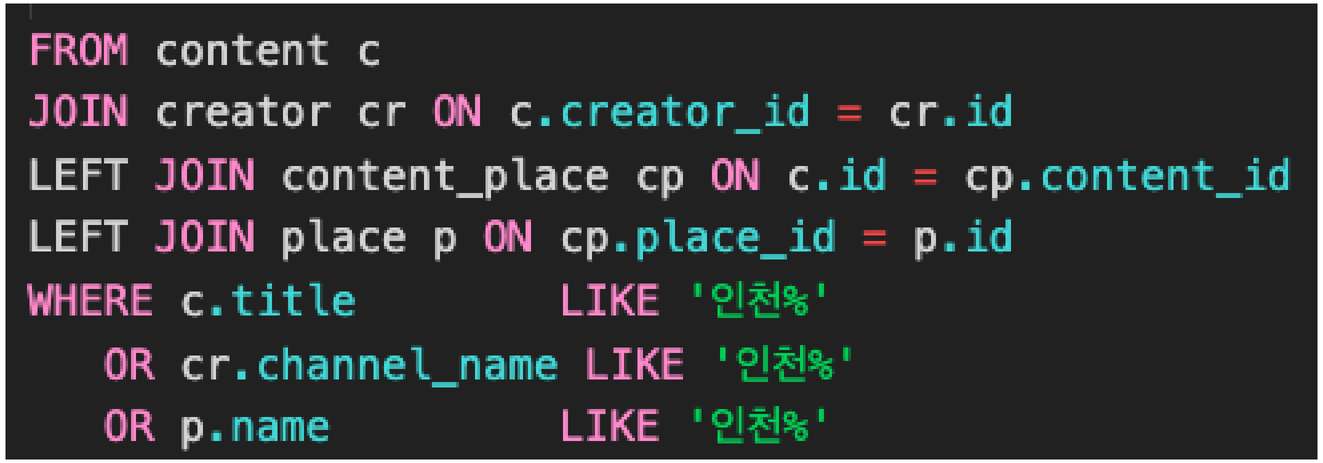


```
✓  344  02:13:26  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  1.018 sec / 0.00057 sec
✓  345  02:13:29  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.926 sec / 0.00058 sec
✓  346  02:13:31  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.915 sec / 0.00065 sec
✓  347  02:13:32  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.897 sec / 0.00044 sec
✓  348  02:13:34  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.956 sec / 0.00048 sec
✓  349  02:13:35  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.896 sec / 0.00084 sec
```

평균 0.918sec

인덱스 추가 - 실제 쿼리 개선 실험

LIKE 인천%은 많이 빨라지려나?



```
FROM content c
JOIN creator cr ON c.creator_id = cr.id
LEFT JOIN content_place cp ON c.id = cp.content_id
LEFT JOIN place p ON cp.place_id = p.id
WHERE c.title           LIKE '인천%'
   OR cr.channel_name LIKE '인천%'
   OR p.name            LIKE '인천%'
```

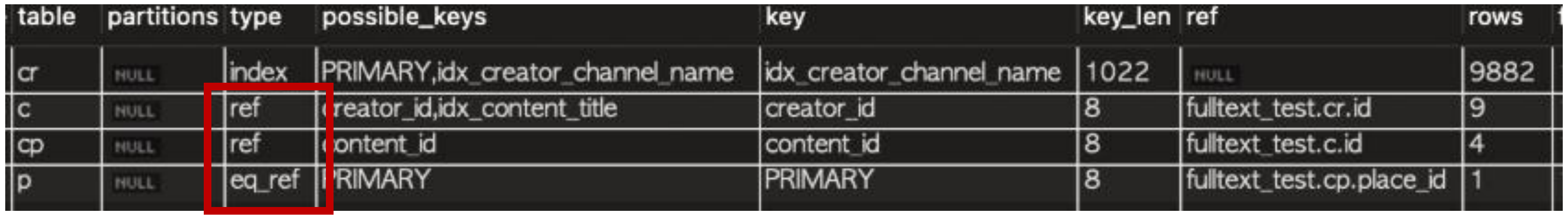
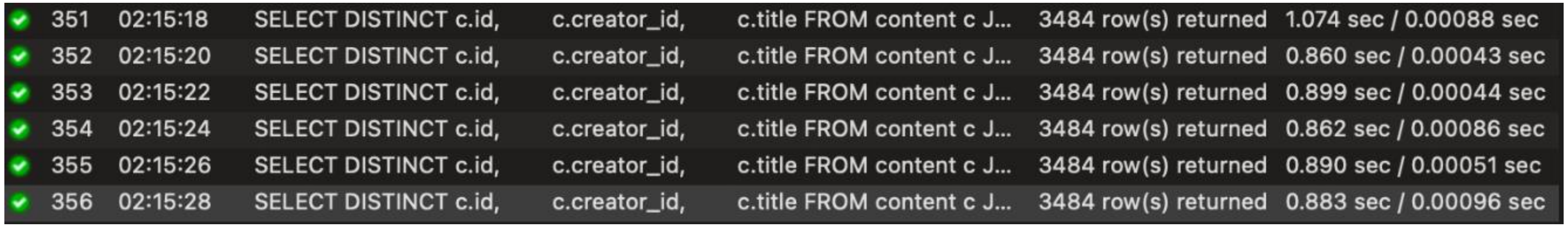


table	partitions	type	possible_keys	key	key_len	ref	rows
cr	NULL	index	PRIMARY,idx_creator_channel_name	idx_creator_channel_name	1022	NULL	9882
c	NULL	ref	creator_id,idx_content_title	creator_id	8	fulltext_test.cr.id	9
cp	NULL	ref	content_id	content_id	8	fulltext_test.c.id	4
p	NULL	eq_ref	PRIMARY	PRIMARY	8	fulltext_test.cp.place_id	1

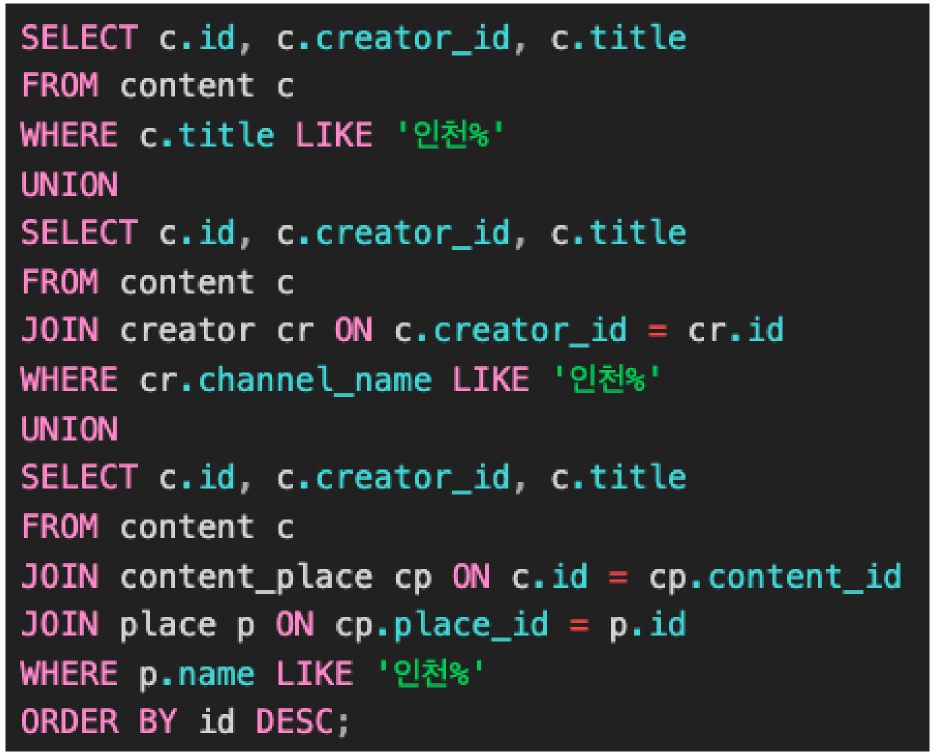


```
351  02:15:18  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  1.074 sec / 0.00088 sec
352  02:15:20  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.860 sec / 0.00043 sec
353  02:15:22  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.899 sec / 0.00044 sec
354  02:15:24  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.862 sec / 0.00086 sec
355  02:15:26  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.890 sec / 0.00051 sec
356  02:15:28  SELECT DISTINCT c.id,  c.creator_id,  c.title FROM content c J...  3484 row(s) returned  0.883 sec / 0.00096 sec
```

평균 0.878sec

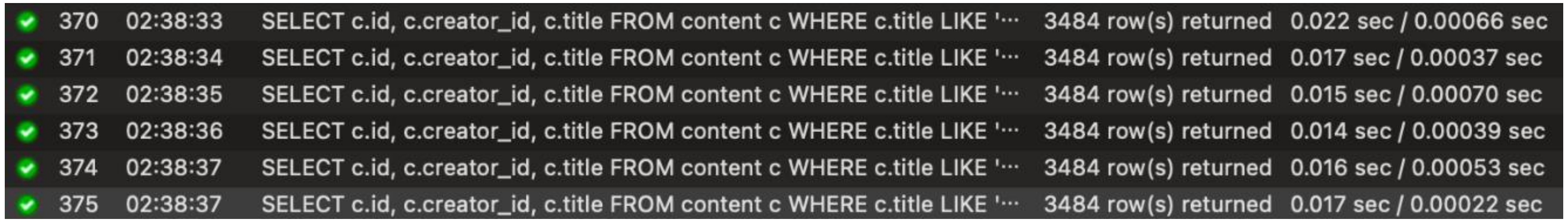
인덱스 추가 - 실제 쿼리 개선 실험

LIKE 인천%은 많이 빨라지려나?



```
SELECT c.id, c.creator_id, c.title
FROM content c
WHERE c.title LIKE '인천%'
UNION
SELECT c.id, c.creator_id, c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
WHERE cr.channel_name LIKE '인천%'
UNION
SELECT c.id, c.creator_id, c.title
FROM content c
JOIN content_place cp ON c.id = cp.content_id
JOIN place p ON cp.place_id = p.id
WHERE p.name LIKE '인천%'
ORDER BY id DESC;
```

WHERE OR 대신 union 활용
=> 각 컬럼 인덱스 활용할 수 있음



```
✓ 370  02:38:33  SELECT c.id, c.creator_id, c.title FROM content c WHERE c.title LIKE '…  3484 row(s) returned  0.022 sec / 0.00066 sec
✓ 371  02:38:34  SELECT c.id, c.creator_id, c.title FROM content c WHERE c.title LIKE '…  3484 row(s) returned  0.017 sec / 0.00037 sec
✓ 372  02:38:35  SELECT c.id, c.creator_id, c.title FROM content c WHERE c.title LIKE '…  3484 row(s) returned  0.015 sec / 0.00070 sec
✓ 373  02:38:36  SELECT c.id, c.creator_id, c.title FROM content c WHERE c.title LIKE '…  3484 row(s) returned  0.014 sec / 0.00039 sec
✓ 374  02:38:37  SELECT c.id, c.creator_id, c.title FROM content c WHERE c.title LIKE '…  3484 row(s) returned  0.016 sec / 0.00053 sec
✓ 375  02:38:37  SELECT c.id, c.creator_id, c.title FROM content c WHERE c.title LIKE '…  3484 row(s) returned  0.017 sec / 0.00022 sec
```

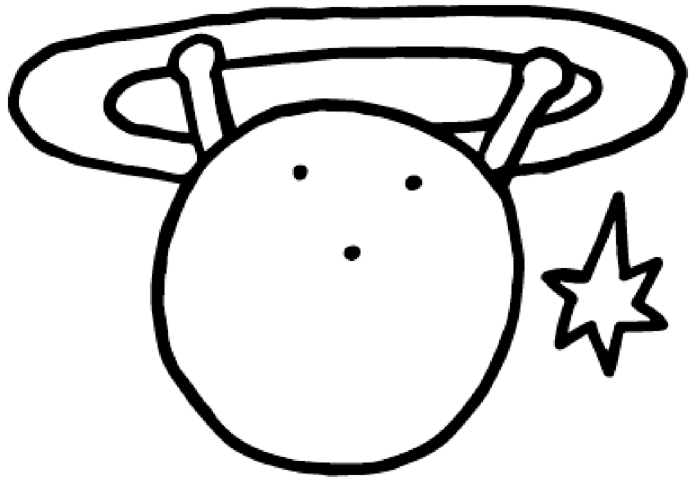
평균 0.0158sec

인덱스 추가 - 실제 쿼리 개선 실험 결과

index가 존재해도, JOIN, WHERE OR ... 다양한 조건을 쓰다보면 인덱스를 잘 활용
하지 못할 수 있다

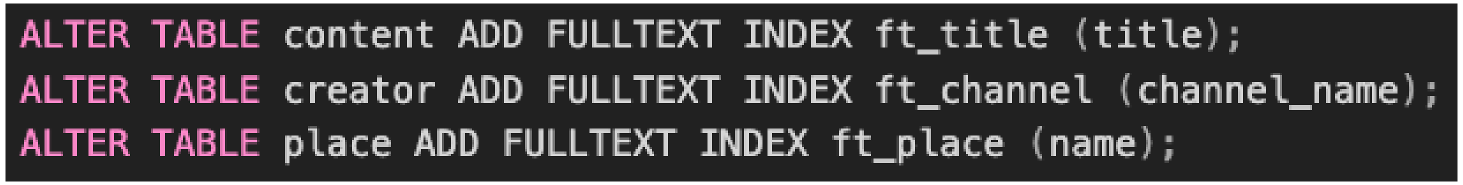
join을 이용하고, where or 조건이 복잡해질 때는 union을 고려해보자

세 번째 실험

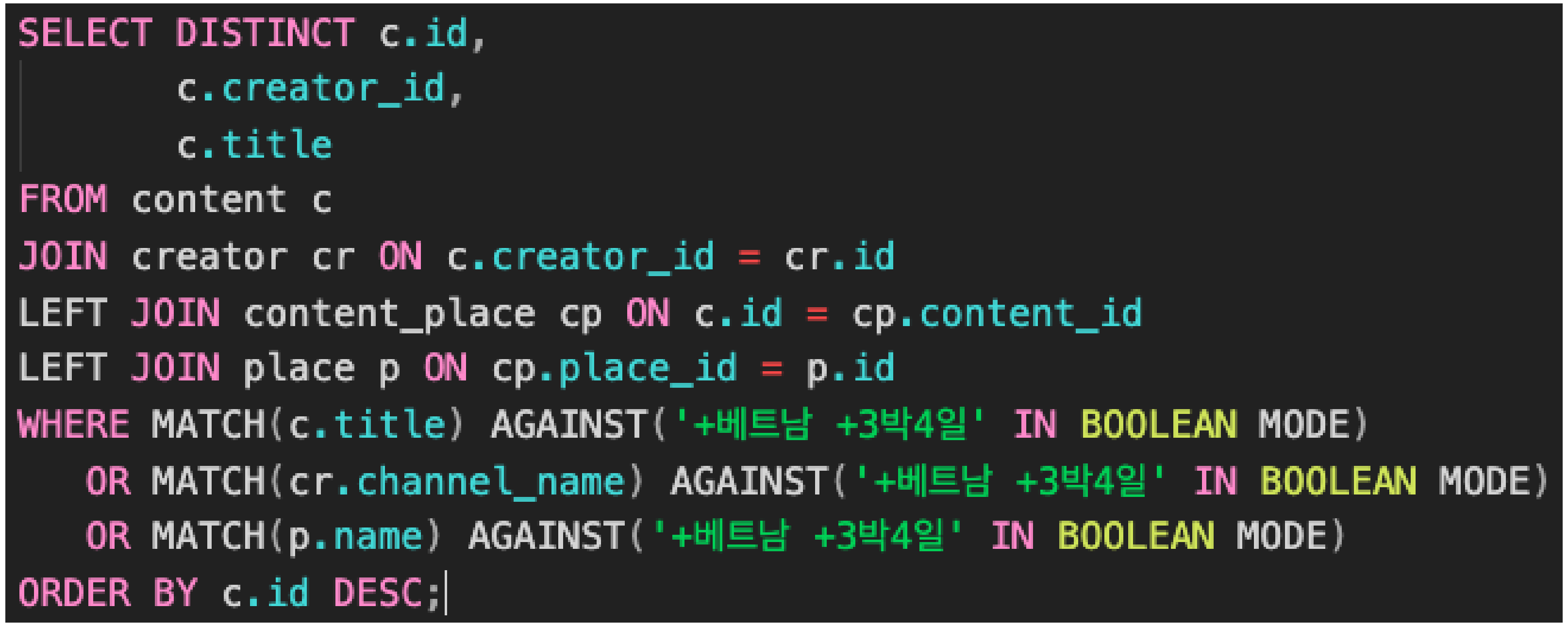


fulltext index 추가 - 다중 키워드 검색 실험

fulltext index를 활용하면 여러 키워드 기반 탐색이 가능할까?



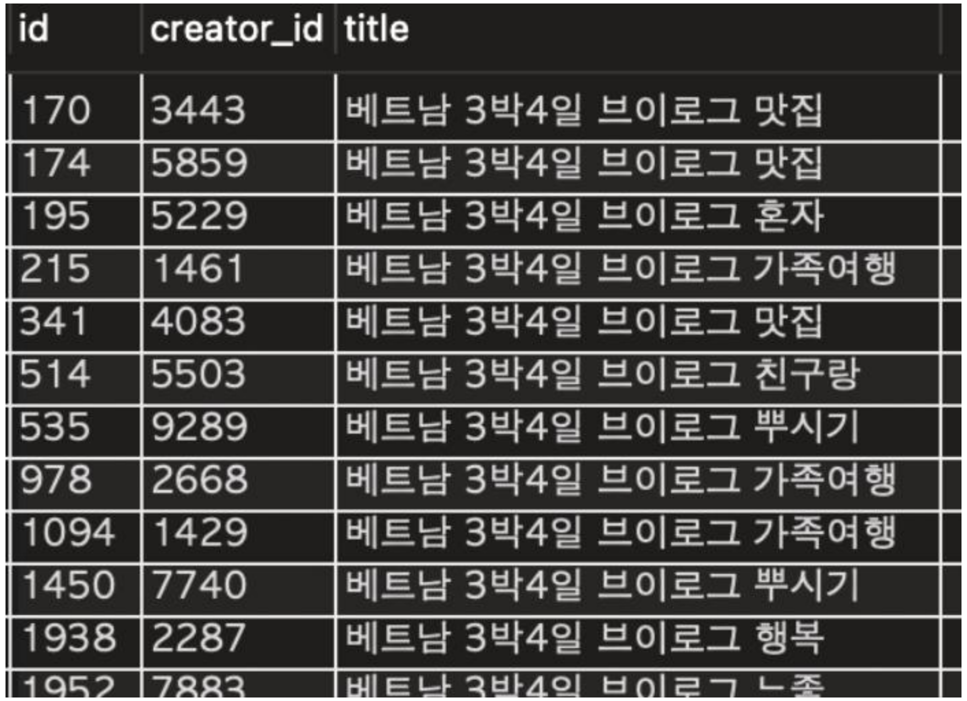
```sql
ALTER TABLE content ADD FULLTEXT INDEX ft_title (title);
ALTER TABLE creator ADD FULLTEXT INDEX ft_channel (channel_name);
ALTER TABLE place ADD FULLTEXT INDEX ft_place (name);
```



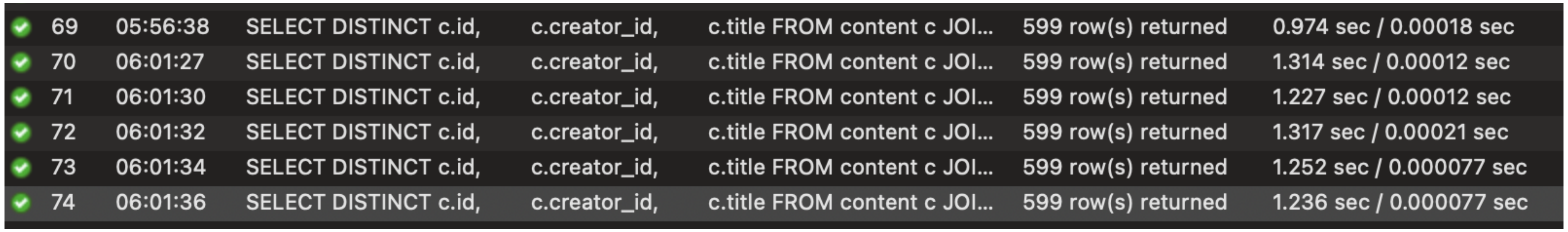
```sql
SELECT DISTINCT c.id,
       c.creator_id,
       c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
LEFT JOIN content_place cp ON c.id = cp.content_id
LEFT JOIN place p ON cp.place_id = p.id
WHERE MATCH(c.title) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)
   OR MATCH(cr.channel_name) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)
   OR MATCH(p.name) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)
ORDER BY c.id DESC;
```

fulltext index 추가 다중 키워드 검색 실험

fulltext index를 활용하면 여러 키워드 기반 탐색이 가능할까?



id	creator_id	title
170	3443	베트남 3박4일 브이로그 맛집
174	5859	베트남 3박4일 브이로그 맛집
195	5229	베트남 3박4일 브이로그 혼자
215	1461	베트남 3박4일 브이로그 가족여행
341	4083	베트남 3박4일 브이로그 맛집
514	5503	베트남 3박4일 브이로그 친구랑
535	9289	베트남 3박4일 브이로그 뿌시기
978	2668	베트남 3박4일 브이로그 가족여행
1094	1429	베트남 3박4일 브이로그 가족여행
1450	7740	베트남 3박4일 브이로그 뿌시기
1938	2287	베트남 3박4일 브이로그 행복
1952	7883	베트남 3박4일 브이로그 느좋

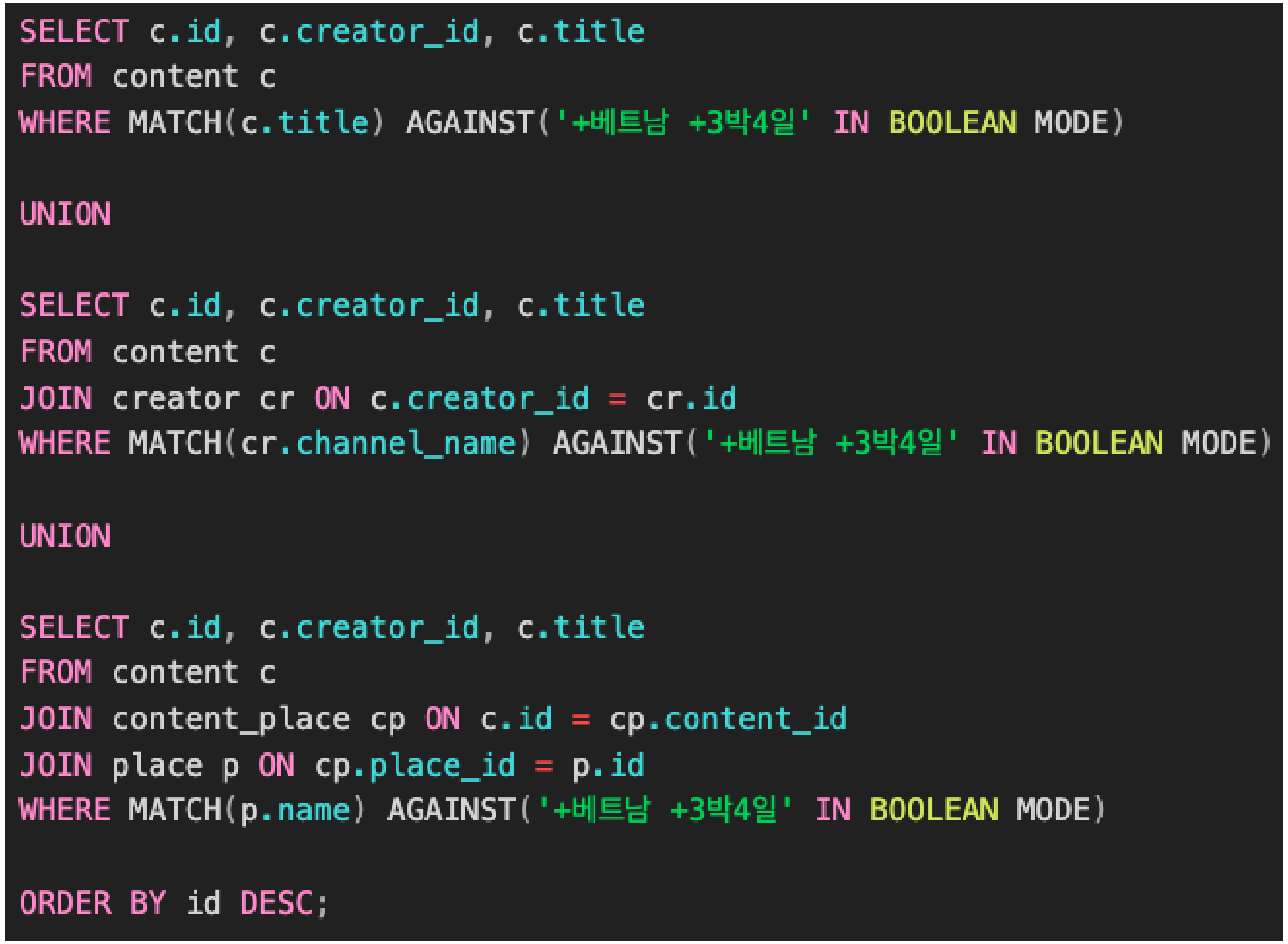


```
69   05:56:38   SELECT DISTINCT c.id,   c.creator_id,   c.title FROM content c JOI...   599 row(s) returned   0.974 sec / 0.00018 sec
70   06:01:27   SELECT DISTINCT c.id,   c.creator_id,   c.title FROM content c JOI...   599 row(s) returned   1.314 sec / 0.00012 sec
71   06:01:30   SELECT DISTINCT c.id,   c.creator_id,   c.title FROM content c JOI...   599 row(s) returned   1.227 sec / 0.00012 sec
72   06:01:32   SELECT DISTINCT c.id,   c.creator_id,   c.title FROM content c JOI...   599 row(s) returned   1.317 sec / 0.00021 sec
73   06:01:34   SELECT DISTINCT c.id,   c.creator_id,   c.title FROM content c JOI...   599 row(s) returned   1.252 sec / 0.000077 sec
74   06:01:36   SELECT DISTINCT c.id,   c.creator_id,   c.title FROM content c JOI...   599 row(s) returned   1.236 sec / 0.000077 sec
```

평균 1.269sec

fulltext index 추가 - 다중 키워드 검색 실험

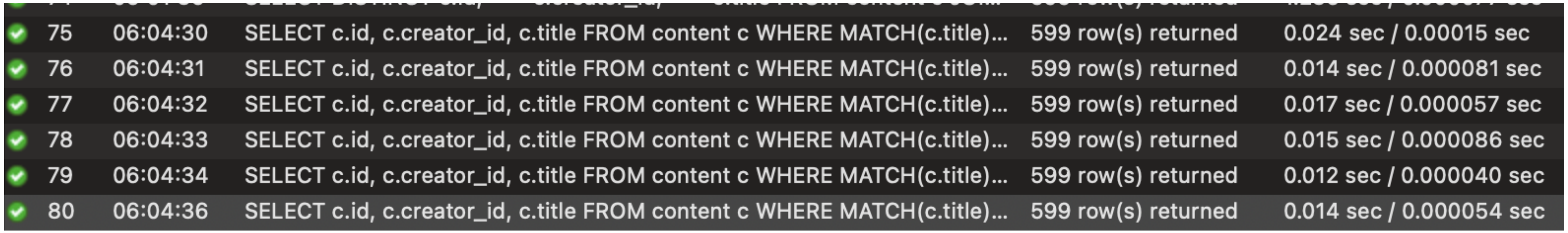
이것도 union으로 바꾸면 빨라지겠지?



```
SELECT c.id, c.creator_id, c.title
FROM content c
WHERE MATCH(c.title) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

UNION

SELECT c.id, c.creator_id, c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
WHERE MATCH(cr.channel_name) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

UNION

SELECT c.id, c.creator_id, c.title
FROM content c
JOIN content_place cp ON c.id = cp.content_id
JOIN place p ON cp.place_id = p.id
WHERE MATCH(p.name) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

ORDER BY id DESC;
```

fulltext index 추가 - 다중 키워드 검색 실험

이것도 union으로 바꾸면 빨라지겠지?

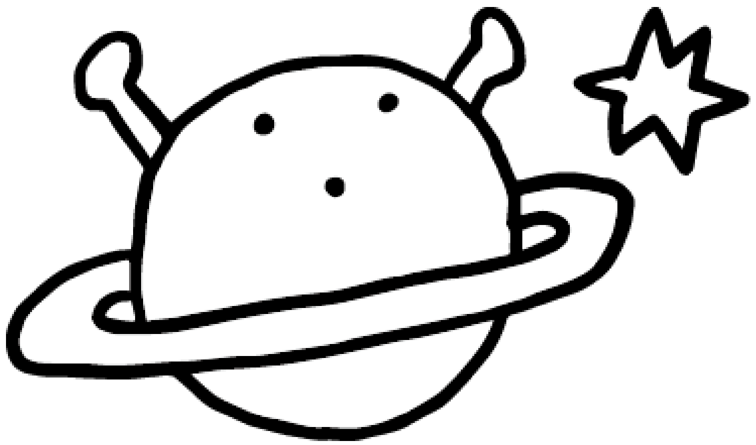


```
75   06:04:30   SELECT c.id, c.creator_id, c.title FROM content c WHERE MATCH(c.title)...   599 row(s) returned   0.024 sec / 0.00015 sec
76   06:04:31   SELECT c.id, c.creator_id, c.title FROM content c WHERE MATCH(c.title)...   599 row(s) returned   0.014 sec / 0.000081 sec
77   06:04:32   SELECT c.id, c.creator_id, c.title FROM content c WHERE MATCH(c.title)...   599 row(s) returned   0.017 sec / 0.000057 sec
78   06:04:33   SELECT c.id, c.creator_id, c.title FROM content c WHERE MATCH(c.title)...   599 row(s) returned   0.015 sec / 0.000086 sec
79   06:04:34   SELECT c.id, c.creator_id, c.title FROM content c WHERE MATCH(c.title)...   599 row(s) returned   0.012 sec / 0.000040 sec
80   06:04:36   SELECT c.id, c.creator_id, c.title FROM content c WHERE MATCH(c.title)...   599 row(s) returned   0.014 sec / 0.000054 sec
```

평균 0.072sec

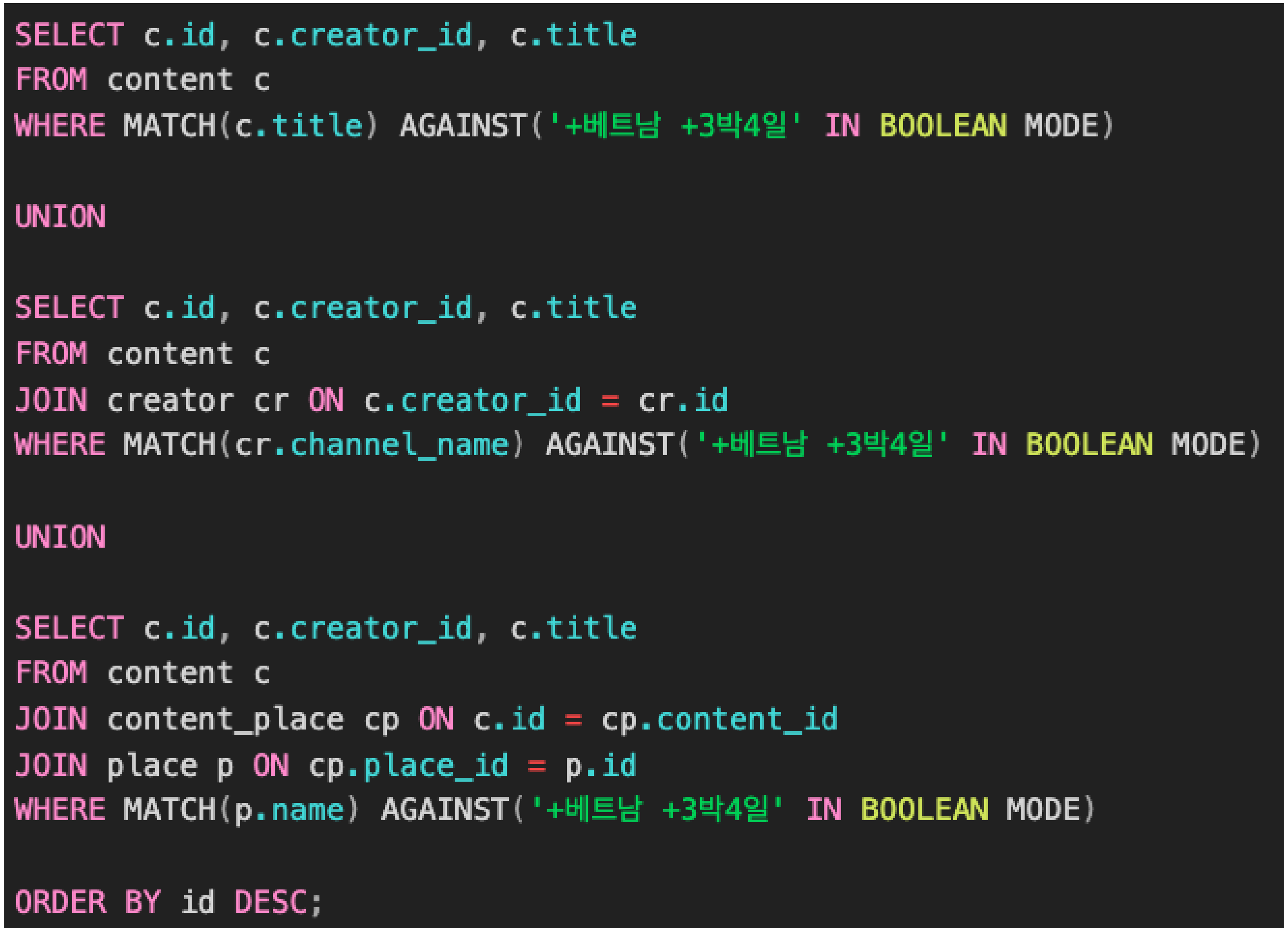
fulltext index 추가 - 다중 키워드 검색 실험 결과

MySQL fulltext index 기능을 사용하면 여러 키워드를 기반으로 검색할 수 있다는 것을 확인



결론

최종 개선 쿼리



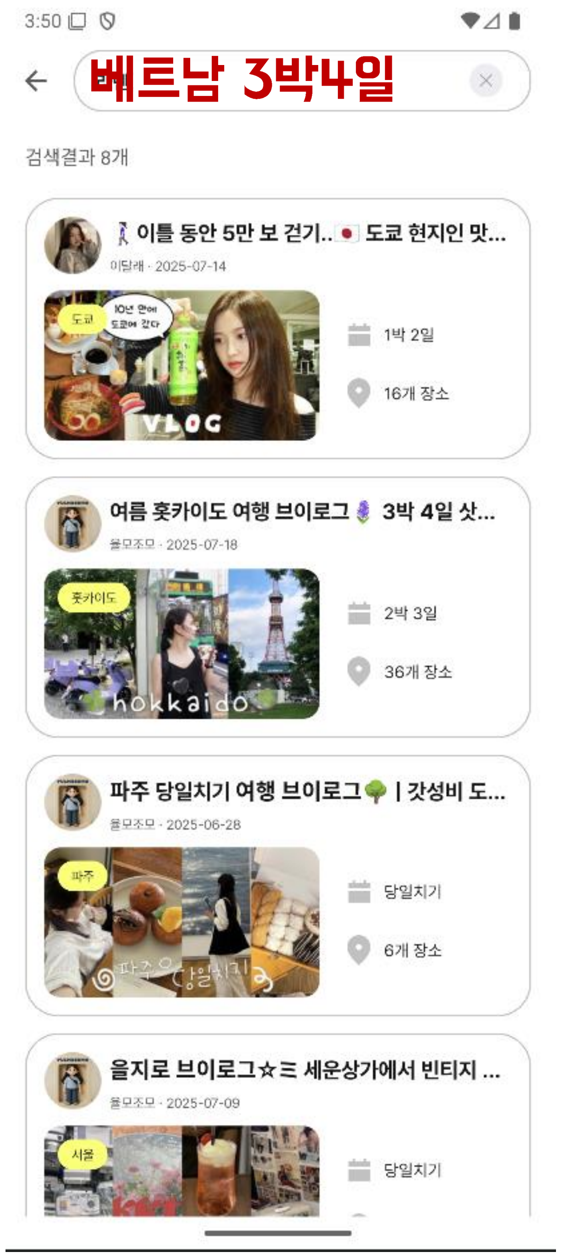
```
SELECT c.id, c.creator_id, c.title
FROM content c
WHERE MATCH(c.title) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

UNION

SELECT c.id, c.creator_id, c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
WHERE MATCH(cr.channel_name) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

UNION

SELECT c.id, c.creator_id, c.title
FROM content c
JOIN content_place cp ON c.id = cp.content_id
JOIN place p ON cp.place_id = p.id
WHERE MATCH(p.name) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

ORDER BY id DESC;
```

검색 관련 컬럼에 fulltext index 추가

기존 쿼리 : 평균 0.918sec
개선 쿼리 : 평균 0.072sec

-> 12.75배 빨라짐

개선된 검색 기능



3:50

베트남 3박4일

검색결과 8개

🚶 이틀 동안 5만 보 걷기.. 🇯🇵 도쿄 현지인 맛...
이달래 · 2025-07-14
1박 2일
16개 장소

여름 홋카이도 여행 브이로그 🪻 3박 4일 삿...
몰모조모 · 2025-07-18
2박 3일
36개 장소

파주 당일치기 여행 브이로그🌳 | 갓성비 도...
몰모조모 · 2025-06-28
당일치기
6개 장소

을지로 브이로그☆ミ 세운상가에서 빈티지 ...
몰모조모 · 2025-07-09
당일치기

다중 키워드 검색 가능

데이터 3만개, 10만개, 50만개가 존재하는 테이블 기준
검색에 평균 0.072sec 소요

추가로 고려해야 할 부분

- 검색 전용 컬럼 만들어 fulltext index 적용
- fulltext index 단점에 대해 알아보기 (ex. 디스크 용량 차지, 쓰기 성능 저하)
- 실제 운영 환경에서의 속도 확인해보기
- MySQL 쿼리짜는법, DB 열심히 공부하기

Q&A