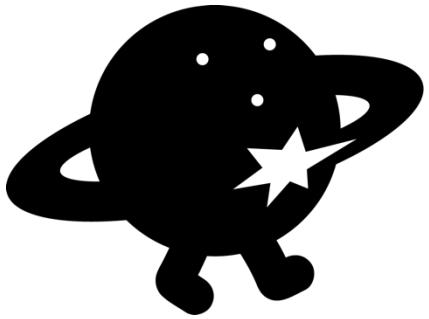
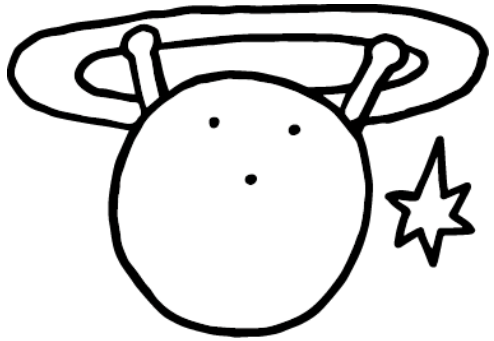
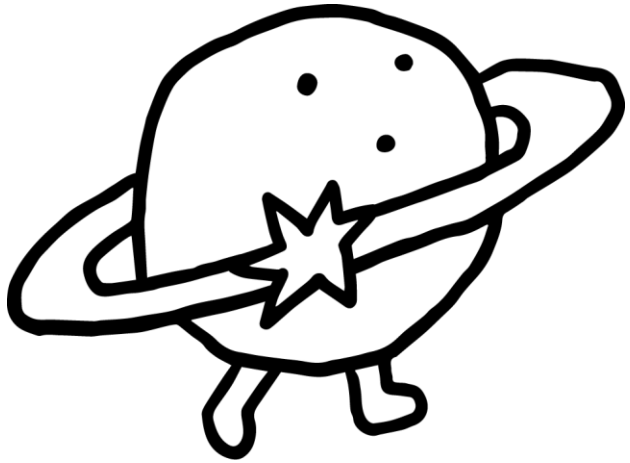


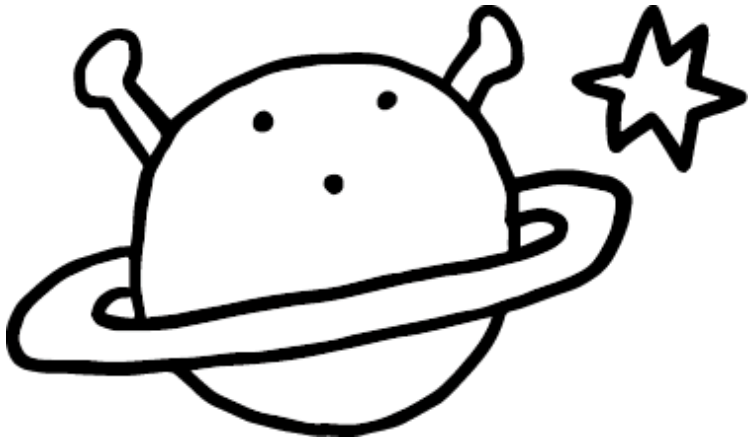


검색 기능 개선 실험하기 2

(innodb 버퍼풀 cache-hit rate 늘리기)

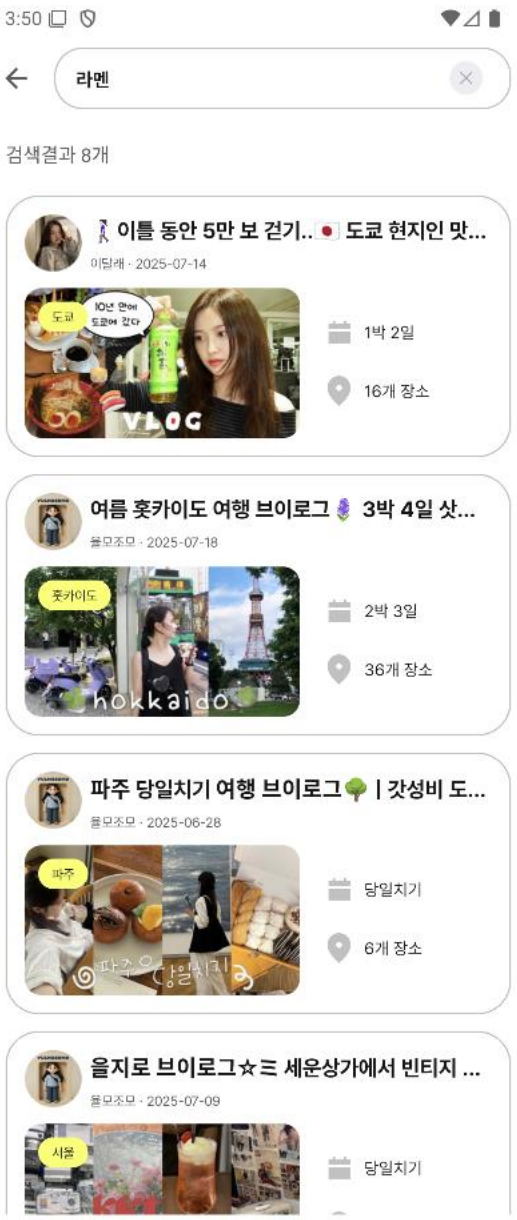


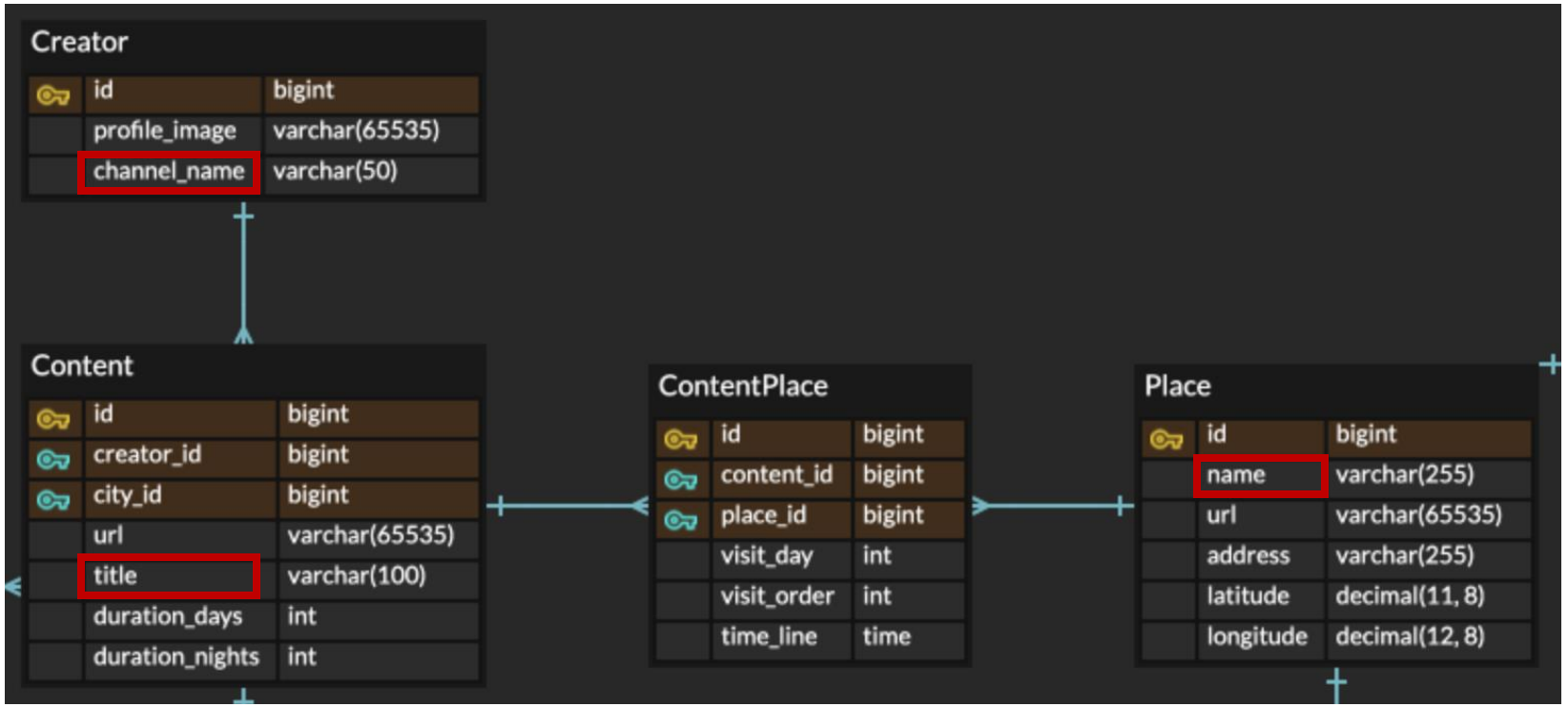
BE 7기 메이



기존 검색 기능

기존 검색 기능





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

기존 검색 쿼리

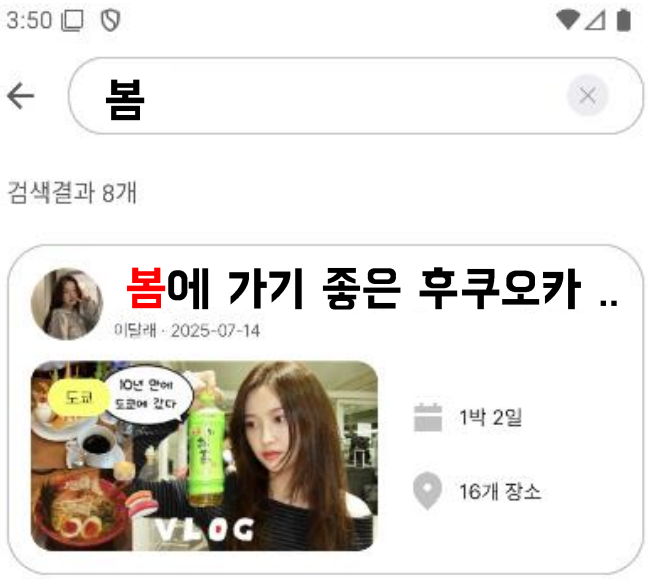
```sql
SELECT c.id, c.creator_id, c.title
FROM content c
WHERE MATCH(c.title) AGAINST('+베트남 +3박4일' IN BOOLEAN MODE)

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;
```

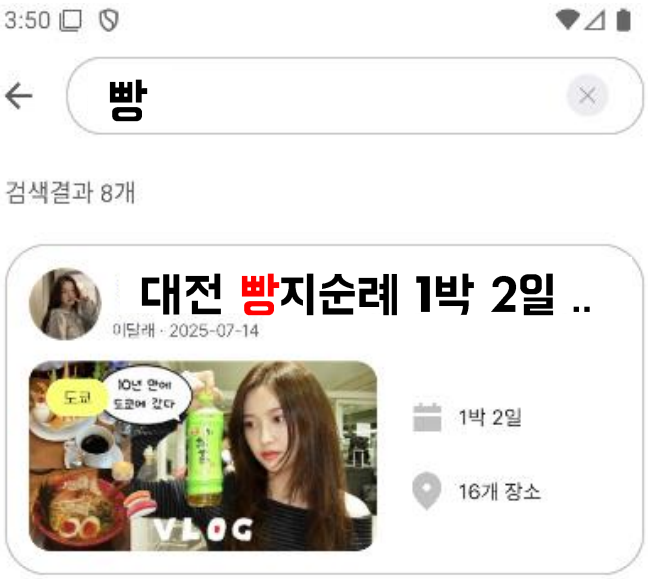
Variable_name	Value
ngram_token_size	1

검색 쿼리

Fulltext index(n-gram parser)

검색 정책

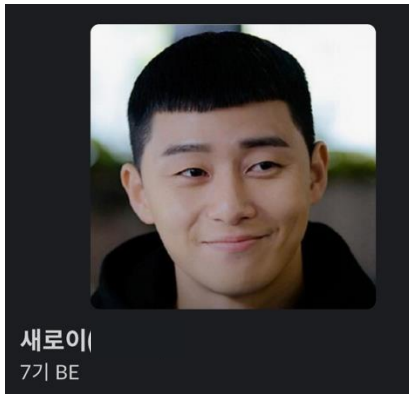




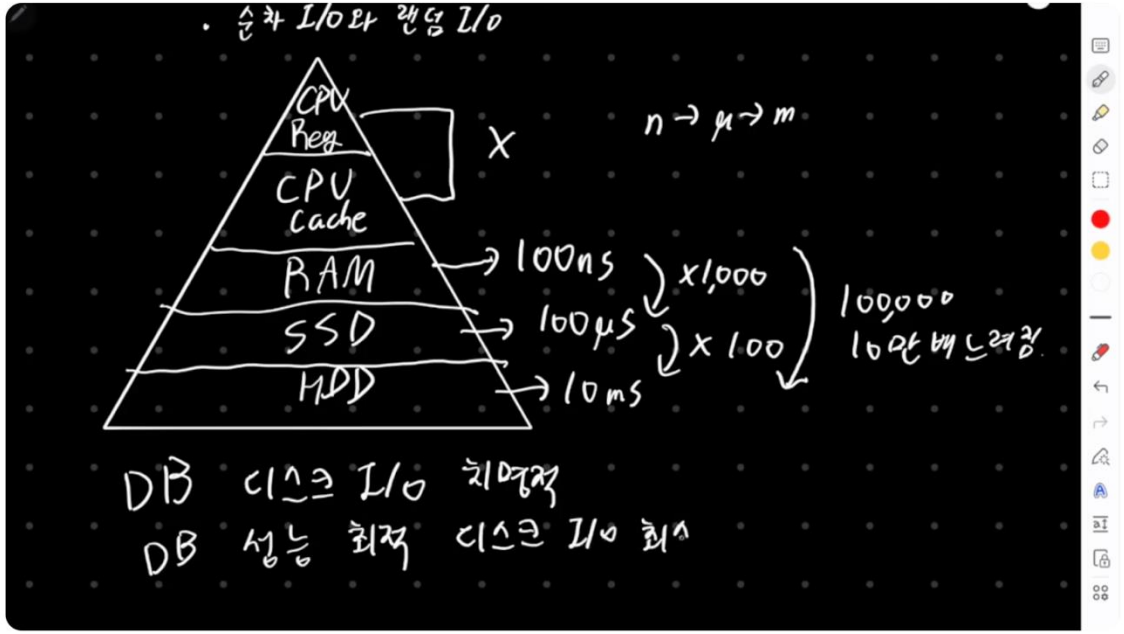
여행 도메인이기 때문에, 1글자 검색이 가능하도록 정책 설정

성능 개선에 대한 고민 배경

검색 기능에 대한 고민 배경



오 ngram size 1?? 버퍼풀 사이즈 고려해서 fulltext index 크기 확인해보삼

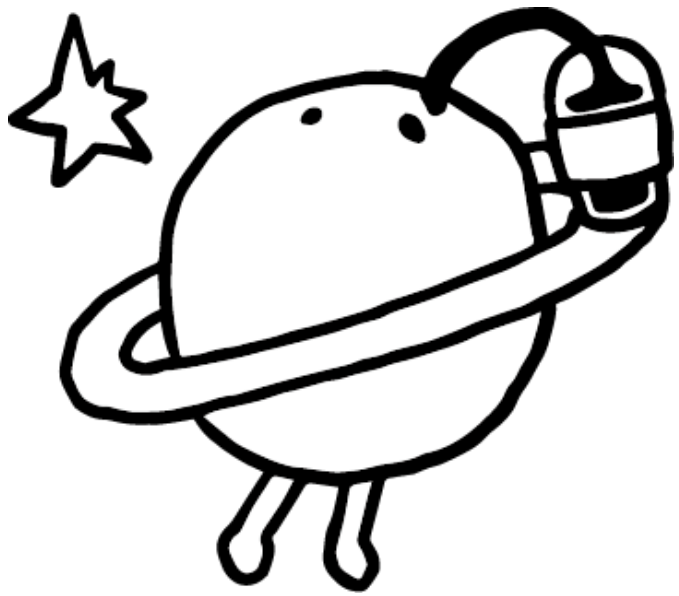


5차시 - I/O와 인덱스 1

모코
구독자 12명

2 공유 오프라인 저장 클립

배경 지식 설명

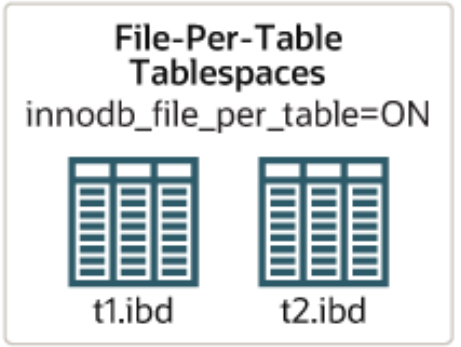


디스크 io

Disk

10,000,000 ns

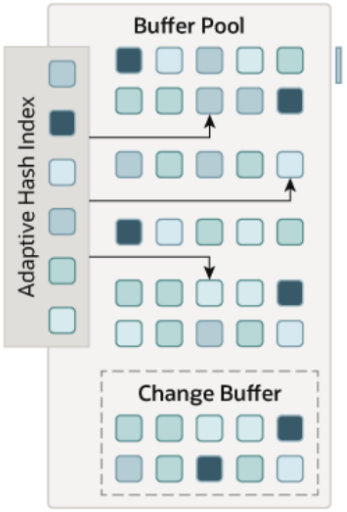
cpu



Memory

10ns

cpu



속도 최대 약 100,000배 차이

MySQL 메모리 구조 - 버퍼풀

MySQL innodb -> 디스크에서 가져온 데이터를 버퍼풀에 저장

17.5.1 Buffer Pool

The buffer pool is an area in main memory where `InnoDB` caches table and index data as it is accessed. The buffer pool permits frequently used data to be accessed directly from memory, which speeds up processing. On dedicated servers, up to 80% of physical memory is often assigned to the buffer pool.

테이블, 인덱스를 캐싱

물리메모리의 80%까지 버퍼풀에 할당 가능

For efficiency of high-volume read operations, the buffer pool is divided into pages that can potentially hold multiple rows. For efficiency of cache management, the buffer pool is implemented as a linked list of pages; data that is rarely used is aged out of the cache using a variation of the least recently used (LRU) algorithm.

페이지 단위 동작

MySQL 메모리 구조 - 버퍼풀

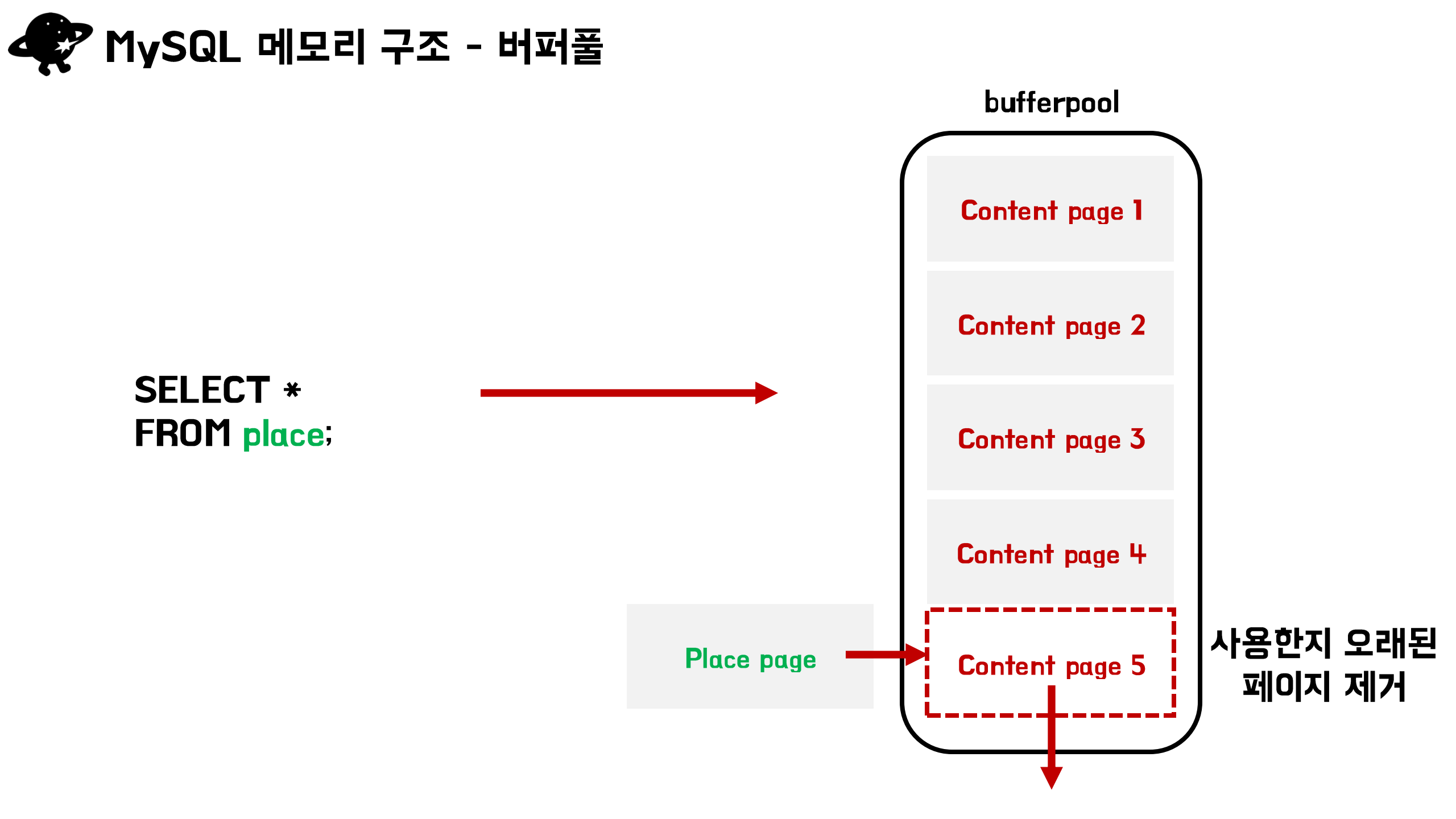
```
# The buffer pool has a default size of 128MB (134217728 bytes)

mysql> SELECT @@innodb_buffer_pool_size;
+-------------------------------+
| @@innodb_buffer_pool_size |
+-------------------------------+
|                 134217728 |
+-------------------------------+
```

기본 버퍼풀 사이즈 128MB

MySQL 메모리 구조 - 버퍼풀

```
SELECT *
FROM content;
```

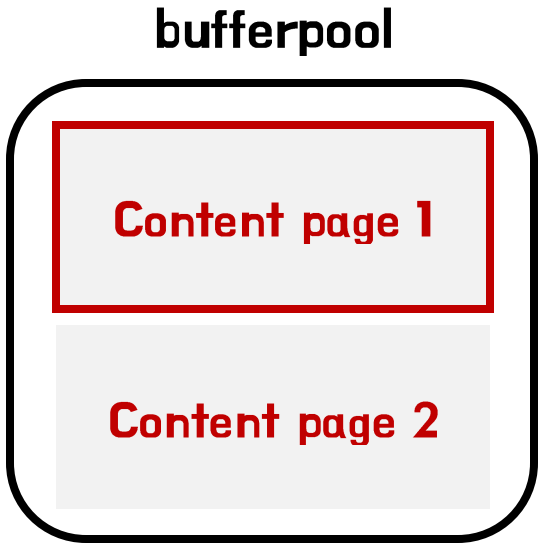
MySQL 메모리 구조 - 버퍼풀



MySQL 메모리 구조 – cache hit / cache miss

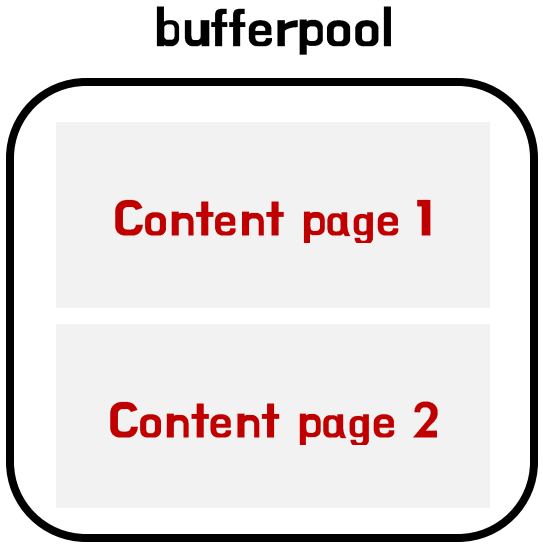
```
SELECT *
FROM content
WHERE id = 1;
```

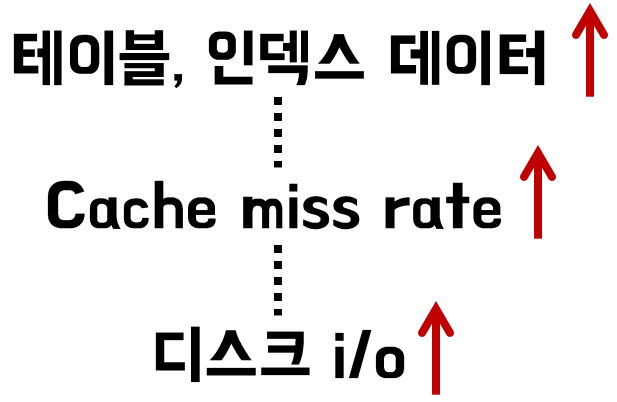
Cache Hit



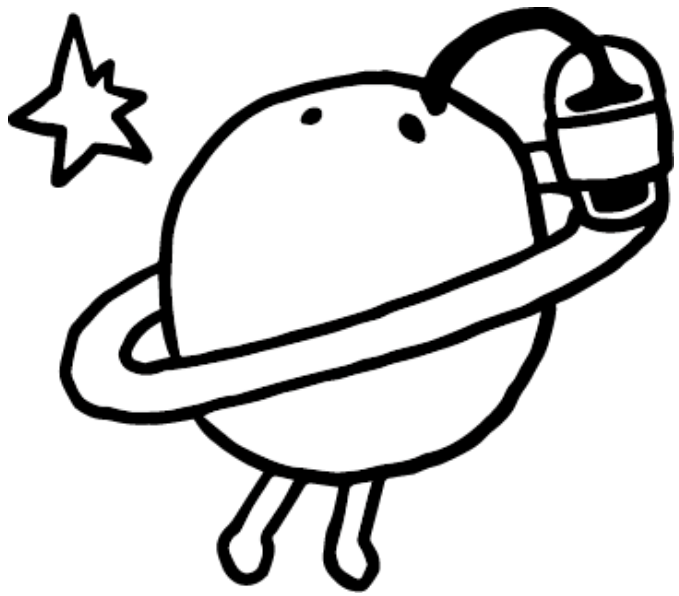

```
SELECT *
FROM creator;
```

Cache miss





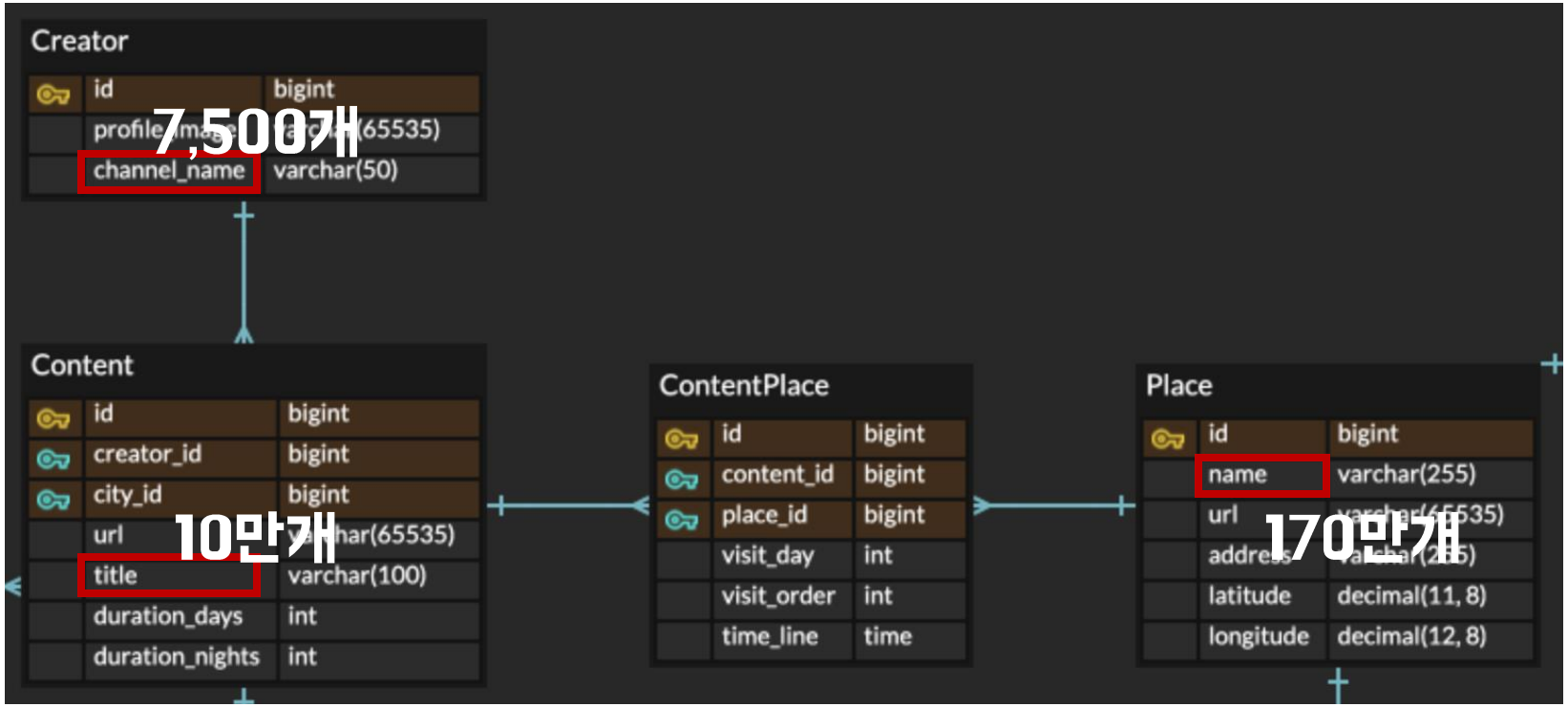
검색 기능에 사용하는 Index 크기 확인



테스트 환경

t4g.micro(1GiB)

운영 환경과 같은 스펙



- 컨텐츠(content) 테이블의 제목(title) 컬럼 -> 20글자
- 크리에이터(creator) 테이블의 채널명(channel_name) 컬럼 -> 7글자
- 장소(place) 테이블의 이름(name) 컬럼 -> 8글자

검색 쿼리에 사용하는 인덱스

```sql
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 크기 확인

FTS 관련 데이터 크기 확인

```sql
SELECT
    SUM(T.actual_mb) AS total_fts_size_mb
FROM
    (
    SELECT
        NAME,
        SPACE,
        ROUND(ALLOCATED_SIZE / 1024 / 1024, 2) as actual_mb
    FROM INFORMATION_SCHEMA.INNODB_TABLESPACES
    WHERE NAME LIKE '%turip_bufferpool/fts%'
    ) AS T;
```

total_fts_size_mb
42.96

N-gram size: 2

total_fts_size_mb
54.99

N-gram size: 1

Clustered Index 크기 확인

	TABLE_NAME	clustered_index_...	secondary_indexes_...
	place	196.75	35.59
	content	24.55	2.52
	creator	0.05	0.02

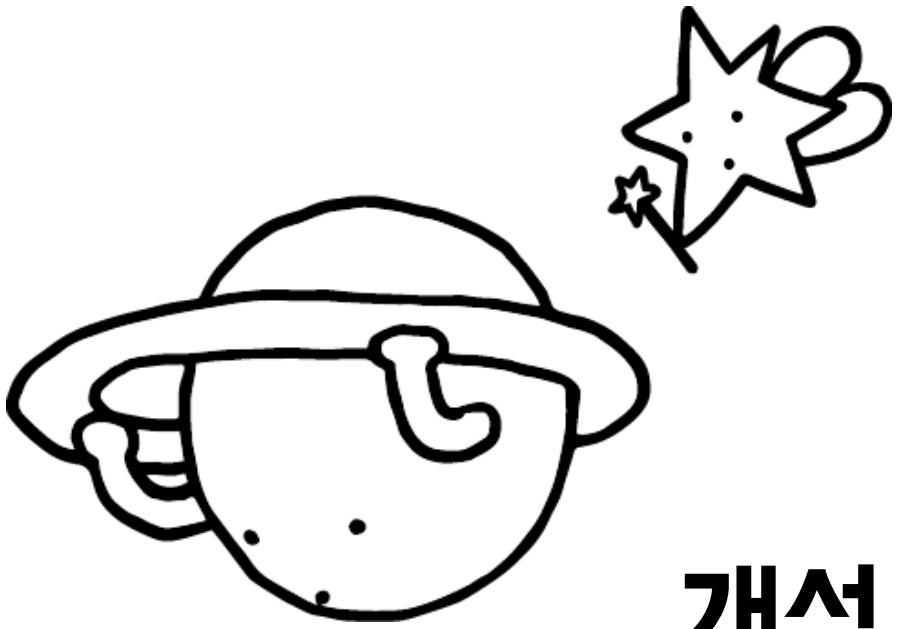
Place의 클러스터드 인덱스만 200MB

검색 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;
```

TABLE_NAME	INDEX_NAME	page_count
`turip_bufferpool`.`place`	PRIMARY	3678
`turip_bufferpool`.`place`	FTS_DOC_ID_INDEX	1556

place 데이터는 필요 없는데 버퍼풀에 올라감



개선 방향:
어떻게 cache hit rate를 올릴 수 있을까?

검색 전용 테이블 관리

```
-- 통합 검색 테이블
CREATE TABLE content_search (
    content_id BIGINT PRIMARY KEY,
    search_text TEXT,  -- title + place + creator 통합
    FULLTEXT INDEX ft_search (search_text) WITH PARSER ngram
);

-- 데이터 예시
content_id | search_text
-----------|-----------------------------------------
1          | "부산 해운대 맛집 존맛탱 빵집 하루"
2          | "서울 강남 카페 투어 커피공장 제리"
```

```
SELECT c.id, c.creator_id, c.title
FROM content c
JOIN content_search cs ON c.id = cs.content_id
WHERE MATCH(cs.search_text) AGAINST('+연산동' IN BOOLEAN MODE) ORDER BY c.id DESC;
```

-> place, creator 클러스터드 인덱스가 버퍼풀에 올라오지 않을 것으로 기대

검색 전용 테이블 관리

content.title: 수원 행궁동 브이로그

creator.channel_name: 튜립

place.name: 아웃백 스테이크 / 로우파이브 / 인생네컷 / 골디스 행궁점 / 노팅힐 베이커리

검색에 사용하는 문구를 한 컬럼으로 관리

content_search.search_text: 수원 행궁동 브이로그 튜립 아웃백 스테이크 로우파이브 ...

검색 전용 테이블 관리 - 인덱스 크기 확인

TABLE_NAME	clustered_index_...	secondary_indexes_...	TABLE_ROWS
content_search	17.55	2.52	98940

검색 테이블의 클러스터드, 세컨더리 인덱스 크기

total_fts_size_mb
21.78

검색 컬럼에 대한 fulltext index 크기

검색 전용 테이블 관리 – 버퍼풀 비교

기존 쿼리

TABLE_NAME	INDEX_NAME	page_count
`turip_bufferpool`.`content_place`	PRIMARY	3144
`turip_bufferpool`.`place`	PRIMARY	1889
`turip_bufferpool`.`content_place`	idx_place_id	515
`turip_bufferpool`.`content`	PRIMARY	201
`turip_bufferpool`.`place`	FTS_DOC_ID_INDEX	416

Innodb_buffer_pool_pages_data	7164
Innodb_buffer_pool_bytes_data	117374976
Innodb_buffer_pool_pages_dirty	0
Innodb_buffer_pool_bytes_dirty	0
Innodb_buffer_pool_pages_flushed	299
Innodb_buffer_pool_pages_free	1023

개선 쿼리

TABLE_NAME	INDEX_NAME	page_count
`turip_bufferpool`.`content_search`	FTS_DOC_ID_INDEX	133
`turip_bufferpool`.`content_search`	PRIMARY	411
`turip_bufferpool`.`content`	PRIMARY	374

Innodb_buffer_pool_pages_data	4373
Innodb_buffer_pool_bytes_data	71647232
Innodb_buffer_pool_pages_dirty	0
Innodb_buffer_pool_bytes_dirty	0
Innodb_buffer_pool_pages_flushed	394
Innodb_buffer_pool_pages_free	3813

같은 검색어에 대한 쿼리인데,
버퍼풀에 올라오는 페이지 차이가 약 43MB(2791 페이지)

검색 전용 테이블 관리 – 쿼리 속도 비교

기존 쿼리

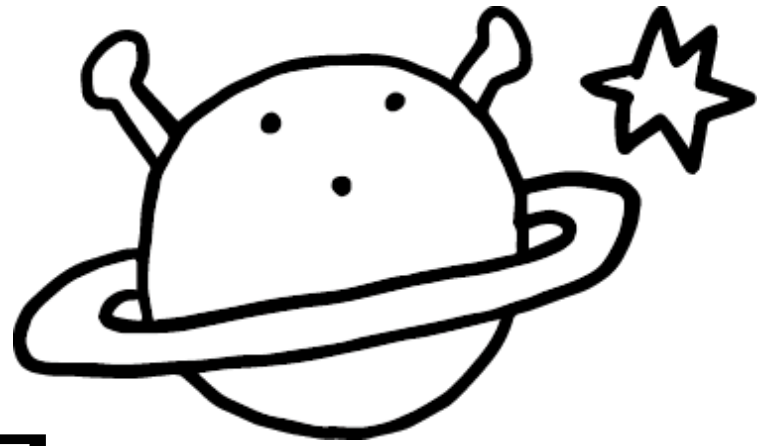
```sql
SELECT c.id, c.creator_id, c.title
FROM content c
WHERE MATCH(c.title) AGAINST('+부산 +여행' IN BOOLEAN MODE)
UNION
SELECT c.id, c.creator_id, c.title
FROM content c
JOIN creator cr ON c.creator_id = cr.id
WHERE MATCH(cr.channel_name) AGAINST('+부산 +여행' IN BOOLEAN MODE)
UNION
SELECT c.id, c.creator_id, 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('+부산 +여행' IN BOOLEAN MODE)
ORDER BY id DESC;
```

warm_avg_ms
1133.981

개선 쿼리

```sql
SELECT c.id, c.creator_id, c.title
FROM content c
JOIN content_search cs ON c.id = cs.content_id
WHERE MATCH(cs.search_text) AGAINST('+부산 +여행' IN BOOLEAN MODE)
ORDER BY c.id DESC;
```

warm_avg_ms
929.913



결론

발표 내용 정리

검색에 fulltext index 이용, n-gram size 1 설정

↓

버퍼풀 용량 차지, 디스크 i/o 증가 문제 인식

↓

검색 쿼리에서 사용하는 인덱스 용량 확인

↓

불필요하게 버퍼풀에 올라오는 클러스터드 인덱스 확인

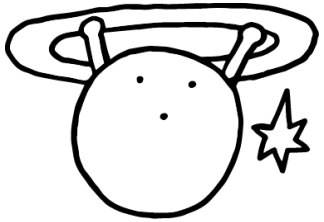
↓

이를 개선하기 위해 검색 전용 테이블 사용

↓

불필요한 버퍼풀 차지 방지, cache hit rate 개선 기대

검색 전용 테이블 관리 - 장단점



장점

- 버퍼풀 사용량 ↓, cache hit rate ↑
- 쿼리 단순화, 속도 개선
- 다중 컬럼 검색 지원

  콘텐츠 제목에 “도쿄”, 장소 이름중에 “라멘” 존재하면
  “도쿄 라멘” 검색 가능

단점

- 검색 테이블 관리 비용
- 검색 정책에 의존하는 테이블
- content, creator, place 데이터에 의존

Q&A