Rhymix 게시판 인기글/베스트글 기능 저부하 개발 방향

모듈 설계, 캐시 전략, 랭킹 테이블 구조, 갱신 방식 요약

핵심 결론: 게시판을 볼 때마다 계산하지 말고, 미리 계산된 인기글 목록을 읽기만 하는 구조가 가장 부하가 적습니다.

1. 결론 구조

가장 부하가 적은 구조는 “모듈 + 랭킹 캐시 테이블 + 짧은 TTL 캐시 + 이벤트/배치 갱신” 조합입니다.

디시/더쿠류 UI 처럼 하단에 “인기글”, “HOT 게시물”, “베스트글” 버튼이 있어도, 그 버튼이나 영역이 매번 documents 전체를 뒤져서 정렬하면 안 됩니다. 화면에서는 미리 만들어진 인기글 목록을 읽기만 해야 합니다.

Rhymix 의 documents 테이블에는 readed_count, voted_count, comment_count, regdate, list_order 같은 인기글 산정에 쓸 수 있는 컬럼이 있습니다. 그러나 “특정 게시판 + 최근 기간 + 추천/댓글/조회수 가중치 + 정렬”처럼 복합 조건이 들어가면, 코어 documents 테이블을 매번 직접 정렬하는 방식은 커뮤니티 규모가 커질수록 비효율적입니다.

2. 실시간 조회 쿼리는 피해야 함

피해야 할 방식은 이런 형태입니다.

```
SELECT *
FROM rx_documents
WHERE module_srl = ?
  AND regdate >= ?
  AND status = 'PUBLIC'
ORDER BY voted_count DESC, comment_count DESC, readed_count DESC
LIMIT 20;
```

이 쿼리를 게시판 목록을 볼 때마다 실행하면 방문자가 많을수록 DB 가 계속 정렬 작업을 하게 됩니다. 특히 전체 게시판 인기글, 최근 24 시간 인기글, 주간 베스트처럼 범위가 넓어질수록 부하가 커집니다.

대신 화면에서는 작은 랭킹 테이블에서 이미 정해진 순서만 읽는 구조가 좋습니다.

```
SELECT document_srl
FROM rx_popular_rank
WHERE scope = 'board'
  AND module_srl = ?
  AND period = 'today'
ORDER BY rank_no ASC
LIMIT 20;
```

핵심은 큰 테이블에서 계산하지 않고, 작은 랭킹 테이블에서 document_srl 목록만 빠르게 가져오는 것입니다.

3. 모듈 형태

애드온 하나로 모든 페이지에 끼어드는 방식보다는, 데이터와 설정을 가진 독립 모듈이 낫습니다.

```
modules/popular/
 ├── conf/module.xml
 ├── schemas/popular_config.xml
 ├── schemas/popular_candidate.xml
 ├── schemas/popular_rank.xml
 ├── src/controllers/
 ├── src/models/
 ├── views/
 └── scripts/rebuild.php 또는 Queue 작업
```

Rhymix 는 모듈의 module.xml 에서 액션별 짧은주소 라우트를 정의할 수 있으므로, /board/hot, /board/best 같은 형태로 연결하기 좋습니다.

4. 게시판 하단 버튼은 거의 부하가 없어야 함

이미지의 HOT 게시물, 개념글, BEST 인기글 버튼 형태라면 가장 가벼운 방식은 단순 링크입니다.

```
<a href="{getUrl('', 'mid', $mid, 'act', 'dispPopularList', 'period', 'today')}">
  HOT 게시물
</a>
```

이 경우 일반 게시판 목록 페이지에서는 인기글 목록을 가져오지 않습니다. 사용자가 버튼을 눌렀을 때만 인기글 목록 액션이 실행됩니다.

하단에 “인기글 5 개 미리보기”까지 보여주고 싶다면, 그 부분은 HTML fragment 캐시로 처리합니다. 예를 들어 popular:html:board:123:today:5 같은 키로 1~5 분 캐시하면 됩니다.

5. 추천 테이블 구조

5.1 rx_popular_config

게시판별 설정 테이블입니다.

```
module_srl
enabled
scope              board / global
period             today / week / month
min_vote
min_comment
min_read
```

```
weight_vote
weight_comment
weight_read
decay_type
list_count
refresh_interval
```

5.2 rx_popular_candidate

인기글 후보 테이블입니다. 모든 글을 넣을 필요는 없고, 최근 글 또는 반응이 있는 글만 넣습니다.

```
document_srl       PK
module_srl
category_srl
regdate
status
is_notice
voted_count
comment_count
readed_count
view_delta
score
dirty
updated_at
```

권장 인덱스 방향은 다음과 같습니다.

```
(document_srl)
(module_srl, regdate, document_srl)
(module_srl, dirty, updated_at)
(module_srl, score, document_srl)
```

5.3 rx_popular_rank

실제로 화면에서 읽는 랭킹 테이블입니다.

```
scope              board / global
module_srl         board scope 면 게시판 번호, global 이면 0
period             today / week / month
rank_no
document_srl
score
generated_at
```

권장 인덱스 방향은 다음과 같습니다.

```
(scope, module_srl, period, rank_no)
```

```
(scope, module_srl, period, score, document_srl)
```

화면에서는 거의 이 테이블만 읽습니다. 그래서 빠릅니다.

6. 갱신 방식

6.1 추천, 댓글, 글 작성은 후보만 갱신

글 작성, 추천, 댓글, 삭제 같은 이벤트에서는 랭킹 전체를 재계산하지 않고 후보 테이블만 갱신합니다.

```
글 작성 후:
  rx_popular_candidate 에 후보 등록

추천 후:
  해당 document_srl 의 voted_count 갱신
  dirty = Y

댓글 작성/삭제 후:
  해당 document_srl 의 comment_count 갱신
  dirty = Y

글 삭제/휴지통/비공개 전환:
  candidate 와 rank 에서 제외

조회수 증가:
  매번 DB 랭킹 재계산 금지
  캐시 카운터에 delta 만 쌓고 주기적으로 flush
```

가장 조심할 부분은 조회수입니다. 조회수는 방문마다 발생할 수 있으므로, 조회수 증가마다 랭킹 테이블을 업데이트하면 인기글 기능이 오히려 부하를 만듭니다. 조회수는 기본 가중치를 낮게 두거나, Cache::incr()로 view_delta만 쌓아두고 1 분 또는 5 분마다 후보 테이블에 반영하는 게 좋습니다.

6.2 랭킹 계산은 Queue 또는 cron 으로

랭킹 재계산은 웹 요청 중에 하지 말고 백그라운드에서 처리합니다. 예를 들어 3~5 분마다 다음 작업을 돌립니다.

```
popular.rebuild today
popular.rebuild week
popular.flushViewDelta
```

랭킹 재계산 로직은 대략 이런 방향입니다.

```
SELECT document_srl,
       voted_count,
       comment_count,
       readed_count,
       regdate
```

```
FROM rx_popular_candidate
WHERE module_srl = ?
  AND regdate >= ?
  AND status = 'PUBLIC'
  AND is_notice = 'N'
  AND (
    voted_count >= ?
    OR comment_count >= ?
    OR readed_count >= ?
  )
ORDER BY score DESC, document_srl DESC
LIMIT 100;
```

그 결과를 rx_popular_rank 에 rank_no = 1, 2, 3... 형태로 저장합니다.

정렬할 때는 score DESC 만 쓰지 말고 document_srl DESC 또는 list_order ASC 같은 고유한 보조 정렬 기준을 넣는 것이 좋습니다. 점수가 같은 글들이 있을 때 고유하지 않은 기준만으로 정렬하면 페이지 이동 시 중복 노출이나 순서 흔들림이 생길 수 있습니다.

7. 점수 공식

초기 버전은 단순한 점수 공식으로도 충분합니다.

```
score =
  voted_count    * 10
+ comment_count  * 3
+ log(readed_count + 1) * 1
- age_hours      * 0.2
```

또는 “베스트글”은 점수 계산 없이 조건형으로 가도 됩니다.

```
오늘의 베스트:
  추천수 10 이상
  댓글수 5 이상
  최근 24시간 글

주간 인기글:
  최근 7일 글
  score 높은 순
```

“인기글”과 “베스트글”은 성격이 다릅니다.

```
베스트글:
  추천수 기준의 안정적인 명예 목록

인기글:
  최근 반응이 빠르게 붙는 글
  시간 감쇠 적용
```

처음부터 복잡한 알고리즘을 넣기보다, 추천수 + 댓글수 + 시간 조건으로 시작하고 조회수는 나중에 넣는 편이 부하와 품질 면에서 안전합니다.

8. 권한 처리

게시판별 인기글은 비교적 쉽습니다. 현재 게시판 안의 글만 보여주면 되므로, 해당 게시판을 볼 수 있는 사용자는 인기글도 볼 수 있다고 보면 됩니다.

문제는 전체 인기글입니다. 전체 인기글에 비밀 게시판, 회원 전용 게시판, 관리자 게시판 글이 섞이면 안 됩니다. 가장 효율적인 선택은 다음 중 하나입니다.

1. 전체 인기글에는 공개 게시판만 포함
2. 게시판별 설정에서 “전체 인기글 포함 여부”를 명시
3. 권한 그룹별 랭킹 캐시를 따로 생성

3 번이 가장 정확하지만 복잡하고 캐시 수가 늘어납니다. 무료 배포용 모듈이라면 1 번 또는 2 번이 현실적입니다.

9. 실제 화면 출력 흐름

9.1 게시판 목록 페이지

```
1. 일반 게시판 목록 출력
2. 하단에 HOT/인기글 버튼만 출력
3. 미리보기 영역이 켜져 있으면 Cache 에서 HTML 읽기
4. 캐시 없으면 rx_popular_rank 에서 document_srl 5~10 개만 읽기
5. 문서 정보는 한 번의 쿼리로 가져오기
6. HTML 캐시 저장
```

9.2 인기글 페이지

```
1. rx_popular_rank 에서 rank_no 순서로 document_srl 조회
2. document_srl 목록을 한 번에 문서 테이블에서 조회
3. 원래 rank_no 순서대로 재정렬
4. 출력
```

여기서 피해야 할 것은 document_srl 마다 getDocument()를 반복 호출하는 N+1 구조입니다. 20 개를 보여주면 문서 20 개를 각각 조회하는 식이 되어버립니다.

10. 추천 MVP

처음 무료 배포용으로 만든다면 기능을 과하게 늘리지 말고 다음 정도로 줄이는 것이 좋습니다.

```
기능:
  게시판별 인기글
  전체 인기글은 공개 게시판만
  오늘 / 주간 / 월간
```

```
  추천수, 댓글수 기준
  조회수는 옵션, 기본 OFF

갱신:
  추천/댓글/글작성/삭제 트리거로 후보 dirty 처리
  3~5분마다 Queue 또는 cron으로 rank 재계산
  화면은 rank table + Cache만 읽기

UI:
  게시판 하단 버튼
  선택 시 인기글 목록 페이지
  옵션으로 하단 미리보기 5개
```

이렇게 만들면 일반 게시판 목록 페이지의 추가 부하는 거의 버튼 HTML 출력 수준이고, 인기글 영역을 보여주더라도 대부분 캐시 hit가 나옵니다.

11. 서버 부하 관점의 최종 순위

```
가장 좋음:
  랭킹 테이블 사전 계산 + HTML/결과 캐시 + Queue/cron 갱신

괜찮음:
  1~5분 TTL 캐시 + 캐시 만료 시에만 documents에서 계산

나쁨:
  게시판 목록을 볼 때마다 documents에서 인기글 실시간 정렬

가장 나쁨:
  모든 페이지 display 트리거에서 전체 인기글 계산
  조회수 증가마다 랭킹 재계산
  document_srl별 개별 조회 반복
```

개발 방향은 “실시간 인기글 계산 모듈”이 아니라 “인기글 랭킹을 미리 만들어두는 모듈”로 잡는 것이 맞습니다. 이 방식이 커뮤니티 규모가 커져도 가장 안전합니다.

참고 링크

Rhymix documents schema: https://github.com/rhymix/rhymix/blob/master/modules/document/schemas/documents.xml
Rhymix Router guide: https://github.com/rhymix/rhymix-docs/blob/master/ko/plugin/router/router.md
Rhymix Cache API: https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Cache.md
Rhymix Queue API: https://github.com/rhymix/rhymix-docs/blob/master/ko/reference/framework/Queue.md
Rhymix crontab guide: https://github.com/rhymix/rhymix-docs/blob/master/ko/misc/crontab.md
Rhymix trigger list draft: https://damoang-users.github.io/rhymix-guide/reference/trigger-list.html
Rhymix Q&A sorting stability example: https://rhymix.org/qna/1805591