Bonus Task - Displaying Trending Articles

This implementation establishes a real-time article popularity tracking system using a dedicated counter column (`ART_STARREDCOUNT_N`) in the `T_ARTICLE` table. The solution employs a B-Tree index on this column with descending sort order to facilitate efficient O(log n) retrieval operations. This indexing strategy eliminates the need for computationally expensive aggregation queries or joins that would otherwise be required to calculate popularity metrics at request time.

The system maintains transactional integrity through atomic increment operations when users star content, ensuring accurate popularity metrics even under high concurrency. By leveraging database-native indexing capabilities rather than in-memory processing, the implementation achieves sub-logarithmic query performance for trending content retrieval while minimizing application server resource utilization.

Code Changes:

Addition of Column & Indexing in `T_ARTICLE`

```
create cached table T_ARTICLE ( ART_ID_C varchar(36) not null, ART_IDFEED_
create index IDX_STARRED_COUNT on T_ARTICLE ( ART_STARREDCOUNT_N
```

/trending endpoint in StarredResource

```
@GET
    @Path("trending")
    @Produces(MediaType.APPLICATION_JSON)
    public Response getTrending() throws JSONException {
        if (!authenticate()) {
            throw new ForbiddenClientException();
        }

        // Get the articles
        ArticleDao articleDao = new ArticleDao();
        List<ArticleDto> articles = articleDao.getTrending();

        // Build the response
        JSONObject response = new JSONObject();

        List<JSONObject> articlesJson = new ArrayList<JSONObject>();
        for (ArticleDto article : articles) {
            articlesJson.add(article.asJson());
        }
        response.put("articles", articlesJson);
        return Response.ok().entity(response).build();
    }
```

Quering the Database in ArticleDAO

```
public List<ArticleDto> getTrending(){
        EntityManager em = ThreadLocalContext.get().getEntityManager();
        String sql="select a.ART_ID_C, a.ART_IDFEED_C, a.ART_URL_C, a.ART_GU
                    "a.ART_CREATOR_C, a.ART_DESCRIPTION_C, a.ART_COMMENT
                    "a.ART_ENCLOSUREURL_C, a.ART_ENCLOSURELENGTH_N, a.A
                    "a.ART_PUBLICATIONDATE_D, a.ART_CREATEDATE_D, a.ART_S

        Query q = em.createNativeQuery(sql);
        q.setMaxResults(5);

        // return q.getResultList();
        List<Object[]> resultList = q.getResultList();
        List<ArticleDto> articleList = new ArrayList<ArticleDto>();

        for (Object[] result : resultList) {
            int i = 0;
            ArticleDto article = new ArticleDto();
            article.setId((String) result[i++]);
            article.setFeedId((String) result[i++]);
            article.setUrl((String) result[i++]);
            article.setGuid((String) result[i++]);
            article.setTitle((String) result[i++]);
            article.setCreator((String) result[i++]);
            article.setDescription((String) result[i++]);
            article.setCommentUrl((String) result[i++]);
            article.setCommentCount((Integer) result[i++]);
            article.setEnclosureUrl((String) result[i++]);
            article.setEnclosureCount((Integer) result[i++]);
            article.setEnclosureType((String) result[i++]);
            article.setPublicationDate((Date) result[i++]);
            article.setCreateDate((Date) result[i++]);
            article.setStarredCount((Integer) result[i++]);
            articleList.add(article);
        }

        return articleList;
    }
}
```

Process Flow

1. **Article Creation**:
    - New article is created in the system
    - `ART_STARREDCOUNT_N` is initialized to 0
2. **User Interaction**:
    - When a user stars an article, an API call is made
    - The system records the star relationship between user and article
    - `ART_STARREDCOUNT_N` for that article is incremented by 1
3. **Trending Articles Retrieval**:
    - User clicks "Trending" button in the UI
    - Frontend calls `/trending` endpoint
    - Backend queries `T_ARTICLE` table, retrieving 5 articles with highest `ART_STARREDCOUNT_N`

- Query leverages B-Tree index for efficient retrieval (O(log n))
- Results are returned to frontend
- UI displays the trending articles to user

Design Patterns

Standard Principles followed through the codebase like separation of concerns with a DAO , DTO , Entity File have been followed.