Design News Feed System (Part 1)

If you have followed our previous posts (http://blog.gainlo.co/index.php/category/system-design-interview/) on system design interview questions, you might be surprised at how common news feed system is.

No matter whether you are building Twitter, Instagram or Facebook, you will need some sort of news feed systems to display updates from follows/friends.

In fact, there are a bunch of interesting details about news feed like how to rank feeds, how to optimize publishing etc.. So in this post, I'll cover this popular question – design news feed system.

Question

To make it simple, let's focus on designing news feed (https://www.google.com/search?q=what+is+facebook+news+feed&oq=what+is+facebook+news+feed&aqs=chrome..69i57j0l2j69i60j0l2.4023j0j1&sourceid=chrome&ie=UTF-8) system for Facebook since different products have different requirements.

To briefly summarize the feature, when users go to their home pages, they will see updates from their friends based on particular order. Feeds can contain images, videos or just text and a user can have a large number of friends.

So how can you design such news feed system from scratch?

Subproblems

If you haven't thought about this problem, it's better to solve it by yourself before reading the rest of the post. Although there's no such a thing as standard answer, you can still learn a lot by comparing your solution with others.

Here we go. As we said before (http://blog.gainlo.co/index.php/2015/10/22/8-things-you-need-to-know-before-system-design-interviews/), when facing such large and vague system design question, it's better to have some high-level ideas by dividing the big problem into subproblem.

For a news feed system, apparently we can divide it into front-end and backend. I'll skip the front-end as it's not that common in system design interviews. For backend, three subproblems seem critical to me:

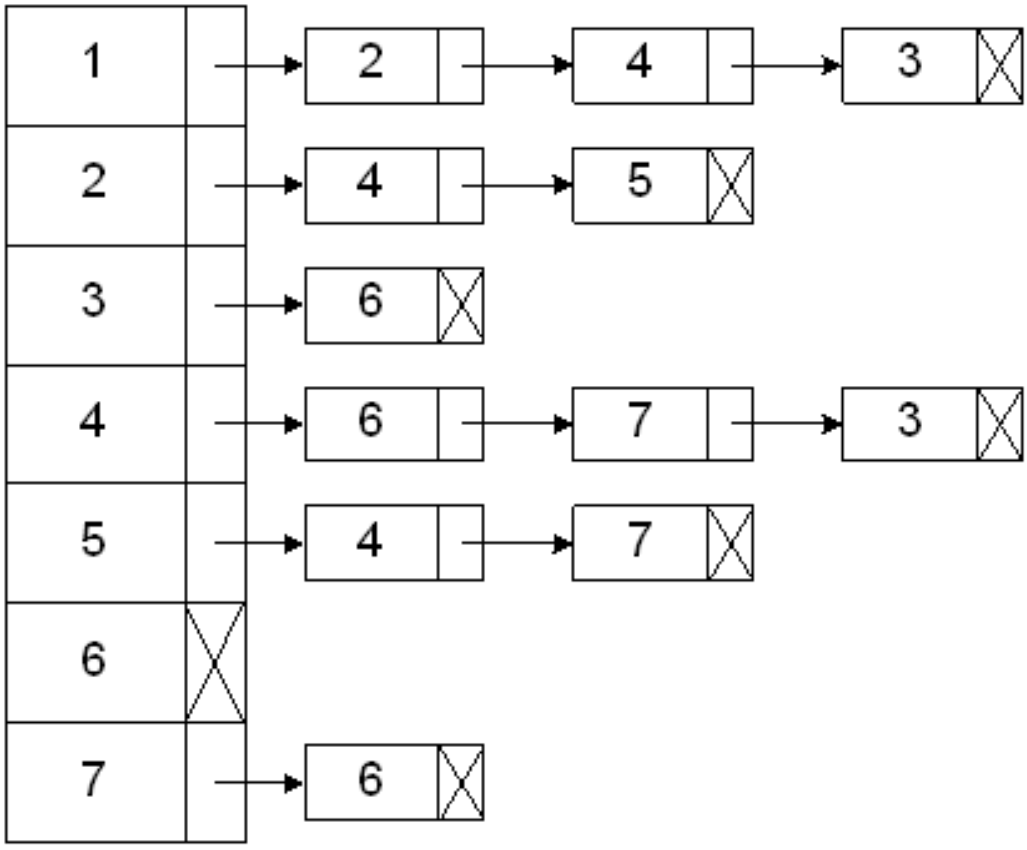
- Data model. We need some schema to store user and feed object. More importantly, there are lots of trade-offs when we try to optimize the system on read/write. I'll explain in details next.
- Feed ranking. Facebook is doing more than ranking chronologically.
- Feed publishing. Publishing can be trivial when there're only few hundreds of users. But it can be costly when there are millions or even billions of users. So there's a scale problem here.

Data model

There are two basic objects: user and feed. For user object, we can store userID, name, registration date and so on so forth. And for feed object, there are feedId, feedType, content, metadata etc., which should support images and videos as well.

If we are using a relational database (https://en.wikipedia.org/wiki/Relational_database), we also need to model two relations: user-feed relation and friend relation. The former is pretty straightforward. We can create a *user-feed table* that stores userID and corresponding feedID. For a single user, it can contain multiple entries if he has published many feeds.

For friend relation, adjacency list (https://en.wikipedia.org/wiki/Adjacency_list) is one of the most common approaches. If we see all the users as nodes in a giant graph, edges that connect nodes denote friend relation. We can use a *friend table* that contains two userIDs in each entry to model the edge (friend relation). By doing this, most operations are quite convenient like fetch all friends of a user, check if two people are friends.



Data model – continue

In the design above, let's see what happens when we fetch feeds from all friends of a user.

The system will first get all userIDs of friends from *friend table.* Then it fetches all feedIDs for each friend from *user-feed table*. Finally, feed content is fetched based on feedID from *feed table.* You can see that we need to perform 3 joins (https://en.wikipedia.org/wiki/Join_(SQL)), which can affect performance.

A common optimization is to store feed content together with feedID in *user-feed table* so that we don't need to join the *feed table* any more. This approach is called denormalization (https://en.wikipedia.org/wiki/Denormalization), which means by adding redundant data, we can optimize the read performance (reducing the number of joins).

The disadvantages are obvious:

- Data redundancy. We are storing redundant data, which occupies storage space (classic time-space trade-off).
- Data consistency. Whenever we update a feed, we need to update both *feed table* and *user-feed table*. Otherwise, there is data inconsistency. This increases the complexity of the system.

Remember that there's no one approach always better than the other (normalization vs denormalization). It's a matter of whether you want to optimize for read or write.

Ranking

The most straightforward way to rank feeds is by the time it was created. Obviously, Facebook is doing more than that. "Important" feeds are ranked on top.

Before jumping to the ranking algorithm, I'd usually like to ask why do we want to change the ranking? How do we evaluate whether the new ranking algorithm is better? It's definitely impressive if candidates come up with these questions by themselves.

The reason to have better ranking is not that this seems the right thing to do. Instead, everything should happen for a reason. Let's say there are several core metrics we care about, e.g. users stickiness, retention (https://en.wikipedia.org/wiki/Customer_retention), ads revenue etc.. A better ranking system can significantly improve these metrics potentially, which also answers how to evaluate if we are making progress.

So back to the question – how should we rank feeds? A common strategy is to calculate a feed score based on various features and rank feeds by its score, which is **one of the most common approaches for all ranking problems.**

More specifically, we can select several features that are mostly relevant to the importance of the feed, e.g. share/like/comments numbers, time of the update, whether the feed has images/videos etc.. And then, a score can be computed by these features, maybe a linear combination. This is usually enough for a naive ranking system.

Summary

I didn't expect to have so many details before writing this post and I had to cut the post in half.

In the second part, we're going to cover more details about ranking, scalability issues with feed publishing and other interesting topics.

If you find this post helpful, I would really appreciate if you can share it with your friends. Also you can check more system design interview questions (http://blog.gainlo.co/index.php/category/system-design-interview-questions/) and analysis here.

(http://www.facebook.com/sharer.php?u=http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/); 8 (http://twitter.com/share?url=http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/&text=Design+News+Feed+System+%28Part+1%29+); (http://www.linkedin.com/shareArticle?mini=true&url=http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/); 28 (http://reddit.com/submit?url=http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/&title=Design News Feed System (Part 1)); 3

Related Posts:

1. **Design News Feed System (Part 2) (http://blog.gainlo.co/index.php/2016/04/05/design-news-feed-system-part-2/)**
2. **How to Design Twitter (Part 2) (http://blog.gainlo.co/index.php/2016/02/24/system-design-interview-question-how-to-design-twitter-part-2/)**
3. **How to Design Twitter (Part 1) (http://blog.gainlo.co/index.php/2016/02/17/system-design-interview-question-how-to-design-twitter-part-1/)**
4. **How to Design Youtube (Part I) (http://blog.gainlo.co/index.php/2016/10/22/design-youtube-part/)**

3 thoughts on “Design News Feed System (Part 1)”

MARYNA (HTTPS://MCHERNYAVSKA.WORDPRESS.COM/)
April 14, 2016 at 8:25 pm (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/#comment-622)

I don’t understand why you should have both user-feeds and feeds table. What is the purpose of storing user-feed relationships separately? If I understand correctly the feed is one post. It always belongs to one user. A user can have many posts. It is a clear one-to-many relationship, where the feed always has a user ID as its property. So, you have users, user-friends, user-feeds; first query gets friend IDs, second query (or join) gets the feeds where user IDs are in the friend IDs list. Am I missing something?

o.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/?replytocom=622#respond)

JAKE
April 15, 2016 at 10:14 pm (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/#comment-628)

Hi Maryna,

You can make user-feeds and feeds table together when it’s one-to-many relationship. I would say making them separate have some advantages. For instance, it’s trivial to expand to many-to-many relationship. If you have more user-feed relationships like view permission, we can reuse the same table.

.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/?replytocom=628#respond)

DAVE WATSON
March 2, 2017 at 2:38 am (http://blog.gainlo.co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/#comment-5847)

The paragraph about denormalizing is highly inaccurate and harmful – I would advise the author to do some research on relational DB performance and denormalization as a technique. This kind of denormalization will not improve performance and in fact only hinder it even at moderate scale. Joins are not expensive, especially not simple joins like in this example.

).co/index.php/2016/03/29/design-news-feed-system-part-1-system-design-interview-questions/?replytocom=5847#respond)

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

< How To Design Google Docs (http://blog.gainlo.co/index.php/2016/03/22/system-design-interview-question-how-to-design-google-docs/)

5 Things To Do After You Failed an Interview > (http://blog.gainlo.co/index.php/2016/04/01/5-things-to-do-after-you-failed-an-interview/)

CATEGORIES

Coding Interview Questions (http://blog.gainlo.co/index.php/category/coding-interview-questions/)

Common Pitfalls (http://blog.gainlo.co/index.php/category/common-pitfalls/)

Common Questions (http://blog.gainlo.co/index.php/category/common-questions/)

Facebook Interview Questions (http://blog.gainlo.co/index.php/category/facebook-interview-questions/)

Mock Interview Thoughts (http://blog.gainlo.co/index.php/category/mock-interview-thoughts/)

Others (http://blog.gainlo.co/index.php/category/others/)

System Design Interview Questions (http://blog.gainlo.co/index.php/category/system-design-interview-questions/)

The Complete Guide to Google Interview Preparation (http://blog.gainlo.co/index.php/category/google-interview-preparation/)

Uber Interview Questions (http://blog.gainlo.co/index.php/category/uber-interview-questions/)