**NewsFeed System HLD & DB Schema Design**

**✅ Requirements**

* Generate personalized home page feed for each user
* Show latest posts (images/videos) by followees
* Support ranking (e.g., recency, engagement)
* Support real-time updates or periodic polling

**🧱 DB Schema Design**

**📄 FeedItem**

CREATE TABLE FeedItems (

id BIGINT PRIMARY KEY,

userId BIGINT, -- Who created the post

photoId BIGINT, -- Media reference

createdAt TIMESTAMP,

likes INT

);

**🔗 UserFollow**

CREATE TABLE UserFollow (

followerId BIGINT,

followeeId BIGINT,

PRIMARY KEY (followerId, followeeId)

);

**🔍 Feed Retrieval Approaches**

**📥 Pull Model (Fan-Out on Load)**

SELECT \* FROM FeedItems

WHERE userId IN (

SELECT followeeId FROM UserFollow WHERE followerId = :userId

)

ORDER BY createdAt DESC

LIMIT 100;

* ✅ Easy to implement
* ❌ High DB load, slow response time

**📤 Push Model (Fan-Out on Write)**

* Generate feed ahead of time and store it (per user)
* On every post, push to followers' cached feed

Map<UserId, TreeMap<FeedItemId, FeedItem>> userFeeds;

**🔁 Hybrid Approach**

* **For regular users:** Push model (better UX)
* **For celebrities (high followers):** Pull model (saves infra)

**⚙️ Caching Strategy**

* Store top 100 feed items per user in **Redis / Memcached**
* Update cache when:
  + User opens app (refresh)
  + New content posted
* TTL (Time-to-Live) based cache expiry

**📶 Real-Time Feed Delivery**

**1. WebSocket / Long Polling**

* For push-based near real-time feed updates

**2. Client Pull**

* Every x seconds, client polls feed endpoint

**🧩 Partitioning Strategy**

**✅ By userId**

* Pros: Each user’s feed is grouped = fast access
* Best for caching and scalability

**❌ By feedId**

* Poor for user-specific queries, not recommended

**⚖️ Summary Table**

| **Feature** | **Pull Model** | **Push Model** |
| --- | --- | --- |
| Pros | Lightweight backend | Real-time, less client effort |
| Cons | Slower feed, wasteful polling | Write-heavy, infra-intensive |
| Used for | Celebrities | Normal users |

**📌 Final Notes**

* Hybrid push-pull feed delivery offers best performance
* Caching & sharding by userId ensures scalability
* Use ranking algorithms (based on createdAt, likes) for better feed
* Real-time updates enhance UX but increase infra cost

Ideal architecture for modern social platforms needing fast, relevant, and scalable news feeds.