**Instagram HLD & DB Schema Design**

**✅ Functional Requirements**

1. Users can post, delete, view photos
2. Users can follow/unfollow other users
3. System generates news feed for users (posts by followees)

**⚙️ Non-Functional Requirements**

* High Availability
* Low latency feed (target: < 200ms)
* Eventual consistency acceptable
* System is read-heavy

**📦 Storage & Infrastructure**

* Store images in **AWS S3** (or similar object storage)
* Use **CDN** (like Cloudflare, Akamai) to speed up image delivery
* Use SQL DB for relational data (e.g., user & follow relationships)
* Consider **replication** and **sharding** strategies

**🧠 Estimation**

* **Users:** 500 million
* **DAU:** 1 million
* **Uploads/day:** 2 million images/day
* **Image size:** 200KB → ~400GB/day
* **10-year storage:** ~1460 TB

**🗄️ SQL DB Schemas**

**👤 Users**

CREATE TABLE Users (

id BIGINT PRIMARY KEY,

userName VARCHAR,

email VARCHAR,

phone VARCHAR,

dob DATE,

followers BIGINT,

followees BIGINT,

totalPosts BIGINT,

creationDate TIMESTAMP,

lastLogin TIMESTAMP

);

**📸 Photos**

CREATE TABLE Photos (

id BIGINT PRIMARY KEY,

userId BIGINT REFERENCES Users(id),

place VARCHAR,

path TEXT, -- S3 URL

creationDate TIMESTAMP

);

**🔗 UserFollows**

CREATE TABLE UserFollows (

followerId BIGINT,

followeeId BIGINT,

PRIMARY KEY (followerId, followeeId)

);

**🧱 Design Considerations**

**✅ SQL vs NoSQL**

* SQL better for joins (User ↔ Posts ↔ Follows)
* NoSQL lacks join support, but easier to shard

**📊 CQRS Pattern**

* Separate read and write DBs for better scalability

**🌐 Load Balancing**

* Round Robin LB for stateless requests
* Consistent hashing not needed unless session affinity required

**🧩 Partitioning Strategies**

* **By userId:** Keeps all user images together → good for locality, but hotspot issue for influencers
* **By imageId:** Equal distribution, but can increase lookup latency

**📈 Data Size Estimations**

* User metadata: 500M users \* ~100B = ~50GB
* Images: 2M/day × 200KB = ~400GB/day → 1.1TB/year
* User follow graph: ~5TB (based on 500 followees avg)

**🛡️ Optimization & Reliability**

* Use RateLimiter to protect backend from abuse
* Use write-ahead queues for image processing pipeline
* Deploy edge caching for feed content
* Support offline viewing with local storage for limited posts

This system handles massive scale with read-optimization, partitioning strategies, and fast delivery through CDNs — ideal for social media use cases like Instagram.