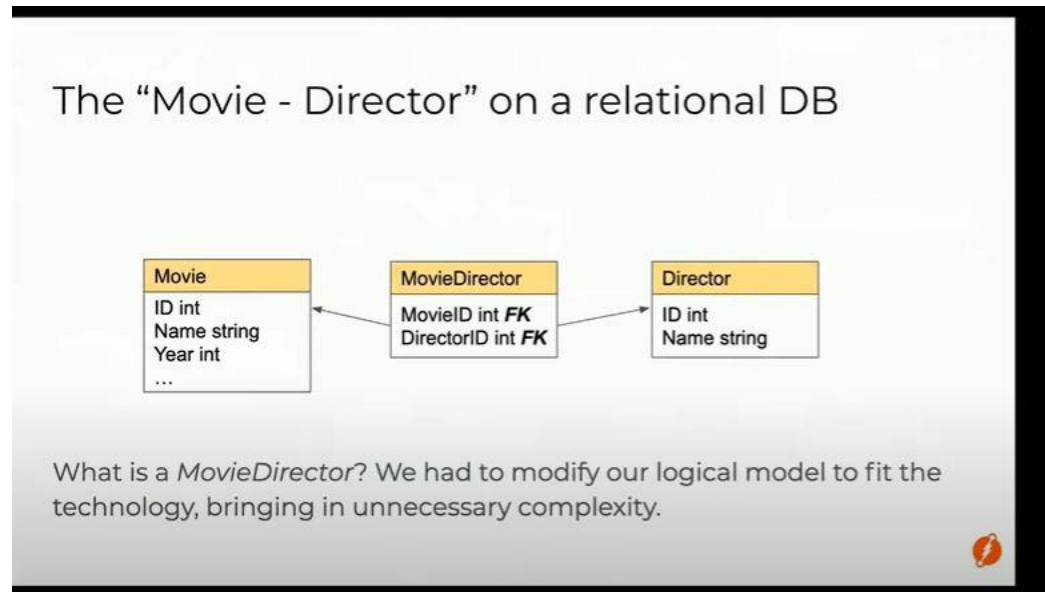


Twitter User -> Followers

A.RDBMS



B. NoSQL (key-value) Database - HBase, Cassandra, Amazon DynamoDB, BigTable + In-Memory (key-value) Database - Redis

1. NoSQL Key Value Database:

User Database:

(key) UID 1 - Sachin Tendulkar -> Details of Sachin (Object)

-> List of Follower keys IDs (1, 2, 3)

(key) UID 2 - Ramesh - Details of Ramesh (Object)

(key) UID 3 - Suresh - Details of Suresh (Object)

2. To achieve Low Latency:

Cache the below in Redis (In-Memory key value DB)

Followers Database:

(key) Sachin Tendulkar - List of Follower keys IDs (1, 2, 3)

Now this allows us to get the followers in $O(1)$

The list can fit in one Redis instance.

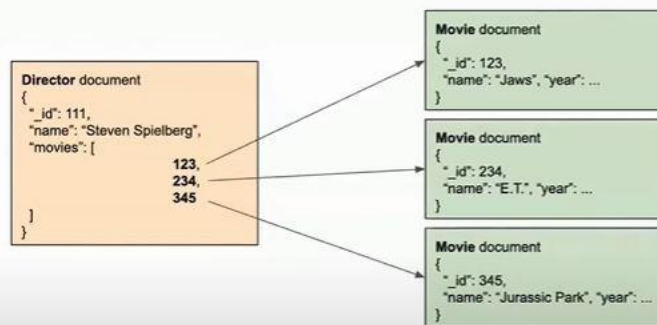
3. Distribute Tasks (like send notification/update timeline of all followers)

* Each task processes a range of follower ids (start to end key) -> it can be a sorted list [or]

The coordinator can split multiple tasks and tell each one with a range to process.

* The coordinator can split into multiple sub lists and pass sub list to each task.

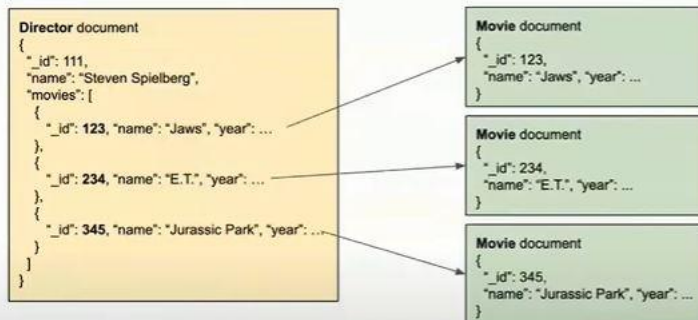
The "Movie - Director" on a no-SQL DB (A)



Fetching all the names of the movies directed by a director requires $n+1$ queries.



The “Movie - Director” on a no-SQL DB (C)



Now we can fetch all movies for a director in a query ... but we might easily lose consistency.

C. Graph DB on top of Key-Value Database:

UID for every object + Predicate = Value or List of other objects

1 + name = Sachin

1 + followers = 2,3

2 + name = Ramesh

3 + name = Suresh

[Dgraph: The Graph Database written in Go - YouTube](#)

UID – Twitter Snowflake

Dgraph data modeling

Predicates are always attached to UIDs.

We associate values and objects to keys composed by UID + predicate

Keys	Values
0x1:<has name>	"Jaws"
0x1:<was recorded in the year>	1975
0x1:<was directed by>	0x2
0x2:<has name>	"Steven Spielberg"

Sometimes a value can be an array of UIDs or values.

