Sharding: Ssystem Ddesign Iinterview Cconcepts (7 of 9)

SOFTWARE ENGINEERING FEB. 09, 2021

If you want to succeed in system design interviews for a tech role, then you'll probably need to understand sharding and when to use it within the context of a larger system.

Sharding is essentially the horizontal scaling of a database system that is accomplished by breaking the database up into smaller "shards", which are -- separate database servers that all contain a subset of the overall dataset.

This is just a high-level definition, so to fill in the gaps we've put together the below guide that will walk you through sharding and how you should approach it during a system design interview. Here's what we'll cover:

1. Sharding Bbasics
2. Approaches to Ssharding
3. Example Ssharding Qquestions
4. System Ddesign Iinterview Ppreparation

Cracking system design interviews

Subscribe to our **Free 4-day System Ddesign Iinterview email course**. Highlights include: example questions (+ answers), step-by-step approaches, and concepts you need to know. We'll also send you the occasional special offer.

Subscribe

Commented [AI1]: Please excuse the layout of this document. I had your webpage pulled up while editing so I could visually see the piece and how it worked with the page before me. This format just makes it easier for you to physically see all my suggestions, comments, and corrections. Before I complete the final piece tomorrow, please let me know if this format is still alright with you. Thanks!
-Lina

Commented [AI2]: As a viewer of the blog, part of me would like this term to be italicized throughout the piece. It grabs my attention and reminds readers that the piece and the information revolve around this concept. It is a personal preference, however, so it is ultimately up to you.

Commented [AI3]: Although not grammatically incorrect, the fluidity of your sentence and the emphasis on the concise definition is more definitive in this format.

Commented [AI4]: Suggested rewrite:

This definition can be unnecessarily complicated, so we've made it easier with a guide to get you through it. Here's what we cover:

Commented [AI5]: When doing bullet points, its standard to keep your titles in the same formatting. So for each of your section headings below you capitalized each of the words, therefore each of these points needs to mimic that as well.

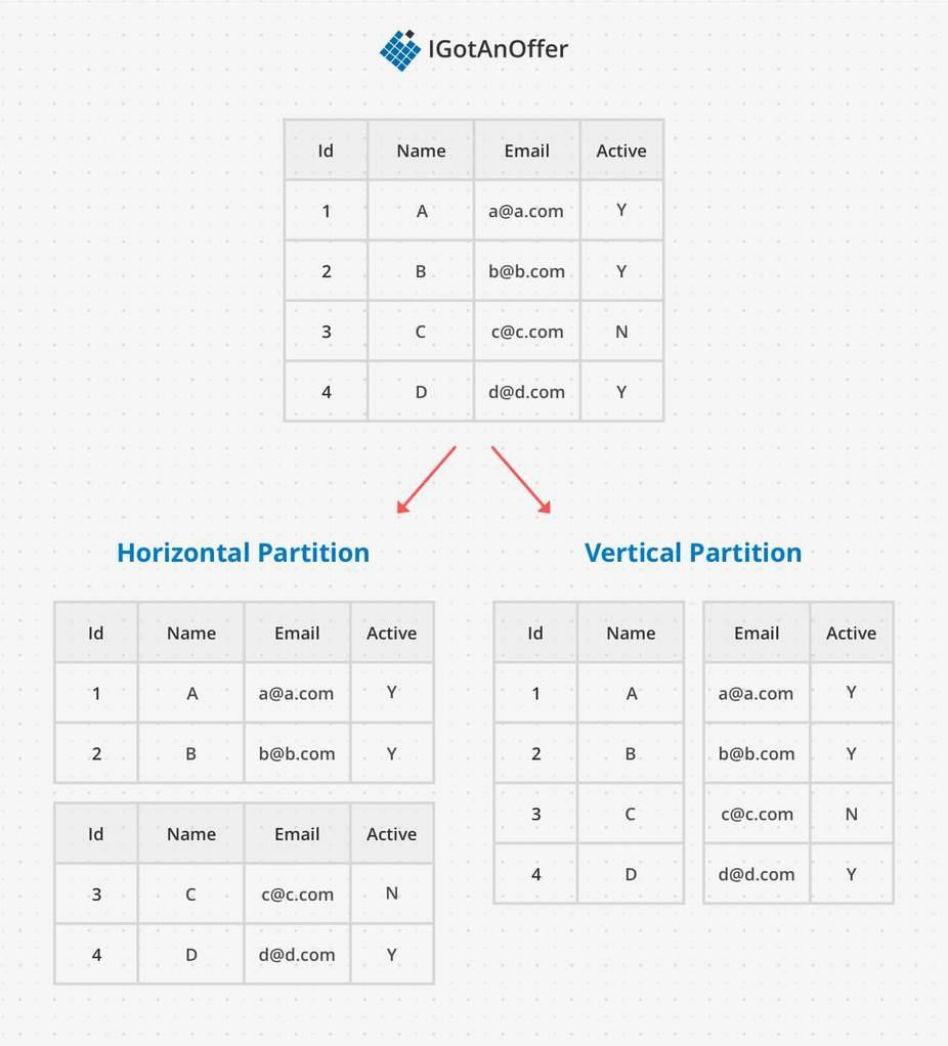
Commented [AI6]: Capitalize these if it is the legit title (name) of the course. If it's not the technical name for your course, then please disregard the capitalization.

1. Sharding Basics

Modern software services increasingly collect and use more data than can fit on a single machine. The capacity of the database server can be increased, but eventually runs into a physical limit. The alternative is splitting up data across a cluster of database servers.

Commented [AI7]: This is well put. I checked to make sure it was correct and then also read from the standpoint of someone who has minimal knowledge on the subject.

Database sharding splits up data ~~in a particular way~~ so that the ~~data~~ access patterns can remain as efficient as possible. A **shard** is a **horizontal partition**, meaning the database table is ~~split up~~divided by ~~drawing~~ a horizontal line between rows. In contrast, ~~This is in contrast to a~~ a **vertical partition** is~~,~~ where partitions are ~~made~~ divided between columns.



Although ~~The~~the underlying goal of sharding is to increase database capacity, ~~but~~ another ~~effect~~ result is ~~that~~ the database can handle more traffic because any one server in the cluster only has to respond to a fraction of the total requests.

There are a couple of key features a sharded architecture needs to be efficient. We'll go over how it's implemented with shard keys~~,~~ and the importance of denormalizing data. Then we'll cover when to use and when not to use sharding. Let's jump in.

Commented [AI8]: I will assume the audience you are addressing will know this term is specific to a database and its computation ability to improve the reading of its information in general.
If this is NOT the case, then you will need to define this term briefly before moving on.

1.1 How Ssharding Wwworks

Each row is assigned a **shard key** that maps it to the **logical shard** it can be found on. More than one logical shard can be located on the same **physical shard**, but a logical shard can't be split between database nodes ~~physical shards.~~

When creating a sharded architecture, the goal is to have ~~many~~ mulitple small shards to create an even distribution of data across the nodes. This prevents **hotspots** from overwhelming any one of the nodes and produces fast response times for all nodes.

Sharding can be implemented either at the application level or the database level. At this point, most databases support sharded architectures, with the notable exception of ~~PostgresQL~~PostgreSQL, one of the top relational databases.

1.2 Denormalization

When implementing a sharded database architecture, you need to ~~take special consideration~~consider ~~about~~ how the relational data model spans shards. Relational data models are optimized for data with many cross-table relationships. If shards aren't isolated, the cluster will ~~spend a lot of~~waste time on multi-shard queries and upholding multi-shard consistency.

This process of making sure there aren't relational constraints between different shards is called **denormalization**. Denormalization is achieved by duplicating data (which adds write complexity) and grouping data so that it can be accessed in a single row. Notably, denormalized data is different from **non-normalized** data where relationships between data are unknown rather than located on a single table.

As an example, let's ~~think about~~consider where to store data for messages between two friends. Both users want fast responses, but what if their user records are split up onto different shards? The answer? We can store the messages in both places. This shard division increases write time slightly~~,~~ because it means the data is written twice.~~,~~ b~~B~~ut it also ensures the message data is fast to look up regardless of which ~~-~~friend is checking their messages.

Commented [AI9]: I double checked to make sure these were the same terms and I believe it might be technically more correct to use nodes for this final statement.

Commented [AI10]: As you write these pieces, make sure you are mindful of possible copywrite issues. For example, I plugged this first sentence straight into google and it pulled up verbatim what you wrote.

Commented [AI11]: This sentence is a bit awkward. Consider this rewrite:
Relational data models best function with cross-table relationships.

Commented [AI12]: This definition is a bit tricky to track with. If I researched correctly, denormalization is just the improvement of the ability to have the information in the database read. Its main goal is to improve the performance. Perhaps a rewrite could look like this:
Oftentimes in order to increase the performance of databases, denormalization is implemented. Denormalization is intentional duplication and groupings of data, which add write complexity, to enable all information to be accessed in a single row…

If denormalization isn't possible (multi-shard queries are necessary,) then the service needs to consider the tradeoffs between consistency and availability. You can read more about this tradeoff and the CAP theorem in our article on databases.

1.3 When to Uuse Ssharding

The benefits of a sharded architecture, versus other kinds of database partitioning, are:

- Leveraging average hardware instead of high-end machines
- Quickly scaling by adding morewith additional shards
- Better performance because each machine is under less loaddue to underloaded machines

Sharding is particularly useful when a single database server:

- Can't hold all the data
- Can't compute all the query responses fast enough
- Can't handle the number of concurrent connections

You might also need sharding when you need to maintain distinct geographic regions, even if the above compute constraints *haven't* been hit. Either your service will be faster when the data servers are physically closer to the users, or there's legislation about data location and usage in one of the countries your service operates in.

1.4 When Nnot to Uuse Ssharding

The disadvantages of database sharding are all about complexity. The queries become more complex because they have to somehow get the correct shard key, and need to be aware of avoiding multi-shard queries.

If the shards can't be entirely isolated, you need to implement eventual consistency for duplicated data or upholding relational constraints. The implementation and deployment of the your database get a lot more complex, as do failovers, backups, and other kinds of maintenance. Essentially - you should only use database sharding when you absolutely necessary.have to.

Commented [AI13]: This rewrite is a suggestion purely for style matching. Each of these bullet points read similarly in style (except the last one) so I suggest this adjustment to reestablish the flow of voice.

Commented [AI14]: To keep with the tone of the piece, it is better to leave this sentence as a statement rather than phrase it as a suggestion.

2. Approaches to Sharding

The different approaches to sharded architectures are based on how the shard key is assigned. Regardless of what they're derived from, shard keys need to be unique *across* shards, so their values need to be coordinated. This leads to a tradeoff between a centralized "name server" that can dynamically optimize logical shards for performance~~,~~ and a predetermined distributed algorithm that is faster to compute.

Shard keys are derived from some invariant feature of the data that utilizes business logic to optimize for the most common queries. Common choices are tenant ID~~id~~s, location, and timestamps. Custom configuration can also tune the performance of a sharded architecture based on usage patterns, actual shard sizes, etc.

Here are the three most common approaches to sharding you should know ~~about~~ when designing systems.

2.1 Range

Data can be assigned to a shard based on what ~~"~~range~~"~~ it falls into. For example, a database with sequential time-based data, like log history, could shard based on month ranges. One big advantage of range-based shard keys is they make sequential access patterns ~~very fast~~quickly~~,~~ because data that is ~~"~~close~~"~~ in the given range will be on the same shard.

Unpredictability with the balance of data is one major downside to ranges.~~One downside to ranges is the balance of data can be unpredictable.~~ For example, an e-commerce company might have ~~a lot~~ more orders in December compared to other months because of holiday shopping, so the shard with the December range could get overwhelmed while the other shards aren't doing much.

2.2 Hash

To address the issue of imbalanced shards, data can be distributed based on a hash of part of the data. An effective hash function will randomize and distribute incoming data to prevent any access patterns that could overwhelm a node. For example, the profile pages of celebrities get

Commented [AI15]: When using a term familiar in the field, no quotations are needed 😊

Commented [AI16]: Although I know what this means, it reads awkwardly. Here's a suggested rewrite:
To address the issue of imbalanced shared, data hashing can be introduced to represent all of the information.

substantially more traffic than the average user, so a hash function can be used to randomly distribute these celebrity users and prevent hotspots.

One major advantage of a hash-based sharded architecture is that the shard key can be computed by any server that knows the hash function.~~,~~ That way~~so~~ there's no centralized point of failure.

One big downside of hashing is that adding shards can require a lot of overhead~~,~~ depending on the implementation. **Consistent hashing** limits this by guaranteeing a minimum amount of data will have to be moved when a new node is added.

2.3 Name Node

The last approach to sharding is any implementation that uses a central "~~n~~name node" to coordinate the mapping of data to shard keys. What's nice about this approach is it makes the business logic very clear and easy to update based on usage patterns.

For example, an important client might want to guarantee a precise location distribution of their data on your service. You can base the shard key on ~~tennant~~tenant ID~~it~~ and have a clear mapping in the name node that assigns their tenant to shards in their desired geographic regions. This~~and~~ is also easy to update if their needs change.

The main downside is a N~~n~~ameN~~ n~~ode is a central point of failure, and additional care is needed to implement good replication and failovers for that node. This approach also adds an additional step of consulting the N~~n~~ameN~~ n~~ode in the lookup process, which can slow down database operations.

3. Example S~~s~~harding Q~~q~~uestions

The questions asked in system design interviews tend to begin with a broad problem or goal, so it's unlikely that you'll get an interview question entirely about sharding.

Commented [AI17]: I'm not entirely sure what this entire paragraph is getting at. Are you saying that literally any implementation to the database using a name node coordinate the mapping of the data? If that's not what you're saying, then we need to revisit this paragraph and figure out a more concise way of saying what needs to be said. 😉

Commented [AI18]: So I couldn't find anything on tenant it but I did read up on tenant ID so I assumed this is what was meant. If it wasn't, please feel free to re-fix this section.

Commented [AI19]: Please correct me if I'm wrong so I don't miss this in the future—all the research I conducted has "name nodes" as "NameNodes". Is this the technical and correct way to use this term?

However, you may be asked to solve a problem where sharding will be an important part of the solution. As a result, what you really need to know is *when*~~WHEN~~ you should bring it up and how you should approach it.

To help you with this, we've compiled ~~the below~~a list of sample system design questions that incorporates sharding.~~. Sharding is relevant for all of the below questions.~~

- Design Dropbox (Read the answer)
- Design Pastebin (Read the answer)
- Design a web crawler (Read the answer)
- Design Flickr (Read about Flickr's actual architecture)

4. System D~~d~~esign I~~i~~nterview P~~p~~reparation

Sharding is just one piece of system design. ~~And t~~To prepare for success ~~succeed on system design interviews,~~ you'll need to familiarize yourself with a few other concepts and practice how to best ~~you~~ communicate your answers.

It's best to take a systematic approach to make the most of your preperations~~practice time~~, and we recommend the following steps:

4.1 Learn the C~~c~~oncepts

There is a base level of knowledge required to ~~be able to~~ speak intelligently about system design. We've taken that pressure off you by publishing a ~~To help you get this foundational knowledge (or to refresh your memory), we've published a~~ full series of articles (just like this one)~~,~~that ~~which~~ cover the primary concepts ~~that~~ you'll need to know:

- Network protocols and proxies
- Databases
- Latency, throughput, and availability

- Load balancing
- Leader election
- Caching
- Sharding
- Polling, SSE, and WebSockets
- Queues and pub-sub

We~~'d~~ encourage you to begin ~~your~~ preparation by reviewing the above concepts and by studying our system design interview guide, which covers a step-by-step method for answering system design questions. Once you're familiar with the basics, ~~you should~~ begin practicing with example questions.

4.2 Practice by Y~~y~~ourself or with P~~p~~eers

Next, ~~you'll want to~~ get some practice with system design questions. You can start with the examples listed above~~,~~ or with the example questions in our system design guide.

We~~'d~~ recommend ~~that~~ you start by interviewing yourself aloud~~out loud~~. ~~You should p~~Play both the role of the interviewer and the candidate, asking and answering questions. This will help you develop ~~your~~ communication skills and perfect your process for breaking down questions.

We ~~would~~ also strongly recommend that you practice solving system design questions with a peer interviewing you. A great place to start is to practice with friends or family ~~if you can~~. If you don't have anyone in your network who can interview you, then you might want to check out our ~~our~~ system design mock interview peer group.

4.3 Practice with E~~e~~x-interviewers

Practicing with peers can be a great help~~,~~ and it's usually free. But, at some point, you'll start noticing that the feedback you are getting from peers has its limitations. ~~isn't helping you that much anymore.~~ Once you reach that stage, we recommend practicing with ex-interviewers from top tech companies.

Commented [AI20]: As mentioned briefly at the beginning, you'll want to match these link titles to the actual titles of the articles you link to. It avoids reader confusion and also reaffirms the given information.

Commented [AI21]: Is this the actual title of the guide? If it is, then it needs to be System Design Guide (proper nouns).

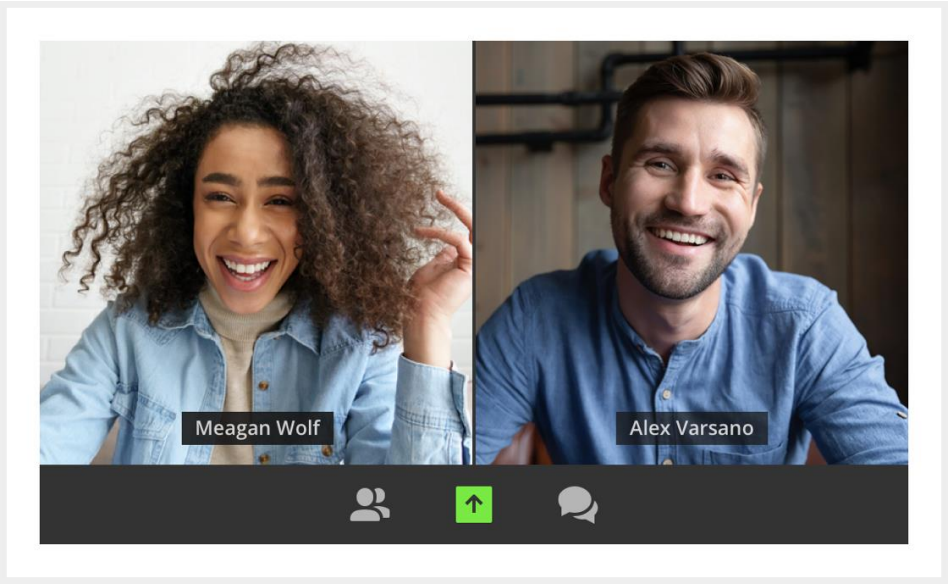
Commented [AI22]: Instead consider:
Another great tool to use is our system design mock interview peer group.

BUT THEN GIVE ME AT LEAST ONE SENTENCE INFORMATING ME OF WHAT THAT IS!

If you know someone who has experience running interviews at Facebook, Google, or another big tech company, then that's fantastic. But for most of us, it's tough to find the right connections to make this happen. And it might also be difficult to practice multiple hours with that person unless you know them ~~really~~ well.

Commented [AI23]: This paragraph feels a bit redundant. Perhaps cut it down to a single sentence and combine it directly with the paragraph following it.

Here's the good news. We've already made the connections for you. We've created a coaching service where you can practice system design interviews 1-on-1 with ex-interviewers from leading tech companies. Learn more and start scheduling sessions today.



BROWSE FAANG EX-INTERVIEWERS

Learn Mmore Aabout Ssystem Ddesign Iinterviews

This is just one of 9 concept guides that we've published about system design interviews. Check out all of our system design articles on our Tech blog.

- ShareShare on Facebook
- TweetTweet on Twitter
-