SYSTEMS DESIGN CONCEPTS PRACTICE TEST (SAMPLE)

STUDY GUIDE

SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://systemsdesignconcepts.examzify.com

Table of Contents

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

Questions

1. Which status code indicates that authentication is required?

A. 200 OK

B. 201 Created

C. 400 Bad Request

D. 401 Unauthorized

2. Why might gRPC outperform REST in some benchmarks?

A. Protocol Buffers produce roughly 10x throughput due to binary encoding and smaller payloads.

B. REST is always faster than gRPC in all scenarios.

C. gRPC uses JSON by default.

D. gRPC bypasses HTTP entirely.

3. What is the primary function of load balancer health checks?

A. To measure health of client devices

B. To configure DNS for failover

C. To monitor SSL certificate validity

D. To remove unhealthy servers from rotation automatically.

4. According to cache capacity guidelines, when should you scale the cache?

A. Scale cache only when memory usage exceeds 50%.

B. Scale cache only when ops/sec exceed 10k.

C. Scale cache only when latency exceeds 10 ms.

D. Scale when memory usage exceeds 80% or ops/sec exceed 100k, or latency falls below 0.5 ms.

5. Which sharding approach stores shard assignments in a lookup table and uses a directory before querying data, offering flexibility but adding latency and a potential single point of failure?

A. Range-Based Sharding assigns contiguous ranges of shard key values to specific shards.

B. Directory-Based Sharding stores shard assignments in a lookup table, but introduces latency and a potential single point of failure.

C. Consistent Hashing minimizes data movement on shard changes.

D. Two-Phase Commit guarantees cross-shard consistency but is slow.

6. Which pagination method breaks large result sets into pages by skipping a set number of records and can produce duplicates or skipped records if data changes during pagination?

A. Cursor-based pagination

B. Offset-based pagination

C. Page-based pagination

D. Keyset pagination

7. What is the primary purpose of an Inverted Index?

A. It maps words to documents containing it for full-text search.

B. It maps values to rows for fast equality lookups.

C. It is the default index type for most databases.

D. It is used for multi-dimensional proximity queries.

8. What is the primary cost associated with not reusing HTTP connections across requests?

A. Without connection reuse, you avoid DNS lookups.

B. Without connection reuse, a full TCP handshake is needed for each request, adding latency.

C. Without connection reuse, requests are queued.

D. Without connection reuse, you can never use TLS.

9. Which algorithm is recommended for SSE or WebSocket workloads?

A. Round Robin

B. Hash-based

C. Least Connections

D. Weighted Round Robin.

10. Which statement about managed sharding is correct?

A. DynamoDB hashes the partition key to internal partitions.

B. Cassandra uses modulo hashing with no virtual nodes.

C. MongoDB uses fixed shards with manual balancing.

D. All three databases require manual shard configuration.

Answers

1. D
2. C
3. D
4. D
5. B
6. B
7. A
8. B
9. C
10. A

Explanations

1. Which status code indicates that authentication is required?

A. 200 OK

B. 201 Created

C. 400 Bad Request

D. 401 Unauthorized

Authentication required in HTTP is signaled by the 401 Unauthorized status. This means the request lacks valid authentication credentials for the target resource, or they were not provided at all. The server uses 401 to prompt the client to authenticate, often with a WWW-Authenticate header describing the required method. Once the client supplies valid credentials, the server may respond with a success code or another appropriate status depending on the outcome. In contrast, 200 OK means the request succeeded and the response contains the requested data. 201 Created indicates a new resource was successfully created as a result of the request. 400 Bad Request means the server could not understand the request due to malformed syntax or invalid parameters, not an authentication issue.

2. Why might gRPC outperform REST in some benchmarks?

A. Protocol Buffers produce roughly 10x throughput due to binary encoding and smaller payloads.

B. REST is always faster than gRPC in all scenarios.

C. gRPC uses JSON by default.

D. gRPC bypasses HTTP entirely.

The main factor is how data is encoded and transported. gRPC typically uses Protocol Buffers, a compact binary format, which keeps message sizes small and speeds up parsing compared to JSON. Smaller payloads mean less serialization work and lower network bandwidth, which can lead to higher throughput in benchmarks. In addition, gRPC runs over HTTP/2, offering features like multiplexed streams and header compression, which further improves efficiency and latency. The statement that gRPC uses JSON by default is a misconception; gRPC uses Protocol Buffers by default, while REST commonly uses JSON.

3. What is the primary function of load balancer health checks?

A. To measure health of client devices

B. To configure DNS for failover

C. To monitor SSL certificate validity

D. To remove unhealthy servers from rotation automatically.

Health checks verify that each backend server can actually handle requests. The load balancer periodically probes the servers, and if a server fails the probes consistently, it is taken out of the traffic rotation so new requests aren't sent to it. This automatic removal of unhealthy servers keeps the service available by avoiding failed nodes, and once a server recovers, it can rejoin the rotation. The other options aren't the primary role of health checks. Checking client device health isn't something the load balancer does. DNS failover is a separate mechanism for directing traffic at a higher level, not about monitoring individual backends. SSL certificate validity isn't the core function of health checks, though some systems can monitor cert expiry, it's not what the health check primarily accomplishes.

4. According to cache capacity guidelines, when should you scale the cache?

A. Scale cache only when memory usage exceeds 50%.

B. Scale cache only when ops/sec exceed 10k.

C. Scale cache only when latency exceeds 10 ms.

D. Scale when memory usage exceeds 80% or ops/sec exceed 100k, or latency falls below 0.5 ms.

The idea is to manage cache capacity using multiple, practical signals rather than a single metric. Scale when you're genuinely approaching the limits or when demand is high, so you keep performance from degrading. If memory usage climbs above 80%, you're near the memory cap and eviction pressure rises, so adding capacity helps. If the workload pushes throughput beyond a high threshold (like 100k ops/sec), the cache is under heavier use and may start showing latency or miss penalties unless you scale. If latency is already very good (below 0.5 ms), you have headroom to scale ahead of expected growth to preserve that fast response as traffic increases. Using these combined conditions helps ensure scaling happens in response to real pressure and also in anticipation of rising load, rather than based on a single metric that might mislead.

5. Which sharding approach stores shard assignments in a lookup table and uses a directory before querying data, offering flexibility but adding latency and a potential single point of failure?

A. Range-Based Sharding assigns contiguous ranges of shard key values to specific shards.

B. Directory-Based Sharding stores shard assignments in a lookup table, but introduces latency and a potential single point of failure.

C. Consistent Hashing minimizes data movement on shard changes.

D. Two-Phase Commit guarantees cross-shard consistency but is slow.

Sharding with a directory means you keep a central lookup structure that maps each shard key to the specific shard, and every query first consults that directory to route to the right shard. This setup offers flexibility because you can reassign data or rebalance shards by updating the directory without changing application logic. The trade-offs are clear: you pay extra latency for the directory lookup on every query, and there's a potential single point of failure if the directory becomes unavailable or bottlenecks the system. Understand how the others differ: range-based sharding assigns contiguous key ranges to shards directly, which is simple and fast for routing but harder to adapt when you need to rebalance or add shards. Consistent hashing uses a hashing ring so keys map to shards with minimal movement when the topology changes, avoiding a central directory and thus reducing rebalancing work. Two-phase commit is about coordinating transactions across multiple shards and can add latency due to coordination overhead, but it doesn't define how shard assignments are stored or routed.

6. Which pagination method breaks large result sets into pages by skipping a set number of records and can produce duplicates or skipped records if data changes during pagination?

A. Cursor-based pagination

B. Offset-based pagination

C. Page-based pagination

D. Keyset pagination

Offset-based pagination uses a fixed offset to skip a set number of records, then returns the next group as the page. This approach is simple to implement—you request, for example, 10 items starting at offset 0, then 10 items starting at offset 10, and so on. The reason it can produce duplicates or skipped records when data changes is that the offset relies on a stable ordering of the underlying data. If new rows are inserted or existing ones are deleted between requests, the relative positions of items shift. A simple illustration: start with A, B, C, D, E and a page size of 2. The first page returns A and B. If a new item X is inserted between A and B, the order becomes A, X, B, C, D, E. The next page using an offset of 2 would return B and C, so B—seen on the first page—appears again on the second page, creating a duplicate. Conversely, some items that were on the second page before might be skipped entirely due to the shift. Because of this behavior, offset-based pagination isn't ideal for dynamic data. Other methods—such as cursor-based or keyset pagination—use a pointer to the last seen value (like a unique ID) to fetch the next page, which remains stable even if new rows are added or removed in between requests.

7. What is the primary purpose of an Inverted Index?

A. It maps words to documents containing it for full-text search.

B. It maps values to rows for fast equality lookups.

C. It is the default index type for most databases.

D. It is used for multi-dimensional proximity queries.

An inverted index is a data structure that supports fast full-text search by mapping each word to the list of documents in which it appears. This lets you quickly locate relevant documents without scanning every document—retrieve the documents for each search term and combine them (for example, by intersecting lists for multi-term queries, or using positions for phrase searches). It's the core mechanism behind search engines and document stores that need rapid text lookup. The other options describe different kinds of indexing or uses (general value lookups, a universal default index, or spatial/proximity indexing), which aren't the primary purpose of an inverted index.

8. What is the primary cost associated with not reusing HTTP connections across requests?

A. Without connection reuse, you avoid DNS lookups.

B. Without connection reuse, a full TCP handshake is needed for each request, adding latency.

C. Without connection reuse, requests are queued.

D. Without connection reuse, you can never use TLS.

When you don't reuse HTTP connections, you pay the cost of establishing a new transport channel for each request. The main expense is the TCP handshake that starts a new connection: a three way exchange where the client and server agree to talk, which requires at least one round-trip time (RTT) before any HTTP data can be sent. That handshake adds latency for every request, and it also uses extra resources on both sides. If you reuse connections (keep-alive or HTTP/2 multiplexing), that handshake is largely paid once and reused for subsequent requests, so the per-request latency drops dramatically. DNS lookups and TLS-related steps can contribute to startup cost as well, but they're not the primary driver in this scenario, and TLS can still occur over new connections via session resumption, so the idea that you can't use TLS isn't correct.

9. Which algorithm is recommended for SSE or WebSocket workloads?

A. Round Robin

B. Hash-based

C. Least Connections

D. Weighted Round Robin.

When dealing with SSE or WebSocket, connections stay open for long periods, so the load a backend bears depends on how many active connections it currently handles, not just how many requests it has seen. A load balancer that uses the Least Connections strategy routes each new connection to the backend with the fewest active connections, keeping the overall load balanced as connections persist and new ones arrive. This dynamic awareness is crucial for long-lived connections, preventing a server that already has many active ties from becoming a bottleneck. Other approaches have drawbacks in this scenario. Round Robin distributes connections evenly without regard to real-time load, which can lead to some servers becoming overloaded while others are underutilized. Hash-based routing fixes a client to a specific backend, causing potential imbalance if traffic patterns are skewed. Weighted Round Robin uses fixed weights and doesn't adapt to current server load, so it can miss sudden changes in how many long-lived connections each server is actually handling. Least Connections directly targets the real-time burden on each server, making it the best fit for SSE and WebSocket workloads.

10. Which statement about managed sharding is correct?

A. DynamoDB hashes the partition key to internal partitions.

B. Cassandra uses modulo hashing with no virtual nodes.

C. MongoDB uses fixed shards with manual balancing.

D. All three databases require manual shard configuration.

Managed sharding means the system handles how data is split and moved across storage nodes, so you don't have to configure partitions yourself. In DynamoDB, the partition key is hashed to determine which internal partitions store the item. This hashing and distribution are automatic, so sharding is fully managed by the service. Cassandra, on the other hand, uses a token-based ring with a partitioner (often Murmur3) and typically employs virtual nodes to spread load; this is not simply modulo hashing with no virtual nodes, so that description isn't accurate for Cassandra. MongoDB's sharding involves a cluster with shards and chunks, with an automatic balancer that moves chunks to keep data and load balanced. It isn't about fixed shards with manual balancing. Because DynamoDB handles the sharding internally by hashing the partition key, the statement about it being hashed to internal partitions reflects managed sharding.

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we'd love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://systemsdesignconcepts.examzify.com

We wish you the very best on your exam journey. You've got this!