Question 15 of 15

What should be your primary focus when designing schemas in system design interviews?

1 Optimizing for the fastest possible queries

2 Creating the most normalized schema possible

3 Demonstrating knowledge of every database constraint type

[✓ 4 Showing you can design a reasonable schema that supports requirements]

Correct!

Your goal is to show that you can design a reasonable schema that supports your system's requirements, then move on. The bar is much lower than in dedicated data modeling interviews - you need something clear, functional, and aligned with requirements.

Question 1 of 15

Which database type should you default to in system design interviews?

1 Document databases

2 Key-value stores

3 Graph databases

4 Relational databases (SQL) ✓

Correct!

Relational databases (SQL) should be your default choice unless your requirements clearly signal a specialized model. PostgreSQL is recommended as it's well-understood and works for most use cases.

Question 2 of 15

Graph databases are commonly used and recommended for social network features in system design interviews.

1 True

[✓ 2 False]

Correct!

Graph databases are a common mistake in interviews. Even companies like Facebook, LinkedIn, and Twitter use SQL for their core relationship data. Graph databases add unnecessary complexity without clear benefits for most use cases.

Question 3 of 15

What are the three key factors that drive schema design decisions?

1 Performance, security, and maintainability

2 Speed, reliability, and flexibility

3 Cost, complexity, and scalability

[✓ 4 Data volume, access patterns, and consistency requirements]

Correct!

Data volume determines where data can physically live, access patterns drive most design decisions, and consistency requirements determine how tightly coupled data can be. These three factors guide all schema design choices.

Question 4 of 15

You should use business data like email addresses as primary keys instead of system-generated IDs.

1 True

[✓ 2 False]

Correct!

Use system-generated IDs like user_id or post_id rather than business data like email addresses. System-generated keys stay stable even when business rules change.

Question 5 of 15

When should you consider document databases over SQL?

1 Never, SQL is always better

[✓ 2 When schema changes frequently or you have deeply nested data]

3 Always, for better performance

4 When you need ACID guarantees

Correct!

Consider document databases when your schema changes frequently, when you have deeply nested data that would require many joins in SQL, or when different records have vastly different structures.

Question 6 of 15

Key-value stores are typically used as a replacement for SQL databases rather than alongside them.

1 True

[✓ 2 False]

Correct!

In practice, you'll often use both together. SQL as your source of truth with a key-value cache (like Redis) in front for hot data. This gives you fast access without sacrificing durability or complex queries.

Question 7 of 15

What type of relationship exists between users and posts in a social media application?

1 No relationship

2 Many-to-many (N:M)

[✓ 3 One-to-many (1:N)]

4 One-to-one (1:1)

Correct!

A user can have many posts, but each post belongs to only one user. This is a classic one-to-many relationship where the 'many' side (posts) contains a foreign key to the 'one' side (users).

Normalization means storing each piece of information in exactly one place.

[✓ 1 True] ✓

2 False

✓ **Correct!**

Normalization prevents data anomalies by ensuring each piece of information exists in only one location. This prevents inconsistencies where updates happen in one place but not another.

Question 9 of 15

When should you denormalize your schema in system design interviews?

1 Only for NoSQL databases

2 Always, for better performance

[✓ 3 Start normalized and denormalize only when needed]

4 Never, normalization is always better

Correct!

Start with a clean normalized model and denormalize only when needed. Denormalization creates consistency problems that are harder to solve than the performance problems you're trying to avoid.

Question 10 of 15

Wide-column databases are optimized for massive write-heavy workloads and time-series data.

[✓ 1 True]

2 False

Correct!

Wide-column databases like Cassandra organize data into column families where rows can have different sets of columns, making them ideal for enormous write volumes, time-series data, and analytics workloads.

Question 11 of 15

What is the primary purpose of database indexes?

1 To enforce data constraints

2 To store data more efficiently

3 To handle database replication

[✓ 4 To help the database find records quickly without scanning every row]

Correct!

Indexes are data structures that help the database find records quickly without scanning every row, like an index in a book that lets you jump directly to the relevant page.

Question 12 of 15

Foreign keys help ensure referential integrity but come at a cost because the database has to validate each insert/update.

1 [✓ True]

2 False

Correct!

Foreign keys prevent orphaned records and ensure data integrity, but they add overhead because the database must validate relationships on every write operation. At very large scale, some companies drop them for performance.

Question 13 of 15

How should you choose your shard key for database partitioning?

1 Use timestamp-based sharding

2 Always use user_id

[✓ 3 Shard by the primary access pattern]

4 Choose randomly to distribute load evenly

Correct!

Shard by the primary access pattern. If you mostly query 'posts by user,' shard by user_id. This keeps related data together and avoids expensive cross-shard queries.

Question 14 of 15

Cross-shard queries are expensive and should be avoided whenever possible.

[✓ 1 True]

2 False

Correct!

Cross-shard queries require querying multiple databases and merging results, which is expensive and complex. Good shard key selection keeps related data together to minimize these queries.