Interview Questions Answers - Advanced Database topics

---

1. What are the key differences between SQL and NoSQL databases?

Feature	SQL (Relational)	NoSQL (Non-Relational)
Schema	Fixed, predefined	Flexible, dynamic
Structure	Tables with rows and columns	Documents, key-value pairs, etc.
Query Language	SQL	Varies (e.g., JSON queries, APIs)
Data Integrity	Strong ACID compliance	Often BASE, eventual consistency
Scalability	Vertical scaling (hard to shard)	Horizontal scaling (built-in)
Best Use Cases	Banking, ERP, inventory	IoT, analytics, real-time apps

---

2. Explain ACID vs. BASE.

ACID (SQL Databases):

- **Atomicity** – All operations in a transaction complete, or none do.
- **Consistency** – Data stays valid and follows rules.
- **Isolation** – Simultaneous transactions don't interfere.
- **Durability** – Committed data survives crashes.

BASE (NoSQL Databases):

- **Basically Available** – System always responds.
- **Soft state** – Data may not be immediately consistent.
- **Eventually consistent** – Data consistency achieved over time.

---

3. What are the different types of NoSQL databases, and when would you use each?

- **Document (e.g., MongoDB):**
  - JSON-like documents
  - Use for content management, user profiles
- **Key-Value (e.g., Redis, DynamoDB):**
  - Keyed access, extremely fast
  - Use for caching, session storage, config stores
- **Columnar (e.g., Cassandra, HBase):**
  - Stores by columns
  - Use for time-series data, analytics, logs
- **Graph (e.g., Neo4j):**
  - Nodes and relationships
  - Use for social graphs, fraud detection, recommendation systems

---

4. When would you prefer MongoDB over PostgreSQL?

You'd prefer **MongoDB** when:

- The data structure is flexible or evolving (e.g., user profiles).
- You're storing nested JSON documents.
- You need fast development cycles and dynamic schemas.
- You're okay with eventual consistency or tuning consistency per use case.

---

5. What are the trade-offs in the CAP theorem?

The **CAP Theorem** states you can only fully achieve two of three guarantees in a distributed system:

- **Consistency** – Latest, correct data on every read

- **Availability** – System always responds
- **Partition Tolerance** – System tolerates network splits

Trade-offs:

- **CP**: No availability during partitions (e.g., HBase)
- **AP**: Allows stale reads to stay available (e.g., DynamoDB)
- **CA**: Only achievable if partitions never happen (rare in distributed systems)

---

6. Where do SQL and NoSQL databases fit within the CAP theorem categories?

- **SQL (e.g., PostgreSQL, MySQL):**
  Typically **CP** — prioritize consistency and partition tolerance (if distributed).
- **NoSQL:**
  - **DynamoDB, Couchbase**: **AP** – prioritize availability and partition tolerance.
  - **MongoDB**: Tunable between CP and AP based on settings.
  - **Cassandra**: Often leans AP, but tunable consistency.
  - **Neo4j**: CP (graph integrity is vital).

---

7. What database model would you choose for:

- **a. A financial ledger system:**
  SQL (e.g., PostgreSQL, MySQL)
  Why: Requires strong consistency, transactional integrity (ACID compliance).
- **b. A product catalog:**
  NoSQL Document DB (e.g., MongoDB)
  Why: Schema flexibility, frequent updates, nested product attributes.
- **c. A real-time chat app:**
  NoSQL Key-Value or Document DB (e.g., Redis or DynamoDB)
  Why: Low latency, high throughput, can handle eventual consistency.

---

8. What are the limitations of relational databases in modern distributed systems?

- Poor horizontal scalability (harder to shard).
- Rigid schemas – not ideal for evolving data structures.
- Joins across distributed nodes can be expensive.
- Less performant for high-velocity, high-volume workloads.
- Can become a single point of failure if not clustered.

---

9. What is polyglot persistence and why is it useful?

Polyglot persistence is the use of multiple types of databases in a single system based on the specific needs of each component.

Why it's useful:

- You can optimize for performance, scalability, and flexibility.
- Use SQL for transactional data and NoSQL for logs or caching.
- Helps decouple systems and use the best tool for each job.

---

10. How does data modeling differ between SQL and NoSQL systems?

- **SQL**:
  - Highly normalized (multiple tables with relationships)
  - Predefined schemas
  - Focus on reducing data duplication
- **NoSQL**:
  - Denormalized or nested structures
  - Schema-less or dynamic schema

- Data often modeled for **access patterns** (read/write efficiency)