Database Deep Dive Interview Q&As

🔹 Section 1: Database Fundamentals

Q1. Difference between SQL vs NoSQL databases?
Answer: SQL databases are relational, schema-based, and use tables (e.g., MySQL, PostgreSQL). NoSQL databases are non-relational, schema-flexible, and use models like key-value, document, or graph (e.g., MongoDB, Cassandra). SQL is best for structured data; NoSQL suits unstructured, high-scale systems.
💡 Tip: Always say — "SQL for transactions (banking), NoSQL for scalability (social media)."

Q2. When to use relational vs non-relational databases?
Answer: Relational DBs fit structured data, transactions, and consistency (e.g., orders, inventory). Non-relational DBs are great for flexible schemas and massive data (e.g., IoT, logs, feeds). Choosing depends on data relationships and scalability needs.
💡 Tip: Best example — "Relational = banking, Non-relational = Netflix recommendations."

Q3. What is normalization vs denormalization?
Answer: Normalization removes redundancy by splitting into smaller tables and using relationships. Denormalization combines data into fewer tables for faster reads at the cost of duplication. Both are trade-offs between performance and consistency.
💡 Tip: Interviewers love — "Normalization = integrity, Denormalization = speed."

Q4. What is database indexing?
Answer: An index is a data structure (like a book's index) that makes searches faster. Instead of scanning all rows, DBs jump directly to indexed records. However, too many indexes can slow down

inserts/updates.
💡 Tip: Say — “Indexes speed up reads but slow down writes.”

Q5. What is a foreign key?
Answer: A foreign key links one table to another by referencing its primary key. It enforces referential integrity so invalid relationships cannot exist (e.g., no order without a valid user).
💡 Tip: Give example — “Orders table → UserID foreign key → Users table.”

🔷 Section 2: Transactions & Concurrency

Q6. What is ACID in databases?
Answer: ACID = Atomicity, Consistency, Isolation, Durability. It ensures transactions are reliable — either all steps succeed or none (Atomicity), data is valid (Consistency), no interference (Isolation), and results persist (Durability).
💡 Tip: Banking transfer is the simplest way to explain ACID.

Q7. What is a database transaction?
Answer: A transaction is a group of operations treated as one unit. Either all succeed (commit) or none (rollback). Transactions ensure consistency in critical systems like banking and payments.
💡 Tip: Always say — “Transactions = all-or-nothing.”

Q8. Optimistic vs pessimistic locking?
Answer: Optimistic locking assumes low conflicts; it checks at commit if data changed. Pessimistic locking prevents conflicts by locking rows until a transaction finishes. Optimistic is better for read-heavy, pessimistic for write-heavy systems.
💡 Tip: Remember — “Optimistic = fast, Pessimistic = safe.”

Q9. What is isolation level (Read Uncommitted → Serializable)?
Answer: Isolation levels control transaction visibility. Read

Uncommitted = dirty reads allowed; Read Committed = no dirty reads; Repeatable Read = stable reads; Serializable = full isolation but slowest.
💡 Tip: Say — "Serializable is safest but reduces performance."

Q10. Deadlock in databases — what is it?
Answer: Deadlock occurs when two or more transactions wait on each other's locks and cannot proceed. DBs resolve it by aborting one transaction. It's common in high-concurrency systems.
💡 Tip: Prevent by consistent lock ordering & timeout policies.

🔷 Section 3: Indexing & Optimization

Q11. What are clustered vs non-clustered indexes?
Answer: Clustered index sorts and stores rows physically in order (only one per table). Non-clustered index creates a separate lookup structure that points to data. Clustered = faster retrieval for range queries.
💡 Tip: "Primary key is usually clustered by default."

Q12. What is a composite index?
Answer: A composite index covers multiple columns to optimize multi-column queries. Useful when queries filter or sort on more than one column. But order of columns in the index matters.
💡 Tip: Say — "Put most selective column first."

Q13. What is a covering index?
Answer: A covering index contains all the columns needed to answer a query, so DB doesn't need to touch the table. It improves performance but uses more storage.
💡 Tip: Mention — "Helps in query optimization for SELECT heavy apps."

Q14. What is query execution plan?
Answer: Execution plan shows how DB executes a query (scans, joins, indexes). Developers use it to optimize slow queries. Tools like EXPLAIN (MySQL, Postgres) show the plan.
💡 Tip: Always say — “Check execution plan before adding random indexes.”

Q15. Difference between indexing in SQL vs NoSQL?
Answer: SQL DBs support B-Tree, hash, and bitmap indexes. NoSQL DBs like MongoDB support indexes on fields, compound indexes, and text search indexes. Indexing strategies vary with data models.
💡 Tip: Show awareness of “NoSQL also supports indexing.”

🔷 Section 4: Replication & Sharding

Q16. What is replication in databases?
Answer: Replication copies data across multiple servers to improve availability and fault tolerance. Common setups: master-slave, multi-master. Helps scale reads and ensure redundancy.
💡 Tip: Say — “Replication improves read performance & reliability.”

Q17. What is database sharding?
Answer: Sharding splits a large database into smaller parts (shards), each on different servers. It distributes load and allows horizontal scaling for big datasets.
💡 Tip: Example — “Shard users A–M on one DB, N–Z on another.”

Q18. Leader-follower vs leaderless replication?
Answer: Leader-follower: one primary accepts writes, others replicate (faster consistency). Leaderless: all nodes accept writes, but conflicts need resolution (better fault tolerance).
💡 Tip: Say — “Leader-follower = simpler, Leaderless = more resilient.”

Q19. Difference between OLTP vs OLAP DBs?
Answer: OLTP handles real-time transactions (insert/update) — e.g., banking. OLAP handles analytical queries on large data sets — e.g., BI dashboards. OLTP = speed, OLAP = insights.
💡 Tip: Interviewers expect — “OLTP = day-to-day, OLAP = analysis.”

Q20. What is CAP theorem?
Answer: In distributed systems, only two can be fully guaranteed: Consistency, Availability, Partition Tolerance. SQL prefers Consistency + Availability, NoSQL often sacrifices consistency for scalability.
💡 Tip: Always give Amazon DynamoDB as an AP example.

🔹 Section 5: Advanced Database Concepts

Q21. What is database partitioning?
Answer: Partitioning splits a table into smaller pieces (ranges, hashes) to improve performance and manageability. Unlike sharding, it’s usually within one DB server.
💡 Tip: Say — “Partitioning improves performance for huge tables.”

Q22. What is a materialized view?
Answer: A materialized view stores the result of a query physically for faster reads. Unlike normal views, it doesn’t calculate results on the fly but must be refreshed.
💡 Tip: Great for analytics dashboards.

Q23. What is database federation?
Answer: Federation combines multiple databases into one virtual DB, letting queries span across them. Useful for integrating heterogeneous systems.
💡 Tip: Example — BI tools querying multiple sources.

Q24. What is polyglot persistence?
Answer: Polyglot persistence means using different types of

databases for different needs (SQL + NoSQL + Graph). E.g., MongoDB for documents, PostgreSQL for transactions, Neo4j for graphs.
💡 Tip: “Right tool for the right job” is the key phrase.

Q25. What is database shadowing?
Answer: Shadowing means maintaining a full copy (shadow) of the database for disaster recovery. It ensures backup is immediately available.
💡 Tip: Mention — “Used in high-reliability systems.”

🔹 Section 6: Best Practices

Q26. How do you handle database migrations safely?
Answer: Use migration tools (Flyway, Liquibase), apply incremental changes, and test before production. Avoid blocking writes during schema updates.
💡 Tip: “Backward compatibility is key in safe migrations.”

Q27. How to prevent SQL Injection?
Answer: Use prepared statements, ORM frameworks, and input validation. Never concatenate raw input into queries.
💡 Tip: Always say — “SQL injection is OWASP Top 10 vulnerability.”

Q28. How do you scale databases?
Answer: Scale reads with replication, scale writes with sharding, use caching (Redis), and optimize queries. Vertical scaling has limits, so horizontal scaling is preferred.
💡 Tip: Use Amazon, Netflix, or Uber examples.

Q29. What is database monitoring?
Answer: Monitoring tracks performance metrics (queries/sec, latency, deadlocks). Tools: Prometheus, Grafana, New Relic. Helps prevent failures and optimize queries.
💡 Tip: Always connect monitoring with reliability.

Q30. What are common database bottlenecks?
Answer: Common issues include slow queries, too many joins, missing indexes, I/O latency, network overhead, and unoptimized schema. They reduce scalability and speed.
💡 Tip: Say — "Profiling + query optimization fixes most bottlenecks."