

Distributed Transactions - Discussion Topics

Architecture & Design

1. When would you choose 2PC over Saga pattern?

- Consistency requirements
- Latency tolerance
- System scale considerations

2. How do you decide between choreography and orchestration for Sagas?

- Team structure and ownership
- Debugging and monitoring needs
- Coupling trade-offs

3. What are the risks of eventual consistency?

- User experience implications
- Business logic complexity
- Conflict resolution strategies

Real-World Scenarios

1. Design a distributed transaction for a flight + hotel booking system

- Multiple external providers
- Partial booking handling
- Timeout and cancellation policies

2. How would you handle a payment that succeeds but order creation fails?

- Compensation timing

- Customer communication
- Audit trail requirements

Failure Handling

1. What happens when compensation itself fails?

- Retry strategies
- Dead letter queues
- Manual intervention workflows

2. How do you ensure idempotency in distributed transactions?

- Idempotency keys
- Deduplication strategies
- At-least-once vs exactly-once semantics

Advanced Topics

1. How do you implement distributed transactions across different databases?

- Heterogeneous systems (SQL + NoSQL)
- Cross-cloud transactions
- Legacy system integration

2. What's your approach to testing distributed transactions?

- Chaos engineering
- Contract testing
- End-to-end transaction testing