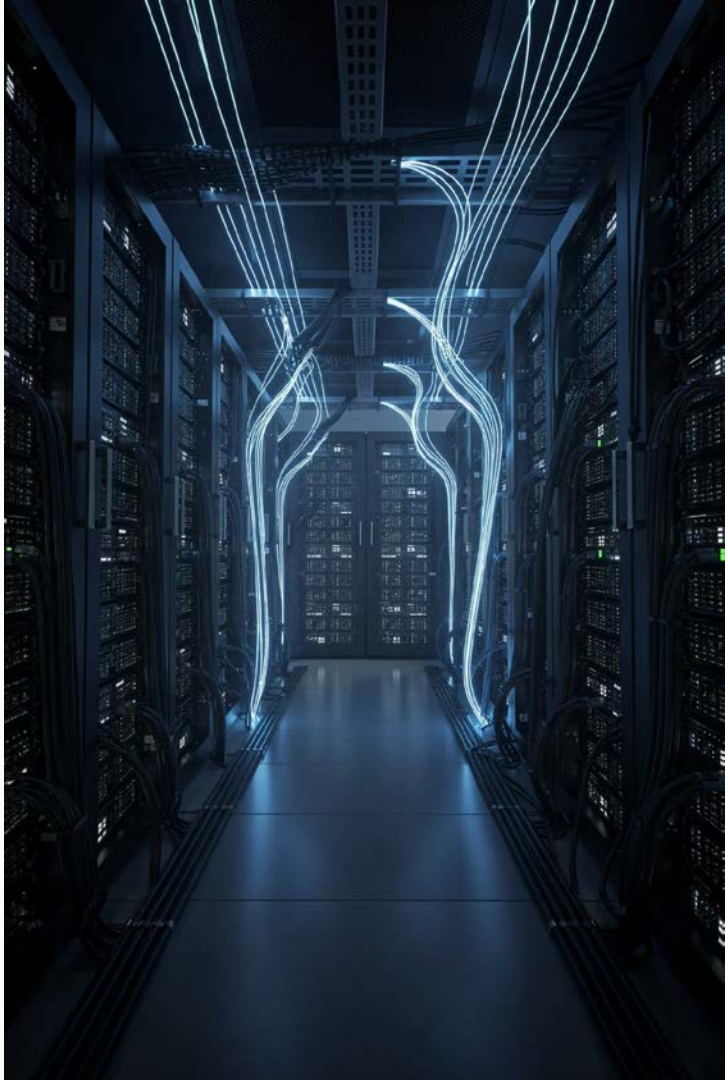
Real-Time Payments for Database DevOps: Scalable, Secure Systems

PRIYATHAM NAGAIYA SEENU NAIDU

LEAD SOFTWARE ENGINEER, MASTERCARD

CONF42 DATABASE DEVOPS 2026



The Challenge

The World Doesn't Wait Anymore

RTP®, FedNow®, UPI, PIX, and SEPA Instant have fundamentally changed what financial infrastructure must deliver. Payment rails are no longer batch-oriented they are always-on, always-processing platforms operating 24/7/365 with sub-second expectations.

Sub-Second Processing

Transactions must settle in milliseconds, not minutes

Near-Zero Downtime

Planned maintenance windows no longer exist

Continuous Load

No off-peak hours volume spikes are unpredictable

A New Era of Payment Networks

Modern payment rails span continents and regulatory jurisdictions. Interoperability, real-time validation, and cross-border settlement are no longer aspirational they are operational requirements.

- **RTP® & FedNow®**

  US domestic real-time rails operated by The Clearing House and the Federal Reserve

- **UPI**

  India's Unified Payments Interface billions of transactions monthly

- **PIX & SEPA Instant**

  Brazil and EU instant payment schemes driving financial inclusion and cross-border reach

Session Roadmap

What We'll Cover Today

01

From Batch to Real-Time

The architectural shift driving event-driven systems

02

Database Engineering

Consistency models, schema evolution, and high-throughput writes

03

Infrastructure Patterns

Microservices, Kubernetes, and active-active multi-region deployments

04

DevOps & Observability

Zero-trust security, AI-driven monitoring, and fraud detection

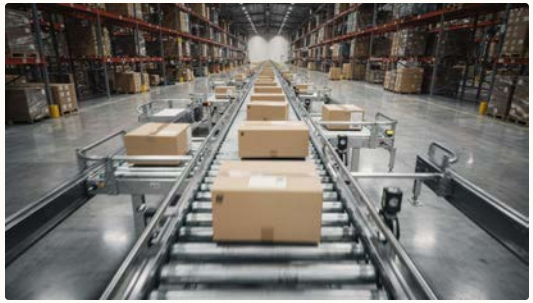
05

Practical Takeaways

Actionable patterns for resilient, data-intensive payment platforms

Chapter 1 · Event-Driven Architecture

From Batch Processing to Continuous Settlement



Batch processing was designed for predictable, scheduled workloads. Real-time payments demand a different paradigm one where every transaction is an event processed the moment it occurs.

Apache Kafka

Durable, distributed event streaming with exactly-once delivery semantics

NATS & RabbitMQ

Lightweight messaging for low-latency, back-pressure-aware pipelines

Event Streaming: The Backbone of Real-Time Payments

Exactly-once semantics, millisecond latency, and reliable back-pressure handling are non-negotiable. Kafka's partition model enables horizontal throughput scaling while preserving per-account event ordering critical for financial consistency.

Chapter 2 · Database Engineering

Consistency, Schema Evolution & High-Throughput Writes

Consistency Models

Strong vs. eventual consistency trade-offs under distributed transaction loads choose deliberately based on payment type

Schema Evolution

Zero-downtime migrations using backward-compatible changes, feature flags, and schema registries

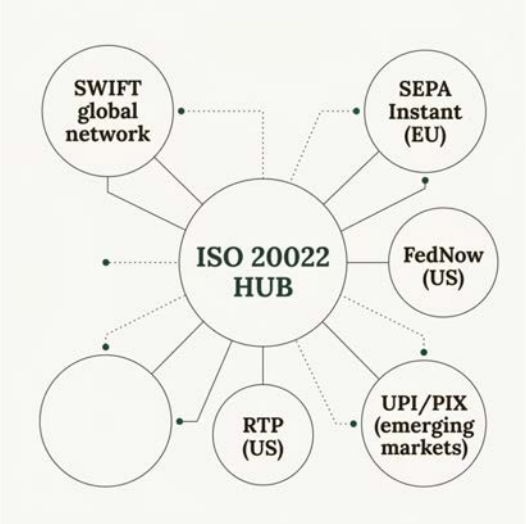
High-Throughput Writes

Write-optimized storage engines, connection pooling, and idempotency keys to handle burst transactional load

ISO 20022: A Common Language for Global Payments

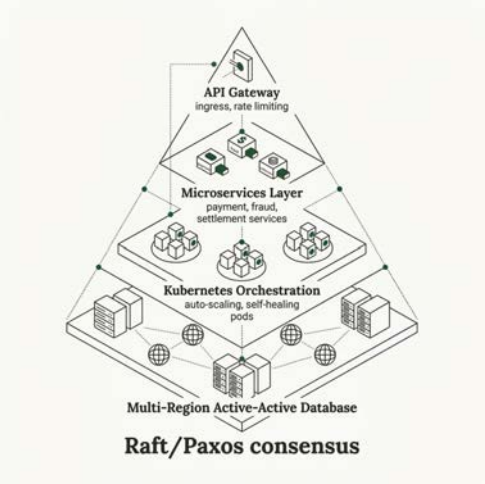
ISO 20022 is more than a messaging standard it is the foundation for real-time validation, rich data exchange, and global interoperability across payment schemes.

- Structured data enables automated straight-through processing
- Rich remittance fields reduce manual reconciliation overhead
- Schema-driven validation enforces data quality at ingestion
- Adopted by SWIFT, SEPA, FedNow, and RTP globally



Chapter 3 · Infrastructure Patterns

Cloud-Native Architecture for Always-On Payments



Why This Stack?

Each layer addresses a specific failure domain. Microservices isolate blast radius. Kubernetes automates recovery. Active-active databases eliminate single points of failure across geographic regions.

- **Microservices**
  Independent deploy, scale, and failure isolation per domain
- **Kubernetes**
  Container orchestration with self-healing and rolling deployments
- **Active-Active**
  Multi-region writes with consensus-based conflict resolution

Consensus Algorithms: Raft & Paxos in Practice

Distributed payment systems rely on consensus to maintain data integrity across nodes and regions. Raft's leader election model offers operational simplicity; Paxos underpins foundational distributed database designs.

Raft

Leader-follower model with clear log replication easier to reason about in operations and debugging

Paxos

Proven correctness guarantees; foundational to CockroachDB, Spanner, and distributed ACID transactions

Fault Tolerance

Quorum-based writes ensure no data loss even when minority of nodes fail simultaneously

Chapter 4 · DevOps & Observability

You Can't Secure What You Can't See

In real-time payment systems, observability is not a luxury it is a safety net. Every latency spike, queue depth anomaly, or failed settlement is a signal requiring immediate triage.

Distributed Tracing

End-to-end request visibility across microservices using OpenTelemetry

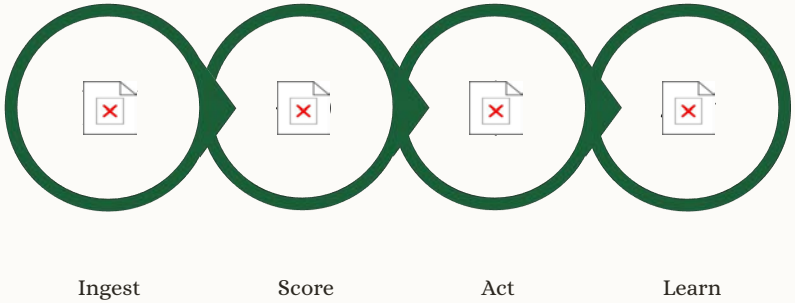
Metrics & Alerting

Latency percentiles, error rates, and throughput as primary SLIs

Structured Logging

Correlation IDs linking every log event to a specific payment transaction

AI-Driven Monitoring: Fraud Detection Within Latency Constraints



Fraud detection must operate within the transaction's own latency budget a model that takes 500ms has no place in a sub-second payment flow. Feature engineering on streaming data and model serving at the edge are key architectural choices.

Zero-Trust Security & Compliance Guardrails

- **Zero-Trust Architecture**

  Every service, user, and API call is authenticated and authorized no implicit trust based on network location

- **PCI DSS**

  Encryption at rest and in transit, tokenization, and strict access controls across cardholder data environments

- **PSD2**

  Strong Customer Authentication and Open Banking API mandates driving secure-by-design integration patterns

Key Takeaways

Building Resilient Payment Platforms: What Matters Most

- **Design for events, not batches**
  Event-streaming platforms with exactly-once semantics are the foundation of real-time payment reliability
- **Consistency is a deliberate choice**
  Understand your consistency model strong or eventual and enforce it explicitly at the database layer
- **Observability is a first-class citizen**
  Instrument everything. Distributed tracing, latency percentiles, and correlation IDs are non-negotiable in production
- **Security must be embedded, not bolted on**
  Zero-trust, PCI DSS, and PSD2 compliance shape architecture from day one not as an afterthought

Thank you!