



# Architecture for Real-Time Air Quality Monitoring and Personalized Health Recommendations in Bogotá

Jose Alejandro Cortazar López, Johan Esteban Castaño Martínez, Stivel Pinilla Puerta  
Systems Engineering Program - Francisco José de Caldas District University

Air pollution is a critical public health challenge. This work presents a production-ready architecture for **real-time air quality monitoring in Bogotá** that integrates heterogeneous data sources (AQICN, Google Air Quality API, IQAir) into a unified 3NF PostgreSQL schema with sub-100 millisecond query latencies and personalized health recommendations.

**Primary Goal:** Design and validate a production-ready architecture integrating multi-source air quality data with normalized schema and personalized health recommendations.

## Specific Objectives:

- Integrate AQICN, Google, & IQAir with 216 readings/hour via Python APScheduler
- Implement 3NF PostgreSQL schema (8 entities) with temporal partitioning
- Achieve sub-100ms query latencies across 85,000+ readings
- Support 50–100 concurrent users with documented concurrency mitigation
- Deliver rule-based health recommendations per EPA/WHO AQI standards

## 8 Normalized Entities:

Database Schema - 8 Entities (3NF)		
Geospatial & Monitoring		
Station	Pollutant	Provider
AirQualityReading	AirQualityDailyStats	
Users & Access		
AppUser	Alert	Recommendation
Key Features:		
• Monthly partitioning (85K+ readings)		
• Composite B-tree indexes		
• Materialized views (35× reduction)		
• Referential integrity constraints		

*Relational schema with referential integrity, monthly partitioning, and composite indexes*

## Key Features:

- Temporal Partitioning:** Monthly tables for 78% latency improvement
- Materialized Views:** 35× row reduction (85K → 2.4K)
- Composite Indexes:** B-tree on (station, pollutant, datetime)
- MVCC Isolation:** Zero-downtime concurrent reads/writes

Scalability & Performance
Current Performance:
50-100 concurrent users
70-75% CPU usage
Sub-150ms latency
140 req/sec throughput
Scaling Strategies:
Vertical: 8+ vCPUs → 1,000+ users
Horizontal: Read replicas + Load balancer
Partitioning: Monthly tables (78% improvement)
10-Year Projection:
525M+ readings
Partition pruning
Consistent sub-200ms latency

## Current Capacity:

- 50–100 concurrent users (70–75% CPU)
- 140 requests/second throughput
- Sub-150ms latency at 500 users

## Scaling Strategies:

- Vertical:** 8+ vCPUs → 1,000+ users
- Horizontal:** Read replicas + load balancer
- Partitioning:** 78% latency improvement (10-year projection)