Faysal Aziz

PostgreSQL Performance Engineering

Advanced planner insight, query tuning, and time-series architecture for high-performance PostgreSQL systems - 2 days

FA-0353 · 2 days

Data & Analytics

Introduction

This advanced two-day hands-on course is designed for experienced SQL and PostgreSQL professionals who want to go beyond writing correct queries and learn how to make them fast, predictable, and scalable. The course focuses heavily on PostgreSQL optimization, especially planner behavior, execution internals, indexing strategy, statistics, and performance tuning techniques used in real production systems.

It also covers TimescaleDB from both an implementation and architecture perspective, including hypertable-based design and the role of columnar storage for analytical workloads. The instructor brings over 30 years of industry experience and teaches using practical, industry-demanded content rather than an academic approach.

Learning outcomes

- Read and interpret advanced PostgreSQL execution plans confidently
- Identify root causes of poor query performance
- Understand planner behavior, costing, and row estimation issues
- Tune queries using indexing, statistics, SQL rewrites, and configuration
- Distinguish key plan nodes and know when each is beneficial or problematic
- Design PostgreSQL databases with TimescaleDB implementation in mind
- Use TimescaleDB hypertables, compression, and columnstore appropriately
- Make better architecture decisions for high-ingest and analytical workloads

Prerequisites

- Strong SQL expertise
- Hands-on PostgreSQL experience
- Familiarity with indexes, joins, vacuum, analyze, and partitioning
- Comfortable reading basic EXPLAIN plans

Training outline

1. PostgreSQL performance tuning mindset

1. Performance goals and bottleneck identification
2. Query-level tuning versus workload-level tuning
3. Benchmarking and tuning workflow

2. PostgreSQL internals that affect performance

1. MVCC, heap storage, visibility, and tuple lifecycle
2. WAL, checkpoints, autovacuum, and bloat

3. Planner and executor deep dive

1. Parse, rewrite, plan, execute
2. Cost-based optimization fundamentals
3. Planning time versus execution time
4. Expected row count versus actual row count
5. Filtered rows and rows removed by filter
6. Join order and join strategy selection

4. Reading execution plans effectively

1. Cost, rows, width, loops, actual time
2. Buffers, I/O, timing, and memory clues
3. Finding the first bad estimate
4. Diagnosing plan instability and regression

5. Understanding critical plan nodes

1. Sequential Scan
2. Index Scan and Index Only Scan
3. Bitmap Index Scan and Bitmap Heap Scan
4. Nested Loop, Hash Join, and Merge Join
5. Sort, Hash, Aggregate, and Materialize
6. Gather, parallel nodes, and Memoize
7. Filter versus Join Filter

6. Advanced indexing strategy

1. Composite indexes and column order
2. Partial indexes
3. Expression indexes
4. BRIN for large time-oriented datasets
5. Index bloat, redundancy, and maintenance

7. SQL tuning patterns

1. SARGability and predicate design
2. Join rewrites and pre-aggregation
3. Window function performance
4. CTE and subquery performance considerations
5. Pagination and top-N query strategies

8. Memory, parallelism, and execution behavior

1. Hash and sort memory usage
2. Shared buffers and cache effects
3. Parallel query behavior

9. PostgreSQL architecture for performance

1. Scale-up versus scale-out decisions
2. Connection management and pooling
3. Read replicas and workload separation

4. Partitioning strategy and pruning

10. Database design assuming TimescaleDB

1. Designing for hypertables from the start
2. Time column and partitioning strategy
3. Chunk interval design
4. Indexing hypertables effectively
5. High-cardinality dimension design
6. Retention-aware schema design
7. Designing for continuous aggregates

11. TimescaleDB optimization

1. Hypertables and chunk pruning
2. Ingestion performance patterns
3. Compression and columnstore usage
4. Continuous aggregates and refresh strategy
5. Retention policies and lifecycle management
6. Query patterns for recent versus historical data

12. Columnar capabilities and hybrid storage

1. PostgreSQL row-oriented fundamentals
2. TimescaleDB columnstore for analytical workloads
3. When columnstore helps
4. When rowstore remains the better choice
5. Designing hybrid hot-and-cold data strategies

© 2026 Faysal Aziz · Kuala Lumpur