Oracle Database Migrations to EDB Postgres®: Proven Approaches with EDB Postgres

Raghavendra Rao

enterprisedb.com

Table of Contents

Objective

This comprehensive best-practices guide provides an industry-standard approach for migrating enterprise **Oracle** databases to **EDB Postgres Advanced Server**, leading IT professionals through the complete migration lifecycle from initial assessment to production cutover. Using the proven **ACTIONS** framework and EDB-recommended tools, it addresses the significant technical differences between Oracle and EDB Postgres platforms while providing practical solutions to common migration challenges.

The guide leverages EDB's **built-in Oracle compatibility** features—including PL/SQL support, packages, and Oracle-compatible data types—to achieve **60%–80% automatic code conversion**, reducing migration timelines by 40%–60% compared to standard PostgreSQL migrations. By following these best practices, organizations can minimize business risk, preserve critical Oracle functionality with minimal refactoring, ensure optimal performance and security, and enable cost-effective digital transformation through a seamless transition to an enterprise-grade, open source database platform.

Introduction to Oracle migrations

Migration journey overview

Oracle to EDB Postgres migrations follow a structured journey through the ACTIONS framework: **Assessment** (analyzing Oracle features and dependencies), **Conversion** (schema and code transformation), **Tooling** (selecting migration tools), **Implementation** (data migration execution), **Operations** (monitoring setup), **Normalization** (validation), and **Switch** (production cutover). This proven methodology ensures predictable outcomes while minimizing risk.

Migration catalysts

Organizations typically migrate from Oracle due to **license cost reduction** (eliminating expensive Oracle licensing), **cloud modernization** (adopting cloud-native PostgreSQL), **vendor independence** (escaping Oracle lock-in), **compliance requirements** (meeting open source mandates), and **scalability needs** (horizontal scaling with PostgreSQL). Each catalyst drives different priorities in the migration approach.

Technical advantages of EDB Postgres

- **Oracle compatibility benefits:** EDB Postgres Advanced Server provides **native PL/SQL execution**, eliminating extensive code rewrites. **Oracle packages** (DBMS_, UTL_) run directly, preserving business logic. **Compatible data types** (NUMBER, VARCHAR2, DATE) reduce conversion effort. **Oracle-style triggers and functions** work with minimal changes, achieving 60%–80% automatic compatibility.
- **Enterprise features:** Beyond compatibility, EDB Postgres offers **transparent data encryption**, **SQL Protect** for injection prevention, **EDB Failover Manager** for automatic high availability, and **Oracle-compatible partitioning**. These features ensure that enterprise requirements are met while reducing total cost of ownership by up to 80% compared to Oracle.
- **Migration acceleration:** EDB tools and compatibility features typically reduce migration effort by 40%–60% compared to community PostgreSQL, enabling faster ROI and lower project risk while maintaining Oracle's enterprise capabilities.

Migration approach framework

We will use the **ACTIONS** framework—a proven, seven-phase methodology for migrating from **Oracle to EDB Postgres**.

The framework provides a repeatable process that guides IT professionals through the entire migration lifecycle, minimizing risks and ensuring successful outcomes. Each phase builds upon the previous one, creating a logical progression from understanding the current state to executing the migration and establishing ongoing operations.

The **ACTIONS** framework:

A - Architectural compatibility and assessment: Analyze current state, evaluate compatibility differences, identify application challenges, and define migration requirements.

C - Conversion (schema and DDL): Convert database structures and definitions from source syntax to PostgreSQL-compatible formats.

T - Tooling and planning: Select migration tools, develop detailed plans, establish success criteria, and implement risk-mitigation strategies.

I - Implementation (data migration): Execute data migration, optimize performance, and handle data transformations.

O - Operation: Establish monitoring, automation, and observability for ongoing database health post-migration.

N - Normalization and validation: Conduct compliance checks, data validation, and testing to ensure accuracy and regulatory adherence.

S - Switch and support: Execute controlled cutover, provide post-migration monitoring, and complete operational handoff.

This framework applies consistently across Oracle, SQL Server, CockroachDB, and MongoDB while addressing each platform's unique architectural differences and migration challenges. Each database section follows this process with platform-specific guidance.

Migration from Oracle

A - Architectural differences and assessment

Core architectural differences

- **Instance vs. database model:** Compare Oracle's **instance-database** architecture with SGA/PGA memory structures versus EDB Postgres's simpler **cluster-database** model: Oracle **RAC** for multi-node clustering has no direct equivalent, requiring application-level changes or read replicas. **Tablespaces and data files** map differently to PostgreSQL's file-per-table approach. **Undo/redo** architecture versus PostgreSQL's WAL and MVCC implementation.
- **Process architecture:** Compare Oracle's **background processes** (PMON, SMON, DBWn) versus PostgreSQL's simpler process model: **Dedicated/shared server** modes versus PostgreSQL's process-per-connection requiring connection pooling. **Automatic Memory Management** versus manual configuration of shared_buffers and work_mem. **Result cache** and **buffer cache** differences affecting performance tuning.
- **High-availability differences:** Compare Oracle **Data Guard** with physical/logical standby versus EDB Postgres streaming replication. **Fast-Start Failover** versus EDB Failover Manager for automatic failover. **Flashback** technologies require different approaches in PostgreSQL. **Active Data Guard** for read-only replicas versus PostgreSQL hot standby.

SQL and feature compatibility with EDB

- **EDB Oracle compatibility:** EDB supports **85% of PL/SQL** syntax including packages, triggers, and functions. **Oracle packages** (DBMS_OUTPUT, UTL_FILE, DBMS_JOB) work directly. **Oracle data types** (VARCHAR2, NUMBER, DATE with time) are native. **Hierarchical queries** (CONNECT BY) are supported, unlike community PostgreSQL.
- **Remaining differences, RAC-specific features:** These require application redesign. **Advanced Queuing** needs alternative message queue solutions. **Materialized view fast refresh** is not fully supported. **Database Vault** and **Label Security** need different security approaches. **Flashback Query/Table** requires trigger-based solutions.

Application impact assessment

- **Connection layer changes:** These replace **OCI/JDBC thin drivers** with EDB-compatible drivers maintaining most connection parameters. **TNS names** resolution changes to standard host/port configuration. **Connection pooling** moves from Oracle shared servers to PgBouncer/Pgpool.
- **Global temporary tables:** These work differently in session scope.
- **PL/SQL code migration:** Most **PL/SQL procedures** run unchanged in EDB (70%–80% compatibility). **Packages** maintain structure with minor syntax adjustments. **REF CURSORS** and **BULK COLLECT** supported with slight modifications. **Dynamic SQL** requires minor EXECUTE IMMEDIATE adjustments. **Autonomous transactions** are supported through PRAGMA implementation.

Assessment methods and tools

- **EDB assessment solutions:** The **EDB Migration Portal** provides web-based assessment without installing software, analyzing Oracle databases remotely to identify compatibility and complexity. **EDB Migration Toolkit** in assessment mode evaluates database objects and generates compatibility reports. **EDB Professional Services** offers custom assessment tools for single to bulk database evaluations with detailed migration roadmaps.
- **Third-party assessment tools: Ispirer SQLWays** performs comprehensive Oracle assessment with automated complexity scoring and effort estimation. **ora2pg** with `--type SHOW_REPORT` provides detailed migration analysis with A–E complexity ratings and person-day estimates.
- **AWS SCT** generates assessment reports for Oracle to PostgreSQL/Aurora migrations.
- **Manual Oracle analysis:** Query **DBA_FEATURE_USAGE_STATISTICS** for licensed features requiring alternatives. Check **V$SQL** for most-executed statements needing performance validation. Review **DBA_DEPENDENCIES** for complex object relationships. Analyze **AWR reports** for workload patterns and resource usage.

C - Conversion (schema and DDL)

Schema conversion approach

- **Automated conversion with EDB tools: EDB Migration Portal** provides web-based assessment and schema conversion without local installation, analyzing Oracle databases remotely and generating downloadable DDL scripts. **EDB Migration Toolkit** converts 80%–85% of Oracle schemas automatically, including tables, indexes, constraints, views, and PL/SQL objects. Both tools preserve **Oracle-compatible mode** for VARCHAR2, NUMBER, and DATE behaviors while keeping **package structures** intact.
- **Manual conversion requirements: Index-organized tables** need conversion to regular tables with clustered indexes. **Partitioning syntax** requires adjustment despite EDB support. **Virtual columns** convert to generated columns or views. **External tables** need Foreign Data Wrapper configuration. **RAC-specific objects** (services, TAF configurations) require removal or redesign.

Key conversion challenges

- **Oracle-specific SQL features: CONNECT BY NOCYCLE** is supported in EDB but may need optimization. **MODEL clause** requires complete rewrite using CTEs. **PIVOT/UNPIVOT** needs crosstab extension or CASE statements. **Flashback queries** require temporal table design. **MERGE statement** works in EDB with minor syntax adjustments.
- **PL/SQL to EDB SPL packages:** This works directly but might need **EDITIONABLE** keyword removal. **PRAGMA directives are mostly** supported except for RESTRICT_REFERENCES. **Dynamic SQL** with EXECUTE IMMEDIATE requires minor formatting. **Collections and arrays** need syntax adjustments for bulk operations. **Exception handling** preserves Oracle error codes in EDB.
- **Storage and performance objects: Tablespace management** is simplified in PostgreSQL, with less granular control. **Materialized views** lose fast refresh on commit capability. **Function-based indexes** are supported but display different optimizer behavior. **Bitmap indexes** convert to B-tree or GIN indexes. **Result cache** features need application-level caching.

Conversion tools and workflow

- **EDB conversion solutions: Migration Portal** is for initial remote assessment and DDL generation. **Migration Toolkit** supports integrated schema and data migration with dependency management. **EDB SPL Check** extension validates converted PL/SQL code, performing syntax checking, semantic analysis, and identifying performance issues in stored procedures.
- **Third-party tools:** With **ora2pg**, use `SHOW_REPORT` for assessment, then `DDL` export for schema conversion with selective object migration. **Ispirer SQLWays** is a commercial tool with intelligent PL/SQL pattern recognition and optimization recommendations.

Validation and testing

- **Code validation:** Use **EDB SPL Check** to validate all converted PL/SQL procedures and packages. Run `spl_check_function()` for individual functions or batch validation. Review warnings for performance anti-patterns and deprecated syntax.
- **Schema verification:** Compare object counts between Oracle and EDB. Validate dependencies and reference integrity. Test sample procedures with production-like data. Document remaining incompatibilities for manual resolution.

T - Tooling and planning

Tool selection strategy

- **EDB migration suite:** Use **Migration Portal** for cloud-based assessment and DDL conversion without infrastructure setup. **Migration Toolkit** provides comprehensive schema and data migration with Oracle compatibility. **EDB SPL Check** is for PL/SQL code validation post-conversion. Use **LiveCompare** for data validation between Oracle and EDB during migration.
- **Supplementary tools: ora2pg** provides detailed complexity assessment and custom migration scenarios. **EDB Replication Server** offers near-zero downtime migrations via CDC. **pgBackRest** or **Barman** are backup strategies replacing RMAN. **PgBouncer** is for connection pooling, replacing Oracle shared servers.
- **Tool selection criteria:** Use Migration Portal for initial assessment and POC. Deploy Migration Toolkit for production migrations with EDB compatibility. Add Replication Server for minimal downtime requirements. Plan **70% effort on schema/code conversion**, 30% on data migration.

Migration planning framework

- **Phase 1: Discovery:** Run Migration Portal assessment for complexity scoring. Analyze AWR reports for performance baselines. Identify RAC dependencies and Oracle-specific features. Size EDB infrastructure based on Oracle resource usage.
- **Phase 2: Preparation:** Convert schemas using Migration Toolkit, validate with SPL Check. Set up an EDB test environment with production-like data. Configure Replication Server for CDC if required. Train team on EDB administration and PL/SQL differences.
- **Phase 3: Execution:** Execute phased migration: noncritical → read-heavy → core systems. Run LiveCompare for continuous data validation. Test application functionality with the converted database. Document and resolve remaining incompatibilities.

Critical planning considerations

- **Complexity multipliers:** Add **2x time** for heavy PL/SQL usage (packages, procedures). Factor **3x effort** for RAC to single-instance conversion. Include a 2x buffer for applications using Oracle-specific features. Simple migrations average 4–6 weeks; average complex Oracle systems last 3–4 months.
- **Risk mitigation:** Test EDB SPL. Check early to identify code issues. Validate that LiveCompare can handle your data volumes. Plan for Oracle features without EDB equivalents (Flashback, Advanced Queuing). Maintain Oracle for 30-day rollback capability.
- **Success criteria:** Proceed if PL/SQL compatibility exceeds 70% and single-instance performance meets SLAs. Reconsider whether the project requires RAC active-active or Oracle-specific advanced features. Evaluate EDB Enterprise features versus community PostgreSQL based on requirements.

I - Implementation (data and migration)

Migration strategy selection

- **Offline migration (maintenance window):** Use **EDB Migration Toolkit** for complete schema and data migration during downtime; best for databases under 500GB with 4–8 hour maintenance windows. It provides the highest data consistency and simplest execution.
- **Online migration (near-zero downtime):** Deploy **EDB Replication Server** for trigger-based or log-based CDC from Oracle to EDB. It enables continuous sync with minimal downtime for cutover and is required for 24x7 systems or databases over 1TB.
- **Hybrid approach:** Initially, load with Migration Toolkit, then Replication Server for incremental changes. This is recommended for most production migrations and allows an extended testing period before cutover.

Implementation tools

- **EDB Migration Toolkit:** Configure with `-dataOnly` for data migration after schema conversion. Use `-batchSize` parameter for memory optimization (default 1000 rows). Enable parallel processing with `-loaderCount` for multiple tables. Monitor progress through detailed logging and status updates.
- **EDB Replication Server:** Set up **publication** on the Oracle source using trigger or LogMiner-based capture. Configure **subscription** on the EDB target with conflict resolution rules. This supports filtered replication and transformation rules and enables bidirectional sync for phased migrations.
- **Data validation with LiveCompare:** Configure connections to both Oracle and EDB databases. Define comparison rules and exclusions for system columns. Run continuous comparisons during the migration window. Generate detailed reports of data discrepancies.

Performance optimization techniques

- **Pre-migration tuning:** This disables foreign keys and triggers during bulk loads. Set `maintenance_work_mem = 4GB, checkpoint_segments = 256`. Increase `max_wal_size = 10GB` for large transactions. Configure `synchronous_commit = off` temporarily.
- **During migration:** Use Migration Toolkit's -truncLoad to clear target tables before loading. Enable `-dataOnly -constraints` to defer constraint creation. Run multiple toolkit instances for different schemas in parallel. Monitor Oracle wait events and EDB pg_stat_activity.
- **Post-migration optimization:** Reenable constraints with `ALTER TABLE ... VALIDATE CONSTRAINT`. Run `VACUUM ANALYZE` on all migrated tables. Reset configuration parameters to production values. Execute the LiveCompare final validation before cutover.

Cutover strategy

- **Final sync steps:** Stop application writes to Oracle or enable read-only mode. Process final Replication Server changes or run last Migration Toolkit sync. Validate critical data with LiveCompare spot checks. Update sequences to match Oracle current values.
- **Validation checkpoints:** Verify that row counts match between systems. Confirm that PL/SQL packages execute without errors. Test application connectivity and basic transactions. Document any data discrepancies for business acceptance.

O - Operation (monitoring, automation, observability)

Monitoring essentials

- **Performance monitoring transition:** Replace Oracle **AWR/ASH** reports with EDB's **pg_stat_statements** and **edb_wait_states**. Monitor **EDB Advanced Server**–specific views such as `edb_session_waits` for Oracle-like wait analysis. Transition from Oracle Enterprise Manager to **PEM (Postgres Enterprise Manager)** for comprehensive monitoring. Track **PL/SQL performance** using EDB SPL profiler, replacing Oracle's DBMS_PROFILER.
- **Key metrics migration:** Shift from Oracle **V$SYSSTAT** to PostgreSQL **pg_stat_database** and custom metrics. Replace **alert log** monitoring with PostgreSQL log analysis using **pgBadger**. Monitor **tablespace usage** differently due to PostgreSQL's simpler storage model. Track **vacuum activity** and **bloat**, concepts not present in Oracle.

Automation framework

- **Job scheduling migration:** Convert Oracle **DBMS_SCHEDULER** jobs to **EDB DBMS_JOB** package or **pg_cron**. Migrate Oracle **maintenance windows** to PostgreSQL vacuum and analyze schedules. Replace **automatic statistics gathering** with custom analyze scripts. Transform **data pump** export schedules to pgBackRest or Barman policies.
- **Operational automation:** Configure **EDB Failover Manager** to replace Oracle Data Guard broker. Set up **automated backups** with pgBackRest replacing RMAN scripts. Implement **connection pooling** with PgBouncer for Oracle shared server replacement. Create **partition maintenance** scripts replacing Oracle interval partitioning.

Observability platform

- **EDB-specific monitoring:** Deploy **Postgres Enterprise Manager (PEM)** for enterprise monitoring replacing OEM. Configure **EDB Audit** for compliance, replacing Oracle Audit Vault. Monitor **Oracle compatibility layer** performance with EDB-specific metrics. Track **SPL function execution** statistics and performance patterns.
- **Alerting configuration:** Set alerts for **PL/SQL errors** in EDB SPL functions. Monitor **Oracle package** execution failures and performance degradation. Configure **replication lag** alerts for EDB Replication Server. Track **connection exhaustion** without Oracle's connection multiplexing. Alert on **vacuum freeze** approaching replacing Oracle's automatic segment management.
- **Integration points:** Connect PEM to existing monitoring tools via REST API. Export metrics to **Prometheus/ Grafana** for unified dashboards. Configure **log shipping** to centralized logging platforms. Integrate with ticketing systems for alert management.

N - Normalization and validation

Data validation strategy

- **Row count verification:** Compare Oracle's `SELECT COUNT(*) FROM user_tables` with EDB's table counts. Use **LiveCompare** for automated row count validation across all tables. Account for Oracle **partitioned tables** mapped to PostgreSQL partitions. Document expected differences from filtered or excluded objects.
- **Data integrity with LiveCompare:** Configure **LiveCompare** for continuous data comparison during migration. Define comparison rules excluding Oracle-specific columns (ROWID, ORA_ROWSCN). Set tolerance levels for NUMBER to NUMERIC precision differences. Generate detailed discrepancy reports for investigation.
- **NULL and empty string handling critical difference:** Oracle treats empty strings ('') as NULL, while PostgreSQL maintains them as distinct values. **EDB solution:** Enable `edb_redwood_strings = true` in postgresql.conf to make EDB treat empty strings as NULL, as Oracle does. For VARCHAR2 columns, EDB automatically handles this in Oracle-compatible mode. **Alternative approach:** Create database triggers to convert empty strings to NULL on INSERT/UPDATE if redwood mode isn't suitable. This eliminates most application changes for NULL handling.

PL/SQL validation

- **Code functionality testing:** Use **EDB SPL Check** to validate all converted procedures and packages. Run `spl_check_function()` for syntax and semantic verification. Execute test cases for critical business logic procedures. Compare Oracle DBMS_OUTPUT with EDB equivalents.
- **Package validation:** Verify that **Oracle packages** (DBMS_, UTL_) work correctly in EDB. Test **package variables** and **initialization blocks** behavior. Validate **cursor handling** and **REF CURSOR** functionality. Confirm that **exception handling** preserves Oracle error codes.

Schema validation

- **Object completeness:** Compare Oracle **DBA_OBJECTS** count with EDB catalog views. Verify that all **constraints** are maintained, including check, unique, and foreign keys. Validate that **indexes** are converted appropriately, including function-based indexes. Confirm that **sequences** current values match Oracle.
- **Oracle compatibility features:** Test that **hierarchical queries** (CONNECT BY) return the same results. Verify that **synonyms** and **database links** work as expected. Validate **global temporary tables** session behavior. Confirm that **Oracle data types** (VARCHAR2, NUMBER) behave correctly.

Compliance and sign-off

- **Performance baseline comparison:** Run critical queries on both platforms using **EXPLAIN PLAN** comparison. Document performance differences as new baselines. Verify that **PL/SQL execution times** are within acceptable ranges. Validate that batch job completion times meet SLAs.
- **Acceptance criteria:** Achieve 100% data migration, verified by LiveCompare. Verify that all critical PL/SQL packages are executing without errors and that performance is within 20% of Oracle baselines (some queries may be faster). Verify that business process validation has completed successfully.
- **Final validation checklist:** LiveCompare shows zero critical data discrepancies. SPL Check reports no severe issues in converted code. Application testing has completed with no Oracle-specific errors. Backup and recovery tested successfully on EDB platform.

S - Switch and support

Go-live preparation

- **Cutover rehearsal:** Execute complete migration in staging with production data volumes. Test with **edb_redwood_strings** and Oracle compatibility settings enabled. Validate PL/SQL packages and critical business processes. Document exact cutover timings and rollback procedures.
- **Final configuration check:** Verify EDB Oracle compatibility parameters: `edb_redwood_strings`, `oracle_mode`, `edb_stmt_level_tx`. Configure **EDB Failover Manager** for automatic failover readiness. Set up **PEM monitoring** alerts and dashboards. Validate **pgBackRest** or Barman backup schedules.

Application cutover

- **Connection updates:** Replace Oracle TNS names with EDB connection strings. Update JDBC URLs from `jdbc:oracle:thin` to `jdbc:edb://`. Maintain most connection parameters due to EDB compatibility. Configure **PgBouncer** for connection pooling if needed.
- **Cutover execution:** Enable Oracle read-only or restricted mode. Run final EDB Replication Server sync or Migration Toolkit delta load. Execute **LiveCompare** for final validation. Switch application connections to EDB. Keep Oracle running for immediate rollback option.

Post-migration support

- **Stabilization monitoring (first 72 hours):** Monitor **PL/SQL execution** errors in PostgreSQL logs. Track **Oracle package** performance in production load. Watch for **connection pool** sizing issues. Check **vacuum and analyze**, keeping up with transaction volume. Monitor for any NULL/empty string handling issues.
- **Knowledge transfer:** Train DBAs on **EDB-specific features**: SPL debugging, Oracle package management, redwood mode implications. Document differences in **backup/recovery** using pgBackRest versus RMAN. Explain **PEM** navigation for Oracle DBAs familiar with OEM. Create runbooks for common EDB administration tasks.
- **Oracle to EDB operational differences:** Document how **tablespace management** differs from Oracle. Explain **vacuum/analyze** versus Oracle's automatic segment management. Clarify **WAL archiving** versus Oracle archive logs. Define new procedures for **partition maintenance** without Oracle's automation.

Success handoff

- **Acceptance criteria:** These include stable operation for 14 days without critical issues. All PL/SQL packages are running successfully. LiveCompare shows that data consistency is maintained. Performance is meeting or exceeding Oracle baselines.
- **Decommission planning:** Maintain Oracle in read-only for 30 days as a safety net. Archive Oracle data before shutdown if required for compliance. Document any Oracle features not migrated, for future consideration. Transfer Oracle licenses per vendor requirements.
- **Long-term optimization:** Schedule training on **EDB SPL** optimization techniques. Plan a review of converted PL/SQL for PostgreSQL-native improvements. Consider gradual migration from Oracle packages to PostgreSQL extensions where beneficial. Establish performance-tuning practices specific to EDB Postgres.

Appendix: Tool documentation links

EDB migration suite

Assessment and planning

- EDB Migration Portal: www.enterprisedb.com/docs/migration_portal/latest
- EDB Migration tools overview: www.enterprisedb.com/docs/migrating/oracle/edb_migration_tools

Migration and replication

- EDB Migration Toolkit: www.enterprisedb.com/docs/migration_toolkit/latest
- EDB Replication Server: www.enterprisedb.com/docs/eprs/latest

Validation and testing

- EDB SPL Check: www.enterprisedb.com/docs/pg_extensions/spl_check
- LiveCompare: www.enterprisedb.com/docs/livecompare/latest

Operations and management

- EDB Postgres Advanced Server: www.enterprisedb.com/docs/epas/latest
- Postgres Enterprise Manager (PEM): www.enterprisedb.com/docs/pem/latest
- EDB Failover Manager: www.enterprisedb.com/docs/efm/latest

Third-party migration tools

Assessment and conversion

- ora2pg: ora2pg.darold.net/documentation.html
- AWS Schema Conversion Tool: docs.aws.amazon.com/SchemaConversionTool/latest/userguide
- Ispirer SQLWays: www.ispirer.com/products/oracle-to-postgresql-migration

Data migration and CDC

- pgloader: pgloader.readthedocs.io/en/latest
- Debezium: debezium.io/documentation

PostgreSQL ecosystem tools

Backup and recovery

- pgBackRest: https://pgbackrest.org/user-guide.html
- Barman: docs.pgbarman.org

High availability

- Patroni: patroni.readthedocs.io/en/latest

Connection and performance

- PgBouncer: www.pgbouncer.org/usage.html
- Pgpool-II: www.pgpool.net/docs/latest/en/html
- pgBadger: pgbadger.darold.net/documentation.html

Additional resources

- EDB documentation center: www.enterprisedb.com/docs
- EDB Migration Handbook: www.enterprisedb.com/docs/migrating/oracle
- EDB Oracle compatibility guide: www.enterprisedb.com/docs/epas/latest/epas_compat_ora_dev_guide
- PostgreSQL Documentation: www.postgresql.org/docs

About the author

Raghavendra Rao
Senior Practice Leader, Global Migrations, EDB
Connect on LinkedIn

Raghavendra Rao is a senior practice leader in global migration and an accomplished database professional with more than two decades of experience across enterprise and open source platforms. A passionate community contributor, he shares his expertise through blogging, presentations, and training, drawing inspiration from the global PostgreSQL community.