CASE STUDY

MeeSeva: How India's Largest G2C Platform Migrated from Oracle to PostgreSQL

A mission-critical database migration completed with zero data loss and zero service disruption, moving 1,939 tables, 805 indexes, and 85+ citizen services from Oracle 11g to PostgreSQL 17 on a live platform serving 200 million cumulative transactions.

Organisation	ESD, ITE&C Dept., Government of Telangana
Platform	MeeSeva https://meeseva.telangana.gov.in
Source Database	Oracle Database 11g Enterprise Edition
Target Database	PostgreSQL 17 on Ubuntu 24.04.3 LTS
Migration Tool	Ora2Pg v24
Migration Date	04 April 2026
Migration Partners	Postgres1st, OpenSource DB
Application Partner	Vupadhi
Status	Completed, Zero data loss, zero service disruption

01. Platform Overview

MeeSeva, Telugu for “At Your Service”, is the Government of Telangana’s flagship e-governance platform. Operated continuously for over 15 years, it delivers more than 500 Government-to-Citizen (G2C) and Government-to-Business (G2B) services across approximately 50 state departments. The platform processes between 80,000 and 1,50,000 transactions per day through roughly 5,000 service centres, a web portal, a mobile application (T App Folio), and a WhatsApp-based interface. Cumulatively, MeeSeva has served nearly 20 Crore (200 million) transactions.

Telangana has consistently topped the Government of India’s Electronic Transaction Aggregation and Analysis Layer (e-Taal) rankings for e-transaction volume among all Indian states. Any disruption to MeeSeva directly impacts citizen access to essential government services, certificates, licences, permits, registrations, utility payments, and pension services. The platform operates under a zero-tolerance policy for data loss and extended outages.

Years of operation	15+
Daily transaction volume	80,000 – 1,50,000
Cumulative transactions	~20 Crore (200 million)
G2C & G2B services	500+
Critical services migrated	85+
Departments	~50
Service centres	~5,000
Source database	Oracle Database 11g Enterprise Edition
Target database	PostgreSQL 17.7 on Ubuntu 24.04.3 LTS
Primary migration tool	Ora2Pg v24.0

02. Why We Migrated, The Business Case

The decision to migrate was driven by four factors:

Vendor independence. Dependence on a single proprietary database vendor for mission-critical public infrastructure created risk, in pricing negotiations, in upgrade timelines, and in the ability to adapt the technology stack to changing requirements.

Scalability and flexibility. PostgreSQL's extensibility supports evolving service demands without proprietary constraints. The open-source ecosystem allows the team to adopt best-of-breed tooling for high availability, backup, monitoring, and performance tuning.

Government open-source policy alignment. The migration aligns with both the Government of Telangana's and the Government of India's broader policy direction encouraging adoption of open-source solutions in government IT systems.

Licence cost elimination. Oracle licensing represented a significant recurring expenditure. PostgreSQL is open-source with zero licence cost. The one-time migration cost was a fraction of a single year's Oracle licensing fee.

MeeSeva's Oracle database infrastructure carried an annual licensing and maintenance cost of approximately ₹10 Crore (roughly USD 1.2 million). For a government platform, this represented a significant and perpetually recurring expenditure on proprietary software, money that could instead be directed toward service improvements, infrastructure expansion, or other public priorities.

Cost Analysis

Cost Item	Amount
Oracle Licensing + maintenance	₹10+ Crore
Oracle Support recurring	₹1.3 Crore / year
One-time PostgreSQL migration cost	< ₹1 Crore
Yearly PostgreSQL support cost	< ₹25 Lakhs
Estimated annual recurring saving	~₹1.5 Crore /year

03. Database Landscape

The migration scope covered three production application databases supporting 85+ critical citizen-facing services:

Database	Function
DASHBOARD	Operational analytics and monitoring
TSPAYGTY	Payment gateway transactions
SACENTRAL	Central portal, citizen services processing

Object Inventory Migrated

Object Type	Count
Tables	1,939
Primary keys	443
Foreign keys	9
Indexes	805
Functions	11
Procedures	7
Triggers	1
Sequences	56
Views	37

Additionally, approximately 50+ database objects (functions and procedures) required complete manual code conversion beyond what Ora2Pg automated. A further 250+ reports were validated for performance parity between Oracle and PostgreSQL.

04. Migration Approach

The migration was carried out using Ora2Pg v24.0, the open-source Oracle-to-PostgreSQL migration tool developed by Gilles Darold and the open-source community. Ora2Pg was selected for its maturity (over 20 years of active development), its comprehensive schema conversion and PL/SQL-to-PL/pgSQL translation capabilities, and its built-in migration cost assessment.

Assessment was performed by Postgres1st and OpenSource DB (OSDB) using Ora2Pg alongside home-grown Python utilities for post-migration validation. The Vupadhi team managed all application-side changes.

Phased Methodology

Phase 1, Assessment & Planning. Ora2Pg SHOW_REPORT and ESTIMATE_COST used to scope effort, identify manual intervention requirements, and classify objects by complexity.

Phase 2, Dual-Write (Oracle + PostgreSQL in Parallel). Dual-write architecture deployed with real-time data synchronisation. System stress-tested at 2x production load. Daily checksum-based reconciliation to validate data parity.

Phase 3, Schema Conversion & PL/pgSQL Rewrite. 1,939 tables, 805 indexes, and all constraints converted via Ora2Pg. 50+ functions and procedures manually rewritten. Application code patched for PostgreSQL compatibility.

Phase 4, Incremental Integration Testing. 5 services at a time pointed to PostgreSQL with dual-write. All 86 services validated against real production endpoints. Issues resolved incrementally per batch.

Phase 5, Cutover (04 April 2026). ~12-hour controlled window. War-room model with Dev, Infra, DB, QC co-located. Final data sync, validation, and go-live within the planned window.

Phase 6, 72-Hour Hypercare. Continuous tracking of transaction throughput, error rates, CPU, memory, disk I/O, and connection pool metrics. Zero critical issues recorded.

05. Architecture Decision, Three Schemas, One Instance

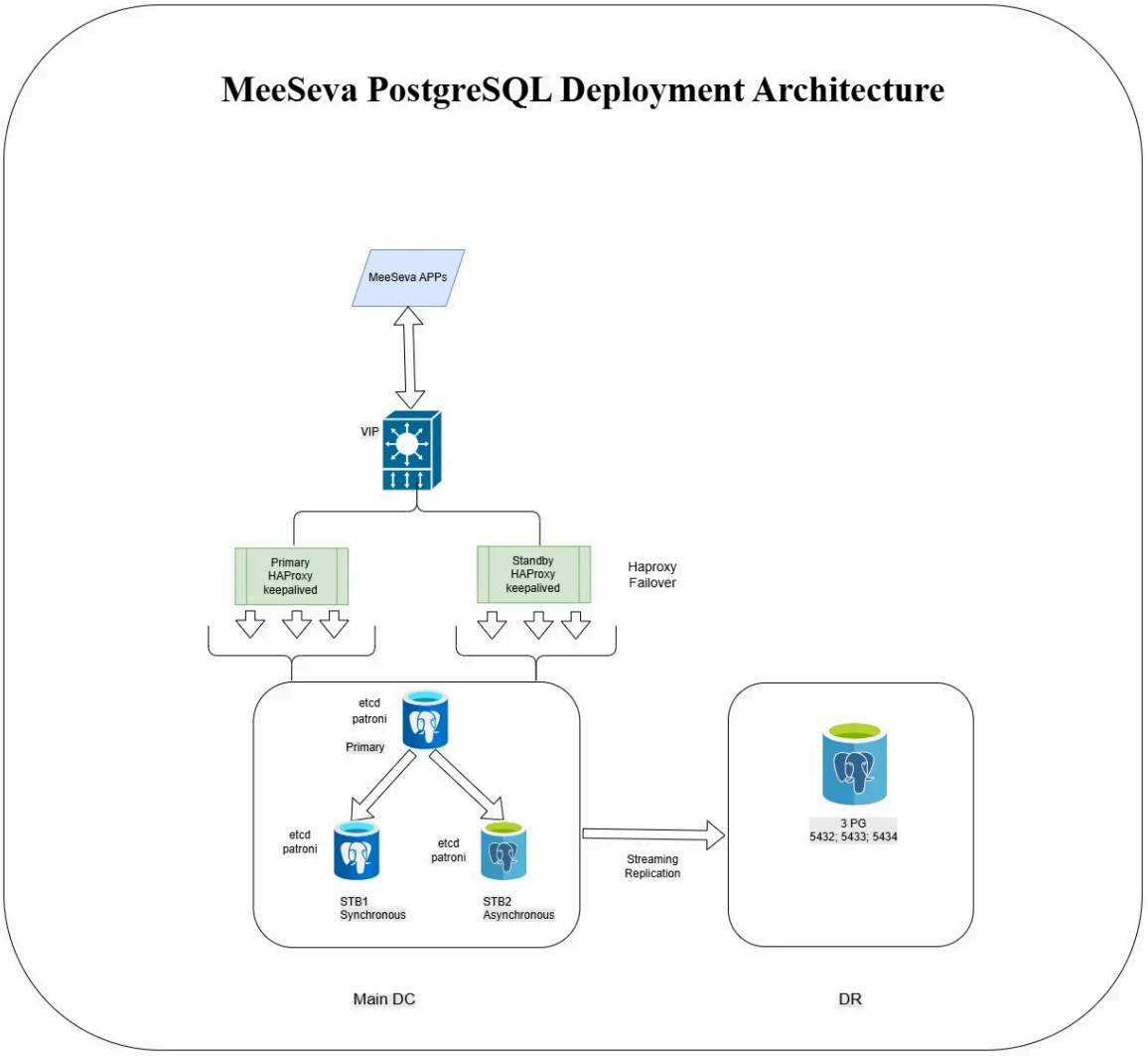
The architecture underwent several iterations before the final design was adopted. The initial proposal called for three separate PostgreSQL clusters, one per database, each in a Primary–Standby–Standby configuration. This would have required 9 nodes with substantial compute resources, exceeding what government procurement could reasonably provision.

A second design reduced each cluster to 2 nodes with DR nodes in different locations, but introduced a performance problem: inter-database SQL execution required PostgreSQL Foreign Data Wrappers (FDW), which proved significantly slower than Oracle's native database links for real-time citizen service transactions.

The Database Audit Team (Postgres1st and OpenSource DB) resolved both problems with a single architectural decision: consolidate all three databases into three schemas within a single PostgreSQL instance, running in an Active–Passive–Passive high-availability configuration. This eliminated FDW overhead entirely (cross-schema queries run at native speed), reduced the infrastructure footprint to a 3-node cluster plus DR, avoided additional application code changes, and ensured more reliable transaction commits.

06. High Availability and Disaster Recovery

The PostgreSQL HA architecture was built using Patroni (cluster management with automatic failover), etcd (distributed consensus), HAProxy (connection routing and load balancing), and Keepalived (Virtual IP management via VRRP). A dedicated DR node receives streaming WAL replication via cascading replication from the async replica.



Component Stack

Component	Version
PostgreSQL	17.7
Operating System	Ubuntu 24.04.3 LTS
Patroni	Latest
HAProxy	3.2.1
Keepalived	2.3.4
etcd	Latest

Cluster Topology

Role	Configuration
Primary Node	Read/Write, handles all production traffic
Sync Replica	Synchronous streaming replication, zero data loss guarantee
Async Replica	Asynchronous replication with configurable delay, feeds DR node via cascading replication
DR Node	Disaster recovery, receives WAL stream from async replica
HAProxy L1 + L2	Dual load balancers with Keepalived VIP failover

Failover Validation Results

All failover scenarios were tested and passed during the HA audit:

Test Scenario	Result	Observed Behaviour
HAProxy failover	✅ PASS	VIP migrated to backup within seconds; preemption on recovery
Keepalived VIP failover	✅ PASS	VRRP failover completed in under 5 seconds
Patroni primary failover	✅ PASS	Sync replica promoted automatically; old primary rejoined as replica
Manual switchover	✅ PASS	Planned switchover completed with WAL continuity
Data replication verification	✅ PASS	Schema changes replicated to all nodes correctly

> *Recovery objectives achieved: RTO < 30 seconds (automatic failover). RPO = Zero (synchronous replication). VIP failover < 5 seconds.*

07. Challenges Faced

Schema Migration Tool Misstep

Early in the project, the application team used Navicat to migrate schema objects without validating datatype mappings. Oracle-to-PostgreSQL conversions are not one-to-one, NUMBER, DATE, and VARCHAR2 all require specific mapping logic. The Database Audit Team identified this risk and recommended Ora2Pg, which applies correct conversion rules automatically. All subsequent schema work was performed using Ora2Pg.

Integration Testing Against Stubs

Integration testing was initially conducted against static stubs rather than live department endpoints, because external government departments do not maintain staging environments. The Database Audit Team proposed a dual-write strategy on production infrastructure, migrating 5 services at a time to PostgreSQL while maintaining Oracle as primary. This gave the team live, production-grade integration testing against real endpoints without risking the entire platform.

FDW Performance Bottleneck

Inter-database SQL queries using PostgreSQL Foreign Data Wrappers were significantly slower than Oracle database links. This was resolved by consolidating three databases into three schemas within a single PostgreSQL instance, eliminating FDW entirely.

Date Format Data Discrepancy

A Category-1 data discrepancy was discovered during pre-production testing: a table was missing rows due to a date format mismatch between Oracle DATE and PostgreSQL timestamp. This was caught by custom Python validation utilities and resolved by patching the application code before go-live.

Performance Tuning

Approximately 20% of queries did not meet SLA targets after initial migration. Advanced indexing strategies, query plan adjustments, and PostgreSQL parameter tuning were applied to match or exceed Oracle performance levels. 250+ reports were benchmarked for execution time parity.

Manual Code Rewrite

Approximately 50+ database objects required complete manual code conversion. Oracle packages, autonomous transactions, and proprietary built-in function dependencies were the primary causes. This was the most labour-intensive component of the migration.

08. Cutover Execution

Cutover date	Friday, 04 April 2026
Window	00:00 AM – ~12:00 PM IST (~12 hours)
Model	War-room, Dev, Infra, DB, QC teams co-located
Monitoring	Real-time: DB performance, connections, latency
Strategy	Pre-migration dual-write validated data parity; controlled downtime for final delta sync and validation
Rollback	Service-level isolation, disable problematic service, resolve, redeploy, re-enable
Outcome	All 85+ services restored within planned window

The cutover was not a leap of faith. By the time the window opened, all 86 services had already been validated against real production endpoints on PostgreSQL through the incremental dual-write testing process.

09. Results

Metric	Result
Critical services migrated	85+
Data integrity	100%, checksum validated
Data loss	Zero
Post-migration critical issues	Zero (72-hr hypercare)
Automatic failover (RTO)	< 30 seconds
Data loss on failover (RPO)	Zero (synchronous replication)
VIP failover time	< 5 seconds
Reports benchmarked	250+

PostgreSQL is running MeeSeva in production. The platform continues to serve citizens across Telangana at the same or improved performance levels as the previous Oracle deployment. Platform metrics, CPU, memory, connection counts, and transaction throughput, have been consistent and stable.

10. Lessons Learned

What Worked Well

- **Ora2Pg's assessment capability was the foundation of planning.** The SHOW_REPORT and ESTIMATE_COST features provided a realistic scope before the first line of code was touched.
- **The dual-write architecture reduced cutover risk to near zero.** Running Oracle and PostgreSQL in parallel with daily checksum reconciliation meant the final cutover was a controlled, validated delta sync.
- **Custom Python validation utilities filled a critical gap.** Ora2Pg does not include post-migration validation. The utilities were essential for sign-off confidence.
- **The three-schema, single-instance architecture solved multiple problems simultaneously.** It eliminated FDW latency, reduced infrastructure from 9 nodes to 3, and simplified operations.
- **The war-room model enabled real-time decision-making.** Co-locating all teams during cutover eliminated communication delays.

What We Would Do Differently

- **Use a purpose-built migration tool from the start.** The initial use of Navicat without datatype validation created rework. Ora2Pg should have been adopted from day one.
- **Enforce a strict application change freeze during migration.** Concurrent changes forced repeated re-validation cycles and extended timelines.
- **Budget 20–30% of total effort for manual PL/SQL rewrite.** Ora2Pg automates 70–80% effectively. The remaining work requires skilled manual effort.
- **Map external integrations early and negotiate staging access.** The absence of staging endpoints forced production-based testing.
- **Set up database-level monitoring and backup policies before go-live.** Storage-level backups are not a substitute for PostgreSQL-native backup and restore capability.

11. Advice for Other Governments and Enterprises

To other state and central government departments, and to private sector organisations running critical platforms on proprietary databases: the MeeSeva migration demonstrates that PostgreSQL is a viable, production-ready alternative for large-scale, mission-critical applications.

PostgreSQL handled a workload of 80,000+ daily transactions across 85+ critical citizen-facing services with zero data loss, zero critical post-migration issues, automatic failover in under 30 seconds, and stable performance metrics. The open-source ecosystem, Ora2Pg for migration, Patroni for high availability, HAProxy for connection routing, pgAdmin for administration, is mature and well-documented.

> *Your first step: Run ora2pg -t SHOW_REPORT --estimate_cost against your Oracle database. It is free, takes minutes, and gives you a realistic scope of the migration effort. Download Ora2Pg at ora2pg.darold.net.*

12. Project Team

Role	Name	Designation	Organisation
Project Champion	Ravikiran Tirumala IFS	Commissioner, MeeSeva ESD	Govt. of Telangana
Project Director	Raja Rao Turaga	Senior Consultant	Govt. of Telangana
Chief Architect	Hari P. Kiran	Founder	OpenSource DB
Migration Leads	Shashidhar Reddy, Tirumala Dasari	Database Architects	OpenSource DB
VMO	Harshad Borker	Regional Director	Postgres1st
Infra Architecture	Sateesh Kumar	Infrastructure SME	Vupadhi
App Architecture	Sagi Srinivas, Ramprasad M	Head of AppDev, App Lead	Vupadhi
Database Leads	Shivasankar Reddy, Phiroji Vannala	Sr. DBAs	Vupadhi

13. Acknowledgements

Gilles Darold & the Ora2Pg Community, For developing and maintaining Ora2Pg as a free, open-source tool since 2001. This migration would not have been feasible at this cost without it.

PostgreSQL Global Development Group, For building and maintaining PostgreSQL, a database engine robust enough to run one of India's largest government platforms.

Postgres1st and OpenSource DB, For architecting and executing the migration.

Vupadhi, For application-side remediation and integration testing.

Open-source tools used: Ora2Pg, PostgreSQL, Patroni, etcd, HAProxy, Keepalived, pgAdmin, Python, and the Perl ecosystem (DBD::Oracle, DBD::Pg).

14. Contact

If you are a government department, IT team, or organisation considering a similar migration and would like to learn from MeeSeva's experience, you may contact:

Ravikiran Tirumala, Commissioner, Electronic Services Delivery (ESD)ITE&C Department, Government of Telangana - dir_eseva@telangana.gov.in

Raja Rao Turaga, Project Director, Electronic Services Delivery (ESD)ITE&C Department, Government of Telangana - srconsultant-esd@telangana.gov.in

Hari P Kiran, Chief Architect, OpenSource DB - harikiran@opensource-db.com

Srinivas Sagi, Head of Application Development, Vupathi - srinivas.sagi@vupadhi.com
Harshad Borkar, Regional Director, Postgres1st - harshad.borkar@postgresfirst.com

15. About Ora2Pg

Ora2Pg is a free, open-source tool for migrating Oracle and MySQL databases to PostgreSQL. Developed by Gilles Darold since 2001 and released under the GPL v3 licence, it is used by organisations worldwide for migrations ranging from simple schema ports to complex, multi-terabyte enterprise database transitions.

Source code: github.com/darold/ora2pg

Documentation: ora2pg.darold.net/documentation.html

This case study was prepared by Postgres1st and OpenSource DB and reviewed by MeeSeva: Electronic Services Delivery (ESD) department, ITE&C, Government of Telangana, India. Migration executed April 2026.