AI-Assisted Enterprise Database Migration & Modernization

---

1. Enterprise Problem Statement (Real-World Context)

Enterprises often operate **mission-critical applications on legacy databases** (Oracle, SQL Server, DB2, MySQL) that have evolved over years with:

- Tight coupling between application code and database logic
- Heavy usage of stored procedures, triggers, and vendor-specific SQL
- Poor or outdated documentation
- Large data volumes (TB–PB scale)
- Strict uptime, compliance, and performance SLAs

Typical migration goals include:

- On-prem → Cloud (Oracle → Aurora / PostgreSQL, SQL Server → Azure SQL)
- Commercial → Open-source (Oracle → PostgreSQL)
- Legacy schema redesign for microservices

Current approaches rely heavily on **manual analysis, partial automation, and risky big-bang cutovers**, leading to:

- Long timelines (6–18 months)
- High cost and human error
- Unexpected performance regressions
- Data integrity issues

---

2. Project Overview

This project aims to build an **AI-agent–driven database migration solution** that automates **analysis, planning, transformation, validation, and explainability** of database migrations using **LangGraph / LangChain**.

The solution will:

- Analyze source DB schema, data, and logic
- Generate a safe, explainable migration strategy
- Automatically transform schemas, queries, and stored logic
- Validate correctness, performance, and data integrity
- Support phased and zero-downtime migration patterns

The PoC will be validated using **open-source databases and schemas** that mimic real enterprise complexity.

---

3. Objectives

Primary Objectives

- Automate database migration with minimal human intervention
- Preserve **data correctness, performance, and transactional integrity**
- Reduce migration risk and time-to-value
- Provide full auditability and explainability of changes

Secondary Objectives

- Support heterogeneous DB migrations
- Enable phased / hybrid migration strategies
- Create reusable AI agents for future migrations

---

4. Scope

In-Scope

- Schema migration (tables, indexes, constraints)
- Data type mapping and transformation
- Stored procedures, functions, triggers migration
- SQL query compatibility refactoring
- Data migration and validation
- Performance benchmarking
- Migration reports and explainability

Out-of-Scope (PoC Phase)

- Application code refactoring (limited to SQL compatibility hints)
- Production cutover automation
- Proprietary vendor tools integration (later phase)

---

5. Target Users

- Database Architects
- Platform & Cloud Migration Teams
- Data Engineering Teams
- SRE / DevOps Teams
- Enterprise Architects

---

6. Supported Migration Scenarios (PoC)

Source	Target
MySQL	PostgreSQL
PostgreSQL	Cloud-managed PostgreSQL
Oracle-like schema	PostgreSQL
SQL Server-like schema	PostgreSQL / Azure SQL

---

7. High-Level Agent Architecture

LangGraph-based Stateful Agent Flow

```
Source DB Introspection Agent
        ↓
Dependency & Impact Analysis Agent
        ↓
Migration Strategy Planner Agent
        ↓
Schema Transformation Agent
        ↓
Stored Logic Conversion Agent
        ↓
Data Migration Agent
        ↓
Validation & Reconciliation Agent
        ↓
Performance Benchmarking Agent
        ↓
Explainability & Reporting Agent
```

---

8. Functional Requirements

8.1 Source Database Introspection

- Connect to source DB (read-only)
- Extract:

    - Schemas, tables, views
    - Indexes, constraints
    - Stored procedures, functions, triggers
    - Row counts, data volumes
- Build dependency graph across objects

8.2 Dependency & Impact Analysis

- Identify:
    - Cross-schema dependencies
    - Circular references
    - Vendor-specific SQL features
- Classify migration complexity:
    - Low / Medium / High risk objects
- Detect anti-patterns (e.g., heavy trigger logic)

8.3 Migration Strategy Planning

- Decide migration approach:
    - Big-bang
    - Phased (schema → data → logic)
    - Dual-write / sync-based (PoC simulation)
- Define:
    - Object migration order
    - Rollback checkpoints
    - Validation gates
- Generate a **human-reviewable migration plan**

8.4 Schema Transformation

- Convert:
    - Data types (e.g., NUMBER → NUMERIC)
    - Index types
    - Constraints and defaults
- Normalize vendor-specific features
- Apply naming and convention standards
- Generate target-DB-specific DDL

8.5 Stored Logic Conversion

- Translate:
    - Stored procedures

    - Functions
    - Triggers
- Replace unsupported constructs with:
    - Equivalent SQL
    - Application-side logic recommendations
- Flag manual review items clearly

8.6 Data Migration

- Support:
    - Full load
    - Incremental load (PoC simulation)
- Handle:
    - Large tables
    - Referential integrity
    - Data type transformation
- Provide resumable migration support

8.7 Validation & Reconciliation

- Row count comparison
- Checksum / hash-based data validation
- Constraint validation
- Referential integrity checks
- Transaction consistency verification

8.8 Performance Benchmarking

- Capture baseline metrics:
    - Query execution time
    - Index usage
    - Storage footprint
- Compare post-migration metrics
- Flag regressions beyond defined thresholds

8.9 Explainability & Reporting

- Generate reports:
    - Objects migrated
    - Transformation rationale
    - Known risks & limitations
- Provide:

    - SQL diffs
    - Mapping tables (source → target)
- Export:
    - PDF / Markdown / JSON formats

---

9. Non-Functional Requirements

9.1 Reliability

- Idempotent migrations
- Safe rollback capability
- Consistent results across runs

9.2 Performance

- PoC scale:
    - 10–50 GB datasets
- No more than ±5–10% performance degradation

9.3 Security

- Read-only source access
- Masking support for sensitive columns
- No credential persistence

9.4 Compliance & Auditability

- Full migration logs
- Change traceability per DB object
- Explainable AI decisions

---

10. AI / Agent Requirements

10.1 Framework

- LangGraph for stateful workflows
- LangChain for:
    - SQL understanding
    - Code transformation
    - Tool orchestration

10.2 Agent Capabilities

- SQL AST analysis
- Cross-DB dialect reasoning
- Stateful retries and self-healing

- Tool calling for:
    - DB clients
    - Benchmark execution
    - Validation scripts

---

11. PoC Validation Strategy

11.1 Test Data

- Open-source schemas:
    - Sakila
    - AdventureWorks
    - TPC-H (scaled)
- Introduce:
    - Triggers
    - Stored procedures
    - Vendor-specific SQL constructs

11.2 Success Criteria

- 100% schema migration success
- ≥ 99.99% data integrity match
- Acceptable performance variance
- Clear migration documentation

---

12. Deliverables

- AI-driven migration PoC
- LangGraph workflow definitions
- Sample migration reports
- Demo walkthrough
- Technical documentation

---

13. Risks & Mitigations

Risk	Mitigation
Complex stored logic	Flag + partial automation
Performance regressions	Automated benchmarking
Data loss	Multi-layer validation

SQL dialect gaps Agent-driven refactoring

Database Migration / Schema Tools Repositories

1. **SQL Server → PostgreSQL Migration Tool**
   Converts SQL Server schema to PostgreSQL and can generate data migration jobs — great for heterogeneous DB migration PoC.
   https://github.com/dalibo/sqlserver2pgsql GitHub
2. **Atlas (Schema as Code & Migration Tool)**
   Declarative schema management and migration tool useful for automated DB schema diff + migration logic.
   https://github.com/ariga/atlas GitHub
3. **postgres_migrator (PostgreSQL Migrations CLI)**
   Generates and applies schema migrations from declarative SQL — useful for migration tooling and workflow tests.
   https://github.com/blainehansen/postgres_migrator GitHub