Smart City Data Warehouse - Project Documentation

Table of Contents

---

1. Project Planning

1.1 Executive Summary

The Smart City Data Warehouse is a data engineering project demonstrating ETL pipeline development using three different technologies (SQL, Python, Talend) with Apache Airflow orchestration. The project implements a Medallion Architecture (Bronze-Silver-Gold) to process data from various smart city services.

1.2 Project Objectives

- **Centralized Data Repository**: Build a unified data warehouse for smart city data
- **Multi-Technology Implementation**: Demonstrate SQL, Python, and Talend ETL approaches
- **Medallion Architecture**: Implement Bronze → Silver → Gold data layers
- **Historical Tracking**: Use SCD Type 2 for building dimension
- **Workflow Orchestration**: Automate ETL jobs with Apache Airflow

1.3 Project Scope

In Scope

- **Data Domains**:
  - Building management
  - Public transportation (buses)
  - Energy consumption
  - Emergency services
  - Traffic monitoring
  - Waste collection
- **Technical Components**:
  - SQL Server database with Bronze/Silver/Gold schemas
  - T-SQL stored procedures
  - Python ETL scripts with Pandas
  - Talend visual ETL jobs
  - Apache Airflow DAGs
  - CSV data generation scripts

Out of Scope

- Real-time streaming
- Cloud deployment
- Production UI/dashboard implementation
- Mobile applications
- Advanced security features

1.4 Architecture

The project uses **Medallion Architecture** with three layers:

1. **Bronze Layer**: Raw data from CSV files
2. **Silver Layer**: Cleaned and validated data
3. **Gold Layer**: Analytics-ready dimensional model (Galaxy Schema)

1.5 Technology Stack

- **Database**: Microsoft SQL Server
- **Languages**: Python 3.8+, T-SQL
- **Libraries**: Pandas, SQLAlchemy, pyodbc
- **ETL Tools**: Talend Open Studio
- **Orchestration**: Apache Airflow 2.0+
- **Data Format**: CSV files

1.6 Deliverables

Technical Deliverables:

1. SQL Server database (Bronze, Silver, Gold schemas)
2. SQL ETL stored procedures
3. Python ETL pipeline
4. Talend ETL jobs
5. Airflow DAG
6. Data generation scripts
7. Sample CSV datasets

Documentation Deliverables:

1. System architecture
2. ETL implementation guides
3. Data dictionary
4. This project documentation

---

2. Stakeholder Analysis

2.1 Primary Stakeholders

2.1.1 City Management

Needs: City-wide performance metrics, trend analysis, budget insights
Data Usage: Executive reports, KPI dashboards
Key Metrics: Service efficiency, response times, resource utilization

2.1.2 Department Managers

Transportation Department:

- Bus route performance
- Traffic flow patterns
- GPS tracking data

Public Safety Department:

- Emergency call analytics
- Response time metrics
- Incident patterns by zone

Environmental Services:

- Waste collection efficiency
- Truck utilization
- Container fill levels

Energy & Utilities:

- Building energy consumption
- Device performance
- Demand patterns

2.1.3 Data Team

Needs: Clean data, documentation, performance
Requirements:

- Comprehensive data dictionary
- Optimized queries
- ETL monitoring
- Data quality validation

2.2 Stakeholder Requirements

Requirement	Priority	Implementation
Historical tracking	High	SCD Type 2 for buildings
Data quality	High	Silver layer validation
Multiple ETL options	Medium	SQL, Python, Talend
Automated workflows	High	Airflow scheduling

Requirement	Priority	Implementation
Documentation	High	Markdown docs

3. Database Design

3.1 Architecture Overview

Medallion Architecture with Galaxy Schema in Gold layer:

- **Bronze**: Raw CSV data with minimal transformation
- **Silver**: Validated, cleaned, standardized data
- **Gold**: Dimensional model optimized for analytics

3.2 Galaxy Schema (Gold Layer)

Dimension Tables (7)

1. **dim_calendar**: Date dimension for temporal analysis
    - Attributes: date, day, month, year, week_day, week_number, is_weekend
2. **dim_buildings** (SCD Type 2): Building information with history
    - Attributes: building_id, name, zone_id, type, owner, address, lat, lon, status
    - Historical fields: valid_from, valid_to, is_current
3. **dim_devices**: IoT device catalog
    - Attributes: device_id, device_type, building_id, install_date, status, manufacturer
4. **dim_trucks**: Waste collection fleet
    - Attributes: truck_id, truck_type, capacity_tons, fuel_type, status
5. **dim_zones**: Geographic zones
    - Attributes: zone_id, zone_name, lat_min/max, lon_min/max, description
6. **dim_bus_routes**: Transit routes
    - Attributes: route_id, route_name, start_point, end_point, distance_km, active_status
7. **dim_event_types**: Emergency event classification
    - Attributes: event_type_id, event_type_name, description

Fact Tables (5)

1. **fact_bus_gps**: Bus location tracking
    - Measures: speed_kmh, occupancy_est, lat, lon
    - Dimensions: date, route, zone, bus_id (degenerate)

2. **fact_emergency_calls**: Emergency incidents
    - Measures: response_time_minutes
    - Dimensions: date, zone, building, event_type, call_id, priority_level
3. **fact_energy_consumption**: Building energy usage
    - Measures: kwh, voltage, current
    - Dimensions: date, building, device, quality_flag
4. **fact_traffic**: Traffic monitoring
    - Measures: vehicle_count, avg_speed_kmh
    - Dimensions: date, zone, device
5. **fact_waste_collection**: Waste operations
    - Measures: fill_level_percent
    - Dimensions: date, zone, building, truck, container_id

3.3 Key Design Patterns

Surrogate Keys: INT identity columns (e.g., building_sk, date_sk)

Natural Keys: Business identifiers (e.g., building_id, device_id)

Degenerate Dimensions: High-cardinality attributes stored in facts (call_id, container_id)

Conformed Dimensions: Shared across facts (calendar, zones, buildings, devices)

SCD Type 2: Tracks building history with valid_from, valid_to, is_current flags

3.4 Schema Relationships

```
dim_calendar (shared by all facts)
    ↓
fact_bus_gps ← dim_bus_routes, dim_zones
fact_emergency_calls ← dim_zones, dim_buildings, dim_event_types
fact_energy_consumption ← dim_buildings, dim_devices
fact_traffic ← dim_zones, dim_devices
fact_waste_collection ← dim_zones, dim_buildings, dim_trucks
```

3.5 Data Flow

```
CSV Files (datasets/silver_data/)
    ↓
Bronze Layer (raw ingestion)
    ↓
Silver Layer (cleaning, validation)
    ↓
```

```
Gold Layer (dimensional model)
    ↓
Analytics & Reports
```

3.6 Naming Conventions

Tables: `{schema}.{type}_{name}`

- Examples: `gold.dim_buildings`, `gold.fact_energy_consumption`

Columns:

- Surrogate Keys: `{entity}_sk`
- Natural Keys: `{entity}_id`
- Foreign Keys: Match referenced dimension SK

Stored Procedures: `proc_load_{layer}`

- Examples: `bronze.proc_load_bronze`, `gold.proc_load_gold`

---

4. UI/UX Design

4.1 Design Philosophy

The UI/UX design focuses on **data visualization and accessibility** for different user roles. Design mockups demonstrate dashboard concepts for executive, operational, and analytical use cases.

4.2 Design Resources

Figma Board: Smart City Project on Figma

Canva Design: Smart City Project on Canva

4.3 Dashboard Concepts

The design prototypes include concepts for:

Executive Dashboard

- City-wide KPI cards
- Department performance comparison
- Trend charts
- Geographic heat maps

Transportation Dashboard

- Interactive map with bus locations
- Route performance metrics
- Time-series analysis
- Route comparison tables

Emergency Services Dashboard

- Alert feed
- Response time analytics
- Event type distribution
- Zone-based heat maps

Energy Dashboard

- Building consumption rankings
- Consumption trends
- Device performance monitoring
- Building detail views

4.4 Design System

Color Palette:

- Primary: #2C3E50 (Navy Blue)
- Secondary: #3498DB (Sky Blue)
- Success: #27AE60 (Green)
- Warning: #F39C12 (Orange)
- Danger: #E74C3C (Red)

Typography:

- Headings: Inter (Sans-serif)
- Data/Numbers: Roboto Mono (Monospace)

Components:

- KPI cards
- Interactive charts (Chart.js/D3.js)
- Data tables with sorting
- Maps with zone overlays
- Filters and date pickers

4.5 Key Features

- **Role-based views**: Different layouts for executives vs. analysts
- **Interactive visualizations**: Drill-down capabilities
- **Responsive design**: Desktop and tablet support
- **Export functionality**: PDF and CSV downloads
- **Real-time updates**: Configurable refresh rates

4.6 Accessibility

- Color contrast compliance
- Keyboard navigation
- Screen reader support

- Clear visual hierarchy

---

5. Implementation Status

5.1 Completed

- [x] Database schema (Bronze, Silver, Gold)
- [x] SQL ETL stored procedures
- [x] Python ETL pipeline
- [x] Talend ETL jobs
- [x] Airflow orchestration
- [x] Data generation scripts
- [x] Sample datasets
- [x] Technical documentation
- [x] UI/UX design mockups

5.2 Future Enhancements

- Dashboard implementation
- Real-time data streaming
- Advanced analytics
- Cloud deployment
- Enhanced security
- Performance optimization at scale

---

6. Getting Started

For Database Developers

1. Review Database Design
2. Check Meta Data Definition
3. Follow SQL ETL Documentation

For Python Developers

1. Review Architecture
2. Follow Python ETL Documentation
3. Check requirements.txt for dependencies

For ETL Developers

1. Review Talend Documentation
2. Check data flow diagrams in docs folder
3. Review Airflow Orchestration

For Business Users

1. Review Stakeholder Analysis

2. Check UI/UX design mockups on Figma/Canva
3. Understand available data domains

---

7. Related Documentation

For detailed technical implementation guides, please refer to the documentation files in the docs directory:

- **System Architecture** - Technical architecture overview
- **Meta Data Definition** - Complete data catalog with table/column specifications
- **SQL ETL Documentation** - T-SQL stored procedures implementation
- **Python ETL Documentation** - Python/Pandas pipeline implementation
- **Talend Documentation** - Visual ETL jobs guide
- **Airflow Orchestration** - Workflow automation and scheduling

---

Document Version: 1.0
Last Updated: November 1, 2025
Authors: Smart City Data Engineering Team
Status: Complete