TimescaleDB Database Structure Documentation

HES (Head End System) Database Architecture

Table of Contents

1. Overview
2. Core Components
3. Table Relationships
4. Schema Organization
5. Time-Series Features
6. Common Use Cases

1. Overview

This document describes the database structure of the Head End System (HES) implemented using TimescaleDB. The system is designed to manage meter data, measurements, events, and push notifications efficiently.

2. Core Components

2.1 Meter Management

- **meters** (Primary table)
    - Primary Key: id (UUID)
    - Key fields: meter_number, meter_type
    - Timestamps: created_at, updated_at
    - Central table referenced by most other tables

2.2 Measurement Tables

All these tables reference the meters table:

- **meter_readings**: Basic meter readings
- **energy_measurements**: Energy consumption data
- **power_readings**: Power-related measurements
- **voltage_measurements**: Voltage readings
- **current_measurements**: Current readings

2.3 Profile Tables

Each profile table maintains specific aspects of meter data:

- **instant_profile**: Real-time meter profiles
- **daily_load_profile**: Daily consumption patterns
- **block_load_profile**: Block-wise load data
- **billing_profile**: Billing-related profiles

2.4 Event Management

- **meter_events**: Stores meter-specific events/alerts
- **event_logs**: System-wide event logging

2.5 Push Notification System

- **push_schedules**: Defines notification timing
- **schedule_entries**: Detailed schedule items
- **push_history**: Notification history
- **push_objects**: Data selection for pushing
- **push_setups**: Push configuration

2.6 Reference Data

- **obis_codes**: Standard OBIS (Object Identification System) codes
    - Used for standardized meter reading identification
    - Contains code, description, and unit information

3. Table Relationships

3.1 Meter-Centric Relationships

```
meters (1) ──┬──(∞) meter_readings
             ├──(∞) meter_events
             ├──(∞) energy_measurements
             ├──(∞) power_readings
             ├──(∞) voltage_measurements
             ├──(∞) current_measurements
             ├──(∞) instant_profile
             ├──(∞) daily_load_profile
             ├──(∞) block_load_profile
             └──(∞) billing_profile
```

3.2 Push Notification Relationships

```
push_schedules (1)──(∞) schedule_entries
      │
      └──(∞) push_history
              ▲
push_objects──┤
              │
push_setups───┘
```

4. Schema Organization

4.1 Public Schema

- Contains core application tables:
    - meters
    - meter_readings
    - meter_events

4.2 _timescaledb_cache Schema

- Contains application data tables and views
- Houses most operational tables

4.3 System Schemas

- **_timescaledb_catalog**: Internal system tables
- **_timescaledb_config**: Configuration tables
- **_timescaledb_internal**: Management tables
- **timescaledb_information**: Metadata views

5. Time-Series Features

5.1 Hypertables

The following tables are automatically partitioned by time:

- meter_readings
- energy_measurements
- power_readings
- voltage_measurements
- current_measurements

Benefits:

- Improved query performance
- Efficient data retention management
- Automatic partitioning

5.2 Continuous Aggregates

Used for:

- daily_load_profile
- Automatic maintenance of aggregated views
- Efficient historical data analysis

6. Common Use Cases

6.1 Meter Reading Management

- Recording periodic meter readings
- Tracking measurement history
- Generating load profiles

6.2 Event Handling

- Recording meter events
- System event logging
- Alert generation

6.3 Data Push Operations

- Scheduled data pushing
- Push history tracking
- Configuration management

Note: This document provides a high-level overview of the database structure. For detailed implementation specifics, please refer to the system documentation or contact the development team.