Table of Contents

# **CBA NPW-DBT Project - Comprehensive Implementation Guide**

## **📋 Project Overview**

**Project Name:** np\_projects\_commbank\_sf\_dbt  
**Purpose:** Commonwealth Bank Australia (CBA) data transformation and ETL processing system for migration to Snowflake  
**Scale:** 80+ SQL models, legacy Teradata-to-Snowflake migration project  
**Architecture:** Multi-layered data pipeline with process orchestration, transformation, and loading capabilities

**Project Structure:**

NPW-DBT/  
├── NPW DBT - My Changes/ # ← Current folder (your customizations)  
├── NPW DBT Project/ # ← Original project structure   
├── translated/ # ← Legacy translated files (not actively used)  
├── models/ # ← Active dbt models  
├── macros/ # ← Custom dbt macros  
├── seeds/ # ← Configuration data  
└── dbt\_project.yml # ← Main project configuration

## **🏗️ Project Architecture**

### **Core Business Domains:**

1. **📱 Application Products** (appt\_pdct) - Customer application and product management
2. **🔄 CSE Data Load** (cse\_dataload) - Customer Service Environment data processing
3. **📊 Process Orchestration** - Stream status tracking and metadata management

### **Data Processing Layers:**

#### **1. Process Management (02processkey)**

* **Process Key Generation** - Unique identifier management for ETL processes
* **Utility Process ISAC** - Source-to-target mapping and conversion
* **Key Models:** proskeyhash\_\_loadgdwproskeyseq, util\_pros\_isac\_\_loadgdwproskeyseq

#### **2. Mapping & Lookups (04MappingLookupSets)**

* **Reference Data Management** - Product mapping and lookup tables
* **Data Standardization** - Code mapping and transformation rules
* **Key Models:** ldmap\_cse\_pack\_pdct\_pllkp

#### **3. Data Extraction (08extraction)**

* **Source System Integration** - CPL Business App data extraction
* **Data Ingestion** - Raw data capture from legacy systems
* **Key Models:** srcplappseq\_\_extpl\_app

#### **4. Mapping Transformation (12MappingTransformation)**

* **Business Rule Application** - Complex transformation logic
* **Data Quality Validation** - Rejection handling and error processing
* **Key Models:** tmpapptpdctds\_\_xfmpl\_appfrmext, tgtapptpdctrejectsds\_\_xfmpl\_appfrmext

#### **5. Temporary Loading (14loadtotemp)**

* **Staging Area Management** - Temporary table processing
* **Data Preparation** - Pre-transformation staging

#### **6. Transform Delta (16transformdelta)**

* **Change Data Capture** - Delta processing and change tracking
* **Historical Data Management** - SCD (Slowly Changing Dimension) processing
* **Key Models:** dltappt\_deptfrmtmp\_appt\_dept

#### **7. Load to GDW (18loadtogdw)**

* **Final Data Loading** - Target system data insertion
* **Data Validation** - Post-load quality checks
* **Key Models:** ldapptpdctupd

#### **8. Process Metadata (24processmetadata)**

* **ETL Orchestration** - Process status tracking
* **Audit Trail** - Run stream monitoring and logging
* **Key Models:** tgt\_step\_stus\_tbl\_\_processrunstreamstepoccrbeginandend

## **💾 Database Configuration**

### **Target Environment:**

Database: GDW1\_IBRG # Final production tables  
Staging: GDW1\_STG # Intermediate views and datasets   
Control: GDW1 # Control tables and procedures

### **Key Schemas:**

* **cse4\_ctl** - Control and metadata tables
* **files** - File-based datasets
* **datasets** - Processed data staging
* **stage\_views** - Default dbt schema

### **Current Connection Settings:**

Account: next\_pathway\_partner.us-east-1  
User: npcommbank01  
Role: ROLE\_NPCOMMBANK01  
Database: commbankdb  
Warehouse: DBT  
Schema: stage\_views

## **🚀 Import to Snowflake Native dbt**

### **Step 1: Prerequisites**

-- Create required databases and schemas  
CREATE DATABASE IF NOT EXISTS GDW1\_IBRG;  
CREATE DATABASE IF NOT EXISTS GDW1\_STG;   
CREATE DATABASE IF NOT EXISTS GDW1;  
  
-- Create schemas  
CREATE SCHEMA IF NOT EXISTS GDW1.cse4\_ctl;  
CREATE SCHEMA IF NOT EXISTS GDW1\_STG.files;  
CREATE SCHEMA IF NOT EXISTS GDW1\_STG.datasets;  
CREATE SCHEMA IF NOT EXISTS GDW1\_STG.stage\_views;  
  
-- Create warehouse  
CREATE WAREHOUSE IF NOT EXISTS DBT   
 WITH WAREHOUSE\_SIZE = 'MEDIUM'   
 AUTO\_SUSPEND = 300;

### **Step 2: Setup Snowflake Native dbt**

1. **Navigate to Snowflake Console** → Projects → dbt
2. **Create New Project** → Import from Git
3. **Repository URL:** https://github.com/sfc-gh-mhindi/REPO\_CBA.git
4. **Subdirectory:** GDW1/NPW-DBT

### **Step 3: Environment Configuration**

# Update profiles.yml in Snowflake dbt  
name: np\_projects\_commbank\_sf\_dbt  
version: '1.0.0'  
  
profile: 'snowflake\_profile'  
outputs:  
 dev:  
 type: snowflake  
 account: [YOUR\_ACCOUNT]  
 user: [YOUR\_USER]  
 role: [YOUR\_ROLE]  
 database: GDW1\_STG  
 warehouse: DBT  
 schema: stage\_views  
 threads: 4

### **Step 4: Required Variables Configuration**

# Add to dbt\_project.yml vars section  
vars:  
 files\_schema: files  
 datasets\_schema: datasets  
 mart\_db: GDW1\_IBRG  
 intermediate\_db: GDW1\_STG   
 stg\_ctl\_db: GDW1  
 etl\_process\_dt\_tbl: YYYYMMDD  
   
 # Job-specific parameters (update as needed)  
 base\_dir: cba\_app\_\_csel4\_\_prod  
 run\_stream: CSE\_COM\_BUS\_APP\_PROD  
 etl\_process\_dt: 20240101  
 app\_release: CSEL4  
 gdw\_acct\_db: STAR\_CAD\_PROD\_DATA  
   
 # Environment-specific settings  
 envs:  
 env\_apt\_startup\_status: 'True'  
 env\_apt\_show\_component\_calls: 'True'  
 env\_ds\_pxdebug: '0'  
 env\_apt\_disable\_combination: 'True'  
 env\_apt\_no\_sort\_insertion: 'True'  
 env\_cc\_tera\_truncate\_string\_with\_null: '1'  
 env\_apt\_string\_allpads\_not\_empty: 'True'

## **🎯 Model Execution Strategy**

### **Phase 1: Foundation Setup**

# 1. Seed configuration data  
dbt seed  
  
# 2. Install dependencies   
dbt deps

### **Phase 2: Core Infrastructure**

# 3. Build process key models  
dbt run --select models/cse\_dataload/02processkey  
  
# 4. Build mapping and lookup sets  
dbt run --select models/cse\_dataload/04MappingLookupSets

### **Phase 3: Data Processing Pipeline**

# 5. Extract source data  
dbt run --select models/cse\_dataload/08extraction  
  
# 6. Apply transformations  
dbt run --select models/cse\_dataload/12MappingTransformation  
  
# 7. Load to temporary staging  
dbt run --select models/cse\_dataload/14loadtotemp

### **Phase 4: Delta Processing & Final Load**

# 8. Process delta changes  
dbt run --select models/cse\_dataload/16transformdelta  
  
# 9. Application product processing  
dbt run --select models/appt\_pdct  
  
# 10. Load to target GDW  
dbt run --select models/cse\_dataload/18loadtogdw

### **Phase 5: Process Management**

# 11. Update process metadata  
dbt run --select models/cse\_dataload/24processmetadata  
  
# 12. Run data quality tests  
dbt test

### **Complete Pipeline Execution**

# Run entire pipeline in dependency order  
dbt run  
  
# Full pipeline with tests  
dbt build

## **📊 Key Models & Dependencies**

### **Critical Models:**

#### **Process Management:**

* **proskeyhash\_\_loadgdwproskeyseq** - Process key management and hashing
* **util\_pros\_isac\_\_loadgdwproskeyseq** - Utility process ISAC mapping

#### **Application Products:**

* **tgtapptpdctinsertds\_\_ldapptpdctins** - Application product inserts
* **tgtapptpdctinstera\_\_ldapptpdctins** - Application product Teradata integration
* **gtapptpdcttera\_\_ldtmp\_appt\_pdctfrmxfm** - Temporary application product transformation

#### **Process Orchestration:**

* **tgt\_step\_stus\_tbl\_\_processrunstreamstepoccrbeginandend** - Process step status tracking
* **xfm\_step\_status\_\_processrunstreamstepoccrbeginandend** - Step status transformation
* **lkp\_step\_occr\_\_processrunstreamstepoccrbeginandend** - Step occurrence lookup

#### **Delta Processing:**

* **srctmpapptpdcttera\_\_dltappt\_pdctfrmtmp\_appt\_pdct** - Source temporary application product
* **tgtapptpdctinsertds\_\_dltappt\_pdctfrmtmp\_appt\_pdct** - Target application product inserts

### **Execution Dependencies:**

Seeds → Process Keys → Lookups → Extraction →   
Transformation → Temp Loading → Delta Processing →   
Final Loading → Metadata Updates

## **🛠️ Custom Macros**

The project includes several custom macros for enhanced functionality:

### **Stream Management:**

* **check\_stream\_status.sql** - Monitors ETL stream status
* **run\_stream\_check.sql** - Validates stream execution

### **Configuration Management:**

* **load\_job\_params.sql** - Loads job-specific parameters
* **get\_custom\_schema.sql** - Dynamic schema name generation

## **📁 Configuration Data (Seeds)**

### **Project Configuration (project\_config.csv):**

context,parameter\_name,parameter\_value  
CSEL4,GDW\_SERVER,teradata.gdw.cba  
CSEL4,GDW\_ACCT\_DB,STAR\_CAD\_PROD\_DATA  
CSEL4,BASE\_DIR,cba\_app\_\_csel4\_\_prod  
CSEL4,CTL\_SCHEMA,CSE4\_CTL  
CSEL4,APP\_RELEASE,CSEL4

**Key Parameters:** - **GDW\_SERVER** - Source Teradata server - **GDW\_ACCT\_DB** - Account database reference - **BASE\_DIR** - Application base directory - **CTL\_SCHEMA** - Control schema for metadata - **APP\_RELEASE** - Application release identifier

## **⚠️ Important Considerations**

### **Legacy System Integration:**

* **Teradata Origins** - Models contain legacy Teradata-specific logic
* **Source Systems** - References to external systems requiring connectivity
* **Date Parameters** - ETL process dates need proper configuration

### **Data Quality:**

* **Rejection Handling** - Built-in data quality and error processing
* **Audit Trails** - Comprehensive logging and process tracking
* **Post-hooks** - Models use post-hooks for target table loading

### **Performance Optimization:**

* **Warehouse Sizing** - May need larger warehouses for full production loads
* **Clustering** - Consider clustering on process dates and keys
* **Incremental Models** - Some models may benefit from incremental processing

### **File Organization Notes:**

* **translated/ folder** - Contains legacy translated files that are NOT actively used by dbt
* **Active models only** - dbt only processes files in the models/ directory
* **Reference purposes** - translated folder may serve as reference for business logic

## **🔧 Troubleshooting**

### **Common Issues:**

#### **1. Source Table Missing**

# Error: Relation 'database.schema.table' does not exist  
# Solution: Update source references in models

#### **2. Permission Errors**

# Error: Insufficient privileges to operate on database  
# Solution: Ensure proper database/schema access for your role

#### **3. Variable Substitution**

# Error: Unknown variable 'var\_name'  
# Solution: Verify all cvar() variables are defined in dbt\_project.yml

#### **4. Post-hook Failures**

# Error: Post-hook failed during model execution  
# Solution: Check target table structures and permissions

### **Monitoring:**

-- Monitor dbt run progress  
SELECT \* FROM GDW1.cse4\_ctl.STEP\_OCCR   
WHERE RUN\_STRM\_C = 'CSE\_COM\_BUS\_APP\_PROD'  
ORDER BY STEP\_OCCR\_STRT\_S DESC;  
  
-- Check process key generation  
SELECT \* FROM GDW1\_STG.files.PROCESSKEYHASH\_\_HSH  
ORDER BY PROS\_KEY\_I DESC  
LIMIT 10;  
  
-- Monitor data quality rejections  
SELECT \* FROM GDW1\_STG.datasets.\*\_rejects\_\*  
WHERE ETL\_PROCESS\_DT = '20240101';

## **📈 Production Deployment**

### **Environment Promotion:**

1. **Development** → Test configuration and model logic
2. **Staging** → Validate with production-like data volumes
3. **Production** → Full deployment with monitoring

### **Performance Considerations:**

-- Recommended warehouse sizes by phase  
-- Phase 1-2 (Setup): SMALL  
-- Phase 3-4 (Processing): MEDIUM to LARGE  
-- Phase 5 (Metadata): SMALL  
  
-- Clustering recommendations  
ALTER TABLE GDW1\_IBRG.schema.table   
CLUSTER BY (ETL\_PROCESS\_DT, PROS\_KEY\_I);

### **Monitoring & Alerts:**

-- Set up alerts for failed runs  
CREATE OR REPLACE ALERT dbt\_failure\_alert  
WAREHOUSE = DBT  
SCHEDULE = '5 MINUTE'  
IF (  
 SELECT COUNT(\*) FROM GDW1.cse4\_ctl.STEP\_OCCR   
 WHERE STEP\_STUS\_C = 'F'   
 AND STEP\_OCCR\_STRT\_S > DATEADD(MINUTE, -10, CURRENT\_TIMESTAMP)  
) > 0  
THEN CALL alert\_notification\_procedure();

## **📚 Additional Resources**

### **dbt Documentation:**

* [dbt Official Documentation](https://docs.getdbt.com/)
* [Snowflake dbt Integration](https://docs.snowflake.com/en/user-guide/ui-snowsight-projects-dbt)
* [dbt Best Practices](https://docs.getdbt.com/guides/best-practices)

### **CBA-Specific Resources:**

* **Project Repository:** [GitHub - REPO\_CBA](https://github.com/sfc-gh-mhindi/REPO_CBA)
* **Legacy Documentation:** Available in translated folder for reference
* **Business Logic:** Preserved from original Teradata implementation

## **🏁 Conclusion**

This dbt project represents a comprehensive enterprise data pipeline migration from Teradata to Snowflake, featuring:

✅ **Robust Process Orchestration** - Complete ETL workflow management  
✅ **Data Quality Controls** - Built-in validation and rejection handling  
✅ **Business Logic Preservation** - Legacy system logic maintained  
✅ **Scalable Architecture** - Multi-layered processing approach  
✅ **Comprehensive Monitoring** - Full audit trail and process tracking

The project is production-ready with proper configuration and provides a solid foundation for CBA’s data transformation needs in Snowflake.

**Created:** August 6, 2024  
**Version:** 1.0  
**Author:** CBA Data Engineering Team  
**Location:** NPW DBT - My Changes/README\_NPW\_DBT\_Implementation\_Guide.md