**Detailed Solution**

To overcome the challenge of decrypting Oracle-encrypted data within PostgreSQL while maintaining security and compatibility, I developed a sophisticated ETL (Extract, Transform, Load) workflow in Informatica PowerCenter 9.6.1. This solution leverages Informatica’s transformation capabilities to securely process sensitive data—encrypted in Oracle and re-encrypted for PostgreSQL—without exposing it to human visibility. By orchestrating decryption in Oracle and re-encryption in PostgreSQL through Procedure Transformations, the workflow ensures a seamless and secure migration from an Oracle source to a PostgreSQL 17 target. Below is a detailed breakdown of the solution:

1. **Source Configuration and Data Extraction**:
   * **Oracle as Source**: The source database was configured as an Oracle instance containing the emp\_sal\_details table, where sensitive data like the salary column was encrypted using Oracle’s proprietary encryption (e.g., a custom function text\_decrypt). An Oracle relational connection was established in Informatica Workflow Manager using appropriate credentials and connection parameters.
   * **Selective Extraction**: To minimize exposure and optimize performance, the ETL workflow extracts only the encrypted sensitive data (e.g., salary) along with necessary identifiers (e.g., empid, empname, deptid). This was achieved by defining a source definition in PowerCenter Designer and using a Source Qualifier with a simplified SQL query:

SELECT empid, empname, salary, deptid FROM emp\_sal\_details

* + - Here, salary is retrieved in its encrypted form directly from Oracle, avoiding any premature decryption outside the database.

1. **Decryption Using Oracle Procedure Transformation**:
   * **Procedure Transformation Setup**: In the Informatica mapping, I introduced a Procedure Transformation named Proc\_Decrypt\_Oracle to handle decryption within Oracle’s secure environment. This transformation connects to the Oracle database using the established Oracle connection.
   * **Oracle Decryption Function**: The transformation invokes a pre-existing Oracle stored function, text\_decrypt, which is designed to decrypt the salary column. The function, assumed to be part of Oracle’s encryption framework (e.g., using DBMS\_CRYPTO), takes the encrypted salary as input and returns the decrypted value. The call is structured as:

text\_decrypt(:IN.salary)

* + **Security Assurance**: By performing decryption within Oracle, the sensitive data remains confined to the database’s security boundary, ensuring no plain-text exposure occurs during the ETL process. The decrypted output is passed as a variable (e.g., decrypted\_salary) to the next stage of the workflow.

1. **Re-encryption Using PostgreSQL Procedure Transformation**:
   * **Procedure Transformation Setup**: A second Procedure Transformation, Proc\_Encrypt\_Postgres, was added to the mapping, connected to the PostgreSQL 17 target using the ODBC connection (PG17\_ODBC\_Conn with DSN Postgres\_DSN).
   * **PostgreSQL Encryption Function**: This transformation calls a custom PostgreSQL function, pg\_encrypt, created in the public schema of the target database using the pgcrypto extension. The function re-encrypts the decrypted data using PostgreSQL’s symmetric encryption algorithm. The function was defined as:

CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE OR REPLACE FUNCTION public.pg\_encrypt(plain\_text text)

RETURNS text AS $$

BEGIN

RETURN encode(pgp\_sym\_encrypt(plain\_text, 'your\_secret\_key'), 'base64');

END;

$$ LANGUAGE plpgsql;

* + - The decrypted salary from Oracle is passed as input (e.g., :IN.decrypted\_salary), and the function outputs the re-encrypted value compatible with PostgreSQL.
  + **Execution**: The transformation ensures that the data is re-encrypted before it leaves Informatica, maintaining confidentiality throughout the pipeline.

1. **Target Loading into PostgreSQL**:
   * **Target Definition**: The target table in PostgreSQL 17 (e.g., emp\_sal\_details\_pg) was defined in PowerCenter Designer with columns matching the source (empid, empname, salary, deptid), where salary now holds the PostgreSQL-encrypted data.
   * **Connection Configuration**: In the session properties (s\_CROSS\_ENCRYPTION), the target connection was set to PG17\_ODBC\_Conn, ensuring the re-encrypted data is written to PostgreSQL via the ODBC driver (psqlodbc35w.dll).
   * **Data Loading**: The workflow loads the re-encrypted salary along with other columns into the PostgreSQL target table, completing the migration securely.
2. **Workflow Implementation and Validation**:
   * **ETL Workflow**: The complete workflow was constructed in Workflow Manager, linking a Start task to the session s\_CROSS\_ENCRYPTION, which executes the mapping m\_PG17\_to\_Target. The session was configured to use the Oracle connection for the source and the PostgreSQL ODBC connection for the target.
   * **Security Measures**: Throughout the process, sensitive data remains encrypted or processed within database boundaries, with no plain-text exposure in Informatica’s memory or logs.
   * **Testing and Validation**: The workflow was tested by running it in Workflow Monitor, verifying that:
     + Encrypted data from Oracle is successfully decrypted and re-encrypted.
     + The re-encrypted data is correctly loaded into PostgreSQL.
     + A subsequent decryption test in PostgreSQL (using pgp\_sym\_decrypt) confirms the data’s integrity:

SELECT empid, empname, pgp\_sym\_decrypt(decode(salary, 'base64'), 'your\_secret\_key') AS salary, deptid FROM emp\_sal\_details\_pg;

1. **Resolution of Technical Challenges**:
   * **ODBC Configuration**: Initial connectivity issues with PostgreSQL 17 were resolved by installing the 64-bit PostgreSQL ODBC driver (psqlodbc\_15\_00\_0000-x64), configuring the DSN (Postgres\_DSN), and updating powrmart.ini with PostgreSQL=psqlodbc35w.dll.
   * **Procedure Visibility**: Although PostgreSQL procedures/functions weren’t initially visible in Informatica’s Procedure Transformation due to compatibility limitations, I manually specified the function calls (text\_decrypt and pg\_encrypt) after ensuring schema and permission settings.

**Outcome**: This solution effectively bridges the encryption incompatibility between Oracle and PostgreSQL by leveraging Informatica’s Procedure Transformations to perform decryption in Oracle and re-encryption in PostgreSQL. The ETL workflow ensures that sensitive data remains secure without human visibility, achieving a seamless migration from an Oracle source to a PostgreSQL 17 target. The approach eliminates the need to decrypt Oracle data directly in PostgreSQL, providing a scalable and secure method for handling encrypted data across heterogeneous database systems.

**Key Elements**

* **Source and Target**: Clearly defines Oracle as the source and PostgreSQL 17 as the target.
* **Transformations**: Details the use of two Procedure Transformations for decryption and re-encryption.
* **Security**: Emphasizes end-to-end encryption with no plain-text exposure.
* **Technical Details**: Includes specific function names, SQL snippets, and Informatica configurations.
* **Outcome**: Highlights the successful resolution of the encryption challenge.