Sri Lanka Institute of Information Technology

Data Warehousing and Business Intelligence
IT3030 [2022/FEB]

ASSIGNMENT – 01

Submitted by:

Registration Number:

Name: Gobisan A.

Batch:

Date of Submission: 17/05/2022

This report is submitted in partial fulfilment of the requirement of the Data Warehousing and Business Intelligence module.

Declaration

I declare that this project report does not incorporate, without acknowledgment, any material previously submitted for a degree in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text.

Name of the Candidate: Gobisan A.

Table of Contents

List of Figures

1. Dataset Selection

1.1. Description of Data Set

The data set is an OLTP dataset that contains anonymized financial data from a bank. The original purpose of this data set is to predict the customer's behavior on loan payments.

The dataset contains details about bank customers, their accounts, transactions and standing orders made through the account, the granted loans, the issued credit cards, and disposition for the credit cards. The data set has around 4 years of data and the total data set is about 70 Mb in size. The data set has been modified and extended to meet the requirements of the assignment.

1.2. Clickable Link to Dataset

https://www.kaggle.com/datasets/zhunqiang/bank-data-loan-default

1.3. Entity Relationship Diagram

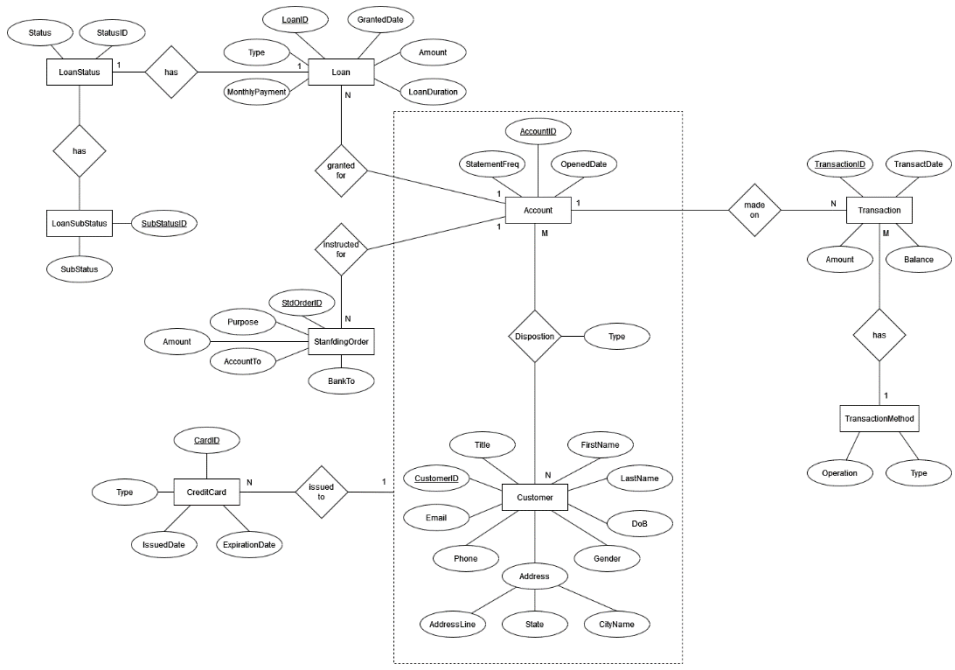


Figure 1. 1 ER Diagram

2. Preparation of Data Source

2.1. Preparation Methodology

The original data set was modified and extended to align with the requirements of the project. The original dataset contains 8 armored ASCII file, of which card file was converted as text file, order file was converted as csv file and other files were converted as database backup file. The client file was extended with some additional columns (name, dob, phone, email etc.) and auto generated records. Customer address file is an auto generated file, and it has proper relationship with customer file. Some values were translated from Czech to English to improve understandability.

2.2. Source Details

2.2.1. SQL Database Source

▪ **Source:** DWBI_BankSourceDB	▪ **Source Type:** SQL Database
▪ **Schema:** dbo	▪ **Object Type:** Table

Object Name	Description
Account	Includes all account details
Customer	Includes all customer information
Disposition	Includes information about rights of customer to operate accounts
Loan	Includes information about loan granted for accounts
LoanStatus	Includes loan status information
LoanSubStatus	Includes loan sub status information
Transaction	Includes details of transactions made on accounts

Figure 2. 1 Data Source Description

2.2.2. Other Sources

Source	Source Type	Description
CreditCard	Text File	Includes details of credit card issued to accounts
CustomerAddress	Excel File	Includes the customer address details
StandingOrder	CSV File	Includes details of standing orders instructed for accounts

3. Solution Architecture

3.1. Solution Architectural Diagram

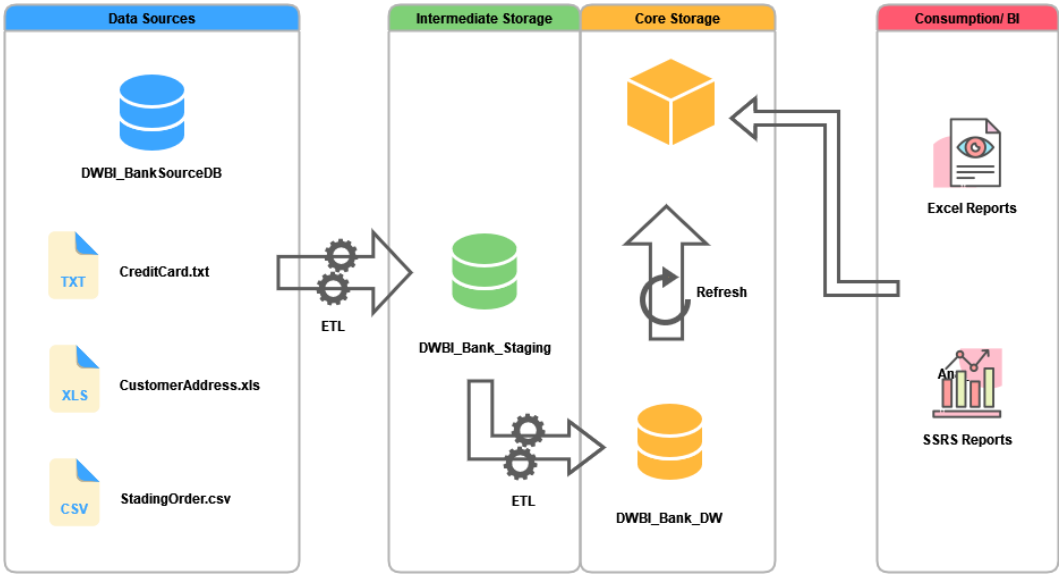


Figure 3. 1 Architectural Diagram

3.2. Architectural Components

3.2.1. Data Sources

Heterogeneous source of data was used for the solution. Actual data was spread across several sources, and they were not integrated.

The source data are available in the,

- Structured Format - DWBI_BankSourceDB a Relational Database
- Semi - structured Format - CreditCard.txt, CustomerAddress.xls and StandingOrder.csv

3.2.2. ETL from Data Sources to Staging Database

- Full extraction approach was used to extract whole original source tables and source files as it is and load them into staging database. In this process no transformations were done it is being done when loading into the target data warehouse.

3.2.3. Intermediate Storage

- DWBI_Banking_Staging database contains the whole data extracted from the data sources, as relations, in a centralized location. This staging database was created to make available all required data before loading it into the Data Warehouse. Before every ETL run each table in the staging database is being truncated since full extraction method is used for extraction.

3.2.4. ETL from Staging Database to Data Warehouse

- Change data capturing approach was used to extract data from staging. Necessary transformations were done as per data warehouse terms. After Transformation, the data was finally loaded into the DWBI_Bank_DW target Data warehouse.

3.2.5. Core Storage

- The DWBI_Bank_DW is a centralized data warehouse that classifies and stores the data according to the subject. This data warehouses uses a unified approach to represent data This data warehouse supports analytics and reporting services through the OLAP cube.

3.2.6. Consumption/ BI

- Reporting and analytics services were used to represent data insights.

4. Data Warehouse Design and Development

4.1. Data Warehouse Schema

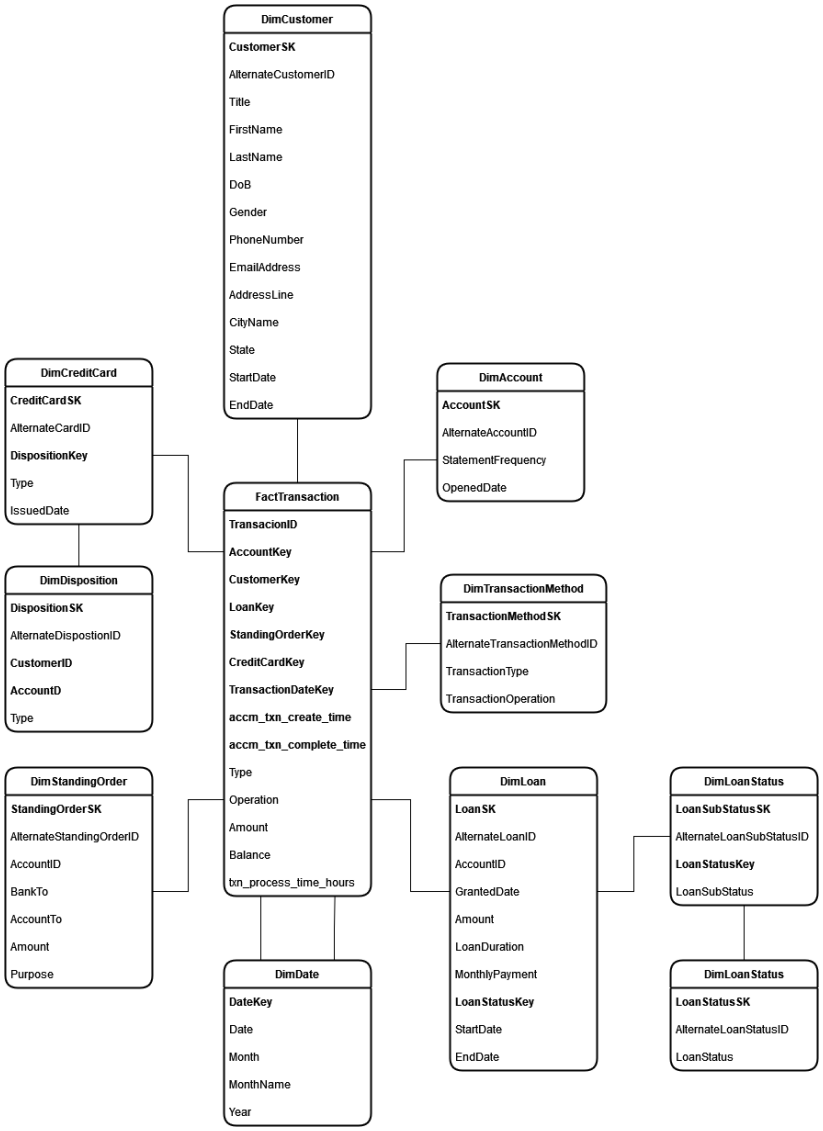


Figure 4. 1 Snowflake Schema

4.2. Data Warehouse Dimensional Model Description

A snowflake schema was used to implement the data warehouse. Snowflake schema was chosen because it is less prone to data integrity issues, easier to maintain, and utilizes less space. In the above snowflake schema, there are 8 dimensional tables, and 1 fact table are available.

4.2.1. Hierarchical Data

The Customer Dimension (DimCustomer) contains hierarchical data on customer address. (AddressLine, City, State)

Loan Dimension have reference to Loan Sub Status Dimension and Loan Sub Status Dimension have reference to Loan Status Dimension. Therefore these three dimensions together forms a hierarchy (Loan, Sub-Status, Status)

4.2.2. Slowly Changing Dimension

The Customer Dimension was modeled as a Slowly Changing Dimension (SCD), to accommodate the history management requirement of customer data. In Customer dimension, phone is a changing attribute (Type 1), and address line, city and state are historical attributes (Type 2).

The Loan dimension (DimLoan) also modeled as a SCD, because Loan Status (Historical Attribute - Type 2) column in Loan Dimension should be maintained for historical data analysis.

4.2.3. Accumulating Fact Table

The Account Transaction Fact table (FactTransaction) was modeled as an accumulating fact table to record the lifetime of a transaction from the time it is created to the time it is completed. The grain of the fact table is the transaction made on a particular account on a particular date.

5. ETL Development

5.1. Source Data Extraction and Staging Area Loading

5.1.1. Process Description

As the first step of the SSIS ETL process data was extracted from source systems to DWBI_Bank_Staging Database. There were three data sources available, respectively DWBI_BankSourceDB (SQL database source), CreditCard details (text file), Customer Addresses (Excel file) and Standing Order details (csv file).

To extract data from source database tables to staging area tables, within the data flow task, OLE DB Source component was used, whereas text and csv files were extracted using Flat File Source component and excel file was extracted using Excel Source component.

Transformations were not done at this stage, because the primary concern was to extract data from heterogenous data sources and load them to the staging area. The extracted source data were loaded into the respective staging tables using OLE DB Destination component. Each data flow task for extraction was attached with OnPreExcute Event Handler to truncate respective staging tables, where the truncation was done using Execute SQL Task component.

5.1.2. Process Design Snippets

1. **Control Flow Design**

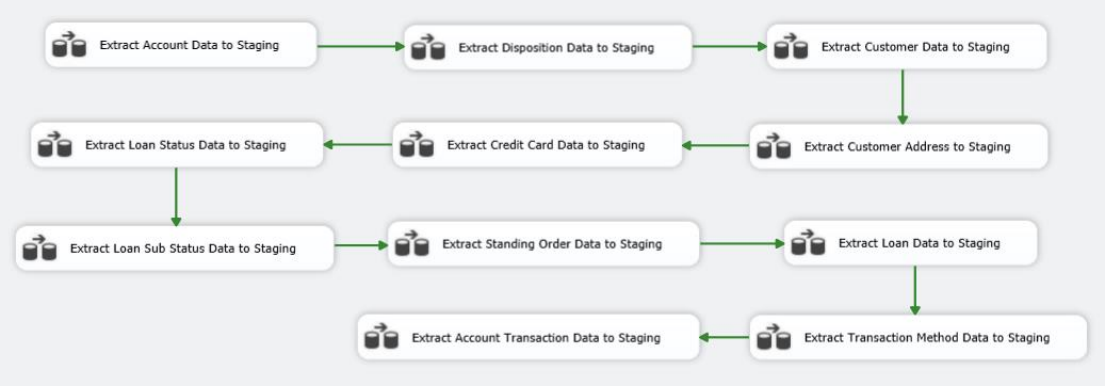


Figure 5. 1 Control Flow Design for Staging Process

2. Data Flow Design

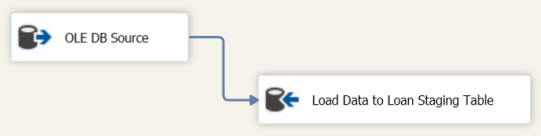


Figure 5. 2 Data Extraction from OLE DB Source

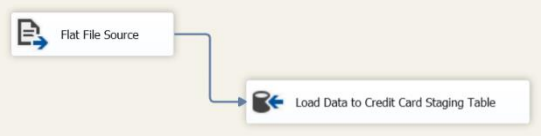


Figure 5. 3 Data Extraction from Flat File Source

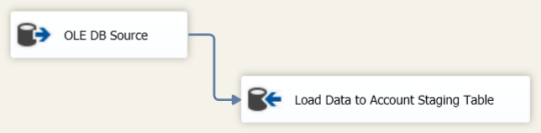


Figure 5. 4 Data Extraction from OLE DB Source

Figure 5. 5 Data Extraction from Excel File Source

3. Even Handlers

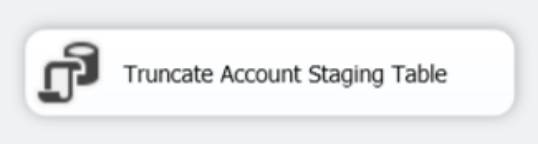


Figure 5. 6 Truncate Staging Table

5.2. Data Profiling

The staging table data was used to analyze how the data looks like to determine what type of transformations (data cleansing) were needed to be done on the data. This process was done to identify data quality issues, such as NULL values, that need to be fixed in ETL process.

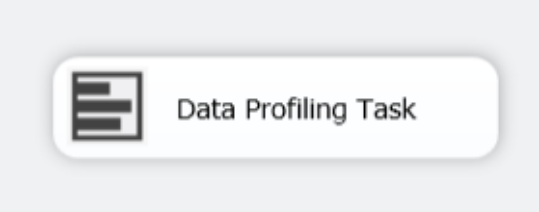
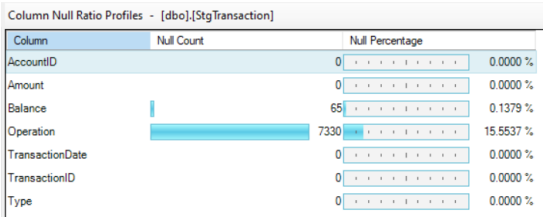


Figure 5. 7 Data Profiling Task


Column Null Ratio Profiles - [dbo].[StgTransaction]

Column	Null Count	Null Percentage
AccountID	0	0.0000 %
Amount	0	0.0000 %
Balance	65	0.1379 %
Operation	7330	15.5537 %
TransactionDate	0	0.0000 %
TransactionID	0	0.0000 %
Type	0	0.0000 %

Figure 5. 8 Transaction Staging Table Data Profile

5.3. Data Transformation and Data Warehouse Loading

5.3.1. Overall Process Description

At this data transformations were performed on the staging database and loaded them into the data warehouse. Order of the tables that are needed to be loaded into data warehouse was determined based on dependencies (foreign key references).

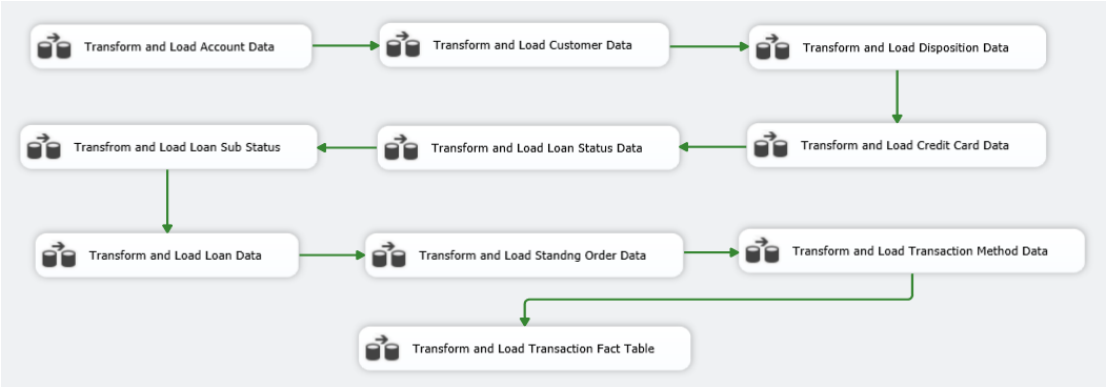


Figure 5. 9 Control flow task of ETL for Data Warehouse

5.3.2. Loading Dimensions

1. Loading Hierarchical Data

DimLoan table contains a reference through a foreign key to DimLoanSubStatus table. Similarly, DimLoanSubStatus table contains a foreign key to DimLoanStatus table. Thus, first DimLoanStatus table was loaded, next the DimLoanSubStatus table and finally the DimLoan table. While loading Hierarchical data lookup components were used.

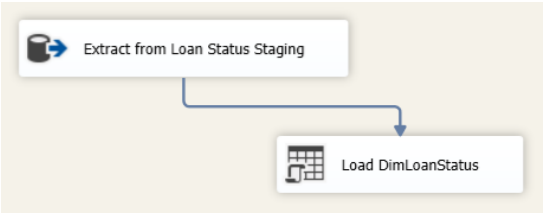


Figure 5. 10 Load Loan Status Dimension

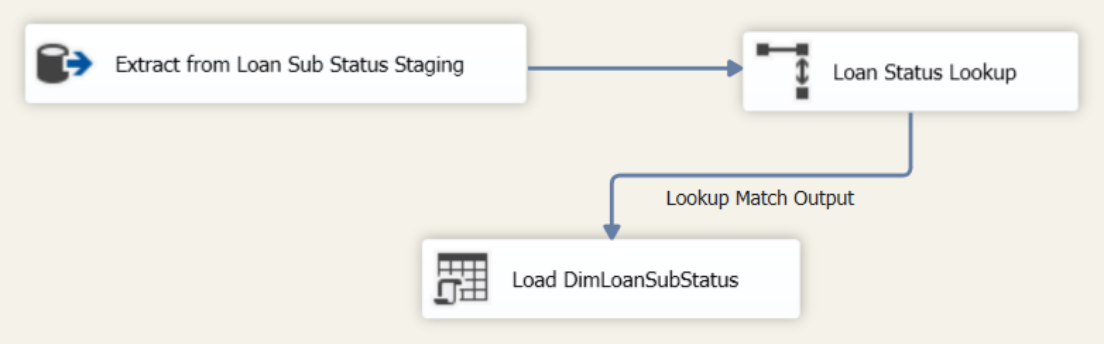


Figure 5. 11 Load Loan Sub Status Dimension

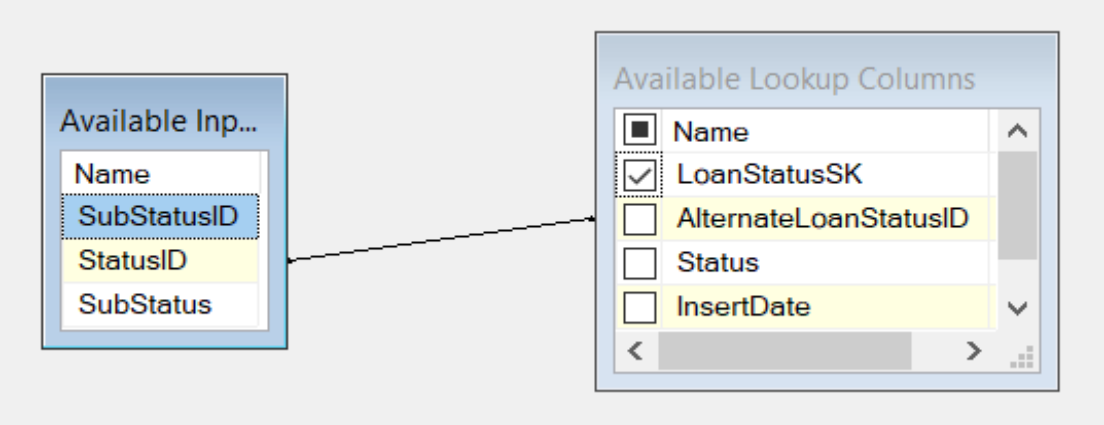


Figure 5. 12 Column Lookup

2. Loading Slowly Changing Dimensions

The Customer Dimension was modeled as a Slowly Changing Dimension (SCD), to keep track of historical addresses of customers. Additionally, if the customer changes their phone number, it should be replaced with the new number.

Since customer data and customer address data are present in two different staging tables, they were sorted based on Customer ID and then merged using a merge join component. Left outer join was used to load customer data even if there were customers without corresponding addresses.

Transformations were done to replace NULL gender values with 'NA', and to derive and replace NULL title values based on gender values. To include the insert and modified date a derived column component was used.

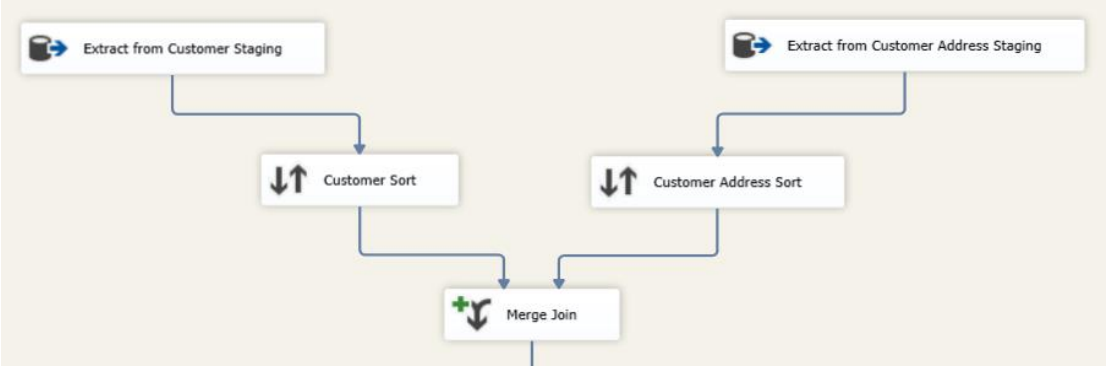


Figure 5. 13 Merge join Customer Address data with Customer Data

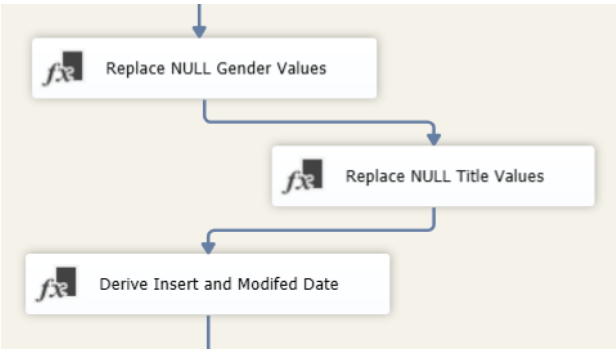


Figure 5. 14 Data Transformations

Slowly changing dimension component was used to load customer data as a Slowly Changing Dimension. The Phone attribute is considered as changing attribute (type 1) where these attributes will get updated when source data values change.

The attributes Address Line, City and State are considered as historical attributes (type 2) where a new record will get inserted in the target table when these attributes change in source table thus ensuring history management.

Each record contains the start date and end date to identify the time between which record was active and to identify the current record.

Rest of the attributes are, by default, specified as fixed attribute (type 0) where nothing changes in the target table when these values are get changed in the source.

Dimension Columns	Change Type
AddressLine	Historical attribute
City	Historical attribute
PhoneNumber	Changing attribute
State	Historical attribute

Figure 5. 15 Slowly Changing Dimension Columns

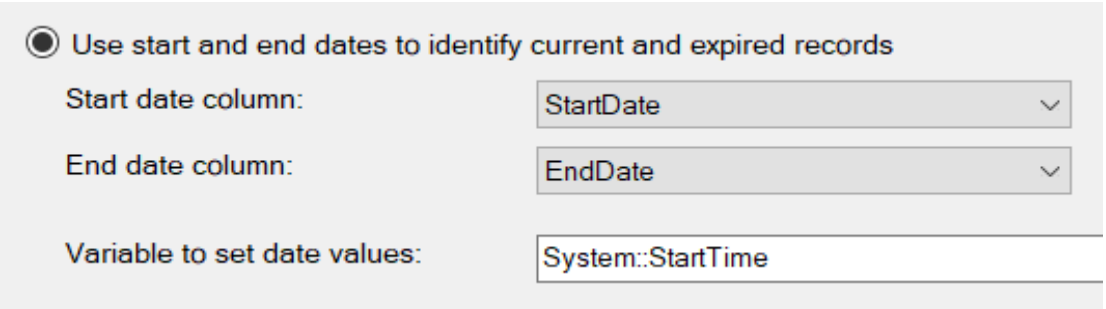


Figure 5. 16 Historical Attribute Options

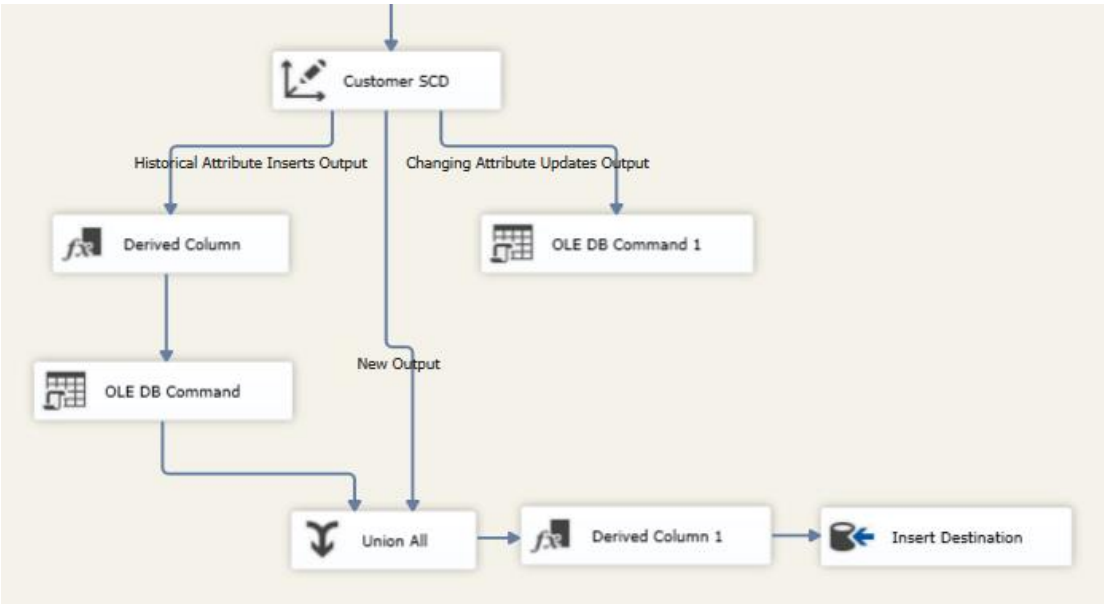


Figure 5. 17 Data flow task to load Customer SCD

5.3.3. Loading Accumulating Fact Table

The transaction fact table (FactTransaction) contains references to account, customer, credit card, loan, standing order, and date dimensions. To get the surrogate keys of respective dimensions as foreign key references, lookup processes were carried out. Data cleansing step was done to replace the NULL operation values to an appropriate value. Derived column component was used to derive insert date and modified date.

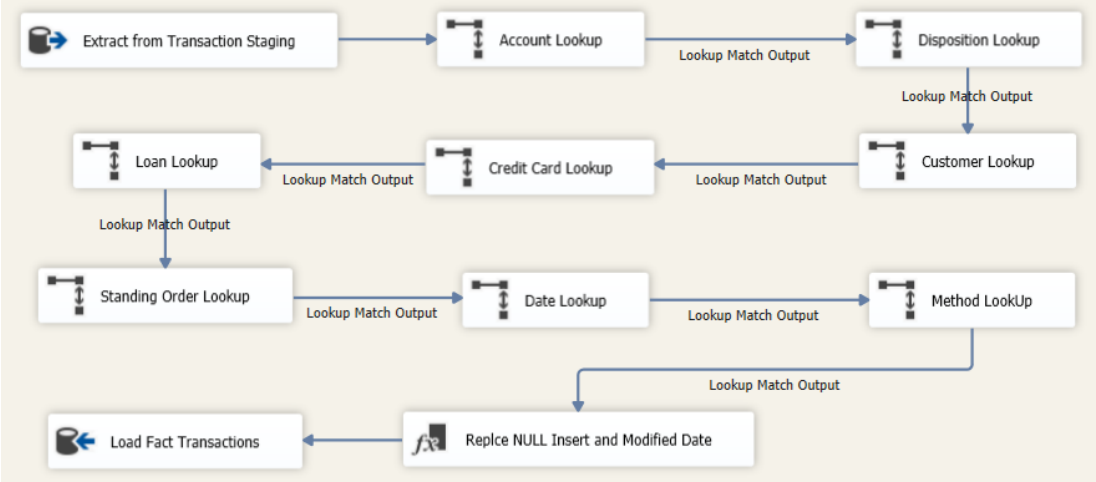


Figure 5. 18 Data flow task to load Transaction Fact table

Derived Column Name	Derived Column	Expression	Data Type	Length
Operation	Replace 'Operation'	REPLACENULL(Operation,"NA")	Unicode string [DT_WSTR]	50

Figure 5. 19 Derived column transformations to replace NULL operation value

Derived Column Name	Derived Column	Expression	Data Type
InsertedDate	<add as new column>	GETDATE()	database timestamp [DT_DBTIMESTAMP]
ModifiedDate	<add as new column>	GETDATE()	database timestamp [DT_DBTIMESTAMP]

Figure 5. 20 Derived column transformations to derive insert and modified data

6. Update Accumulating Fact Table

6.1. Description

The FactTransaction fact table is an Accumulating Snapshot Fact table. The ggrain of the fact table is a transactions done on a particular account on a particular date.

The fact represents account transaction process throughout its creation date(accm_txn_create_time) and completion date(accm_txn_complete_time). Additionally txn_process_time_hours represents the difference in hours between accm_txn_complete_time and accm_txn_create_time.

6.2. ETL Process update accumulating fact table

The CompletedTransaction csv file which contains transaction id and transaction complete date was extracted a using flat file component and loaded into the StgCompletedTransaction table of staging database. Then necessary transformations were done and updated the fact table with respective data.

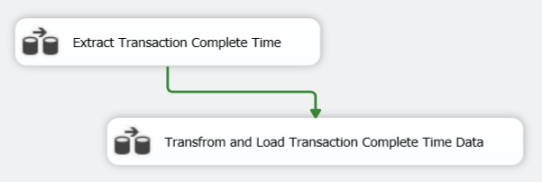


Figure 5. 21 Control flow to update accumulating fact table

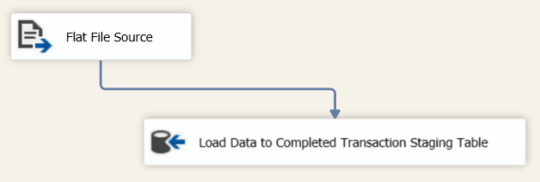


Figure 5. 22 Data flow design to extract and load Completed transaction staging table

Steps followed in in the ETL process for updating accumulating fact table:

- First, Transaction ID (Business Key) and completed data were extracted from Completed Transaction staging table (StgCompletedTransaction)

- Next, a Lookup component was used to perform lookups by joining TransactionID in StgCompletedTransaction with TransactioID in FactTransactions. This lookup was performed to access accm_txn_complete_time data from FactTransactions.

- Then, another Lookup component was used to perform lookups by joining accm_txn_create_time in FactTransactions with DateKey in DimDate. This lookup was performed to retrieve the respective date for accm_txn_create_time.

- After that, a derived column component was used to derive the difference in hours between accm_txn_complete_time and accm_txn_create_time, then the retrieved values were used to populate the txn_process_time_hours column.

- Next, another Lookup component was used to perform lookups by joining accm_txn_complete_time in FactTransactions with Date in DimDate. This lookup was performed to retrieve the respective date key (SK) for accm_txn_complete_time.

- Finally, an OLE DB command components was used to update the FactTransaction with the surrogate key transaction complete time and transaction process time. A stored procedure was used to perform this update operation

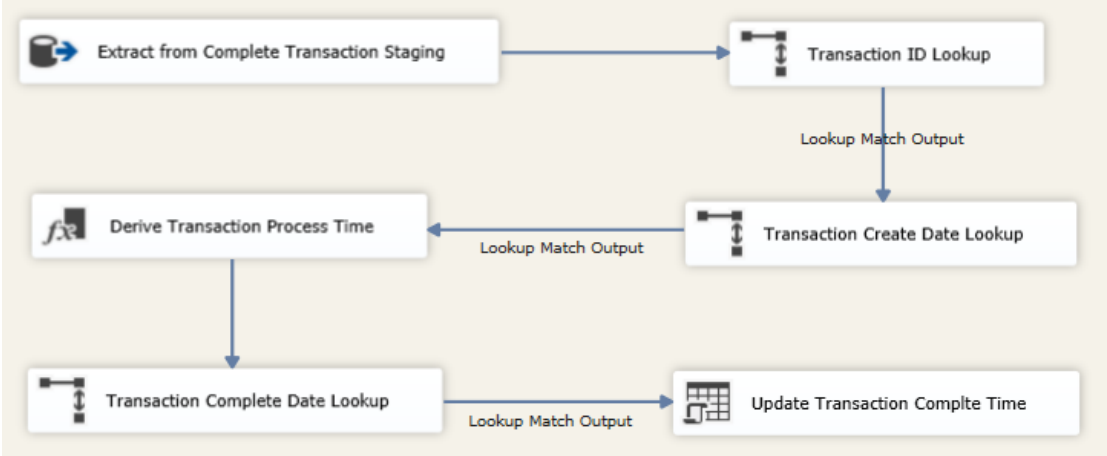


Figure 5. 23 Dataflow Design to Transform and Load Transaction Complete Time Data

Derived Column Name	Derived Column	Expression	Data Type
txn_process_time_ho...	Replace 'txn_process_time_hours'	DATEDIFF("Hh",Date,CompletedDate)	four-byte signed integer [DT_I4]

Figure 5. 24 Derived column transformations to derive transaction process time

```
CREATE PROCEDURE [dbo].[UpdateFactTransaction]
@TransactionID int,
@CompleteTime int,
@ProcessTime int
AS
BEGIN
    UPDATE [dbo].[FactTransactions]
    SET [accm_txn_complete_time] = @CompleteTime
        ,[txn_process_time_hours] = @ProcessTime
        WHERE TransactionID = @TransactionID
END;
```

Figure 5. 25 Stored procedure to update accm_txn_complete_time and accm_txn_create_time

References

[1]"Slowly Changing Dimensions", *Oracle.com*, 2022. [Online]. Available: https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/10g/r2/owb/owb10gr2_gs/owb/lesson3/slowlychangingdimensions.htm. [Accessed: May- 2022]

[2]"IBM Docs", *Ibm.com*, 2022. [Online]. Available: https://www.ibm.com/docs/en/ida/9.1.2?topic=models-fact-tables-entities. [Accessed: 11- May- 2022].

[3]"Integration Services Developer Documentation - SQL Server Integration Services (SSIS)", *Docs.microsoft.com*, 2022. [Online]. Available: https://docs.microsoft.com/en-us/sql/integration-services/integration-services-developer-documentation?view=sql-server-ver15. [Accessed: 11- May- 2022].

Appendix

i. Test Cases for SCD

Test cases were carried out by updating the Customer Staging table to check the data population in the slowly changing dimension thus analyzing the modifications in DimCustomer. Below test was conducted to check the historical attribute city.

DimCustomer Record - Before Updating City:

CustomerSK	AlternateCustomerID	City
34	36	Austin

Update Customer Address Staging Table:

```
UPDATE [DWBI_Bank_Staging].[dbo].[StgCustomerAddress]
SET City = 'Houstan'
WHERE CustomerID = 36;
```

DimCustomer Record - After Updating City:

CustomerSK	AlternateCustomerID	City
34	36	Austin
5370	36	Houstan

ii. Stored Procedures

Dimensions like account, standing order, disposition and credit card does not maintain history. Therefore, to maintain the latest record, stored procedures were created.

A sample stored procedure is given below:

```
CREATE PROCEDURE [dbo].[UpdateDimLoanStatus]
@LoanStatusID int,
@Status nvarchar(50)
AS
BEGIN
    IF NOT EXISTS (SELECT LoanStatusSK
                        FROM [dbo].[DimLoanStatus]
                        WHERE AlternateLoanStatusID = @LoanStatusID)
        BEGIN
            INSERT INTO [dbo].[DimLoanStatus]
               (AlternateLoanStatusID
               ,[Status]
               ,[InsertDate]
               ,[ModifiedDate])
            VALUES
                (@LoanStatusID
                ,@Status
                ,GETDATE()
                ,GETDATE())
        END;

    IF EXISTS (SELECT LoanStatusSK
                        FROM [dbo].[DimLoanStatus]
                        WHERE AlternateLoanStatusID = @LoanStatusID)
        BEGIN
            UPDATE [dbo].[DimLoanStatus]
            SET [Status] = @Status
               ,[ModifiedDate] = GETDATE()
                WHERE AlternateLoanStatusID = @LoanStatusID
                AND [Status] != @Status
        END;
END;
```

Figure 6. 1 Stored Procedure to update Loan Status Dimension

iii. Dimensions and Fact Creation Queries

```
CREATE TABLE [DWBI_Bank_DW].[dbo].[DimAccount](
    AccountSK int IDENTITY(1,1) PRIMARY KEY,
    AlternateAccountID int NOT NULL,
    StatementFrequency nvarchar(50),
    SrcOpenedDate Date,
    InsertDate DateTime,
    ModifiedDate DateTime,
);
```

Figure 6. 2 Query to create DimAccount

```
CREATE TABLE [DWBI_Bank_DW].[dbo].[FactTransactions](
    [TransactionID] [int] NULL,
    [AccountKey] [int] FOREIGN KEY REFERENCES DimAccount(AccountSK),
    [CustomerKey] [int] FOREIGN KEY REFERENCES DimCustomer(CustomerSK),
    [CreditCardKey] [int] FOREIGN KEY REFERENCES DimCreditCard(CreditCardSK),
    [LoanKey] [int] FOREIGN KEY REFERENCES DimLoan(LoanSK),
    [StandingOrderKey] [int] FOREIGN KEY REFERENCES DimStandingOrder(StandingOrderSK),
    [accm_txn_create_time] [int] FOREIGN KEY REFERENCES DimDate(DateKey),
    [accm_txn_complete_time] [int] FOREIGN KEY REFERENCES DimDate(DateKey),
    [Type] [nvarchar](50) NULL,
    [Operation] [nvarchar](50) NULL,
    [Amount] [money] NULL, -- Business Measure
    [Balance] [money] NULL, -- Business Measure
    [txn_process_time_hours] [INT] NOT NULL DEFAULT 0,
    [InsertDate] [datetime],
    [ModifiedDate] [datetime]
);
```

Figure 6. 3 Query to Create FactTransaction