

INFORMATICA

CASE STUDY

The Order Entry Operational Data Model and Schema

By
Akash J

Table of Contents

PROBLEM STATEMENT	2
OVERVIEW OF DATA SOURCE, STAGING DATABASE AND DATA WAREHOUSE	2
1. SOURCE FILES	2
2. STAGING DATABASE.....	2
3. DATA WAREHOUSE	3
ETL PROCESS	4
1. FROM CSV FILES TO STAGING DATABASE	4
2. FROM STAGING DATABASE TO DATA WAREHOUSE	11
TYPES OF SLOWLY CHANGING DIMENSIONS IMPLEMENTED FOR DIMENSION TABLES ..	13
1. SLOWLY CHANGING DIMENSION TYPE 1	13
2. SLOWLY CHANGING DIMENSION TYPE 2	13
MAPPINGS FROM STAGING DATABASE TO DATA WAREHOUSE.....	15
1. CUSTOMER_DIM_SCD2.....	15
2. SALESREP_DIM_SCD2.....	19
3. PRODUCTS_DIM	22
4. PROMOTIONS_DIM	24
5. DATE_DIM	26
6. SALES_FACT	27
CONCLUSION.....	33
SCRIPTS USED FOR TABLE CREATION IN STAGING DATABASE AND IN DATA WAREHOUSE	34

PROBLEM STATEMENT

The scenario is of a business which has some transactional data which needs to be loaded into a data warehouse.

The business offers a set of products for sale and those products fall into a hierarchy of product categories. Customers place orders with the business. Orders consist of multiple order items which are in turn made up of products. An order may be placed under a given promotion and sales rep(employee) may be assisting the customer with the ordering process.

OVERVIEW OF DATA SOURCE, STAGING DATABASE AND DATA WAREHOUSE

The loading of data to the data warehouse(target) is done in two parts.

First the data is loaded into the staging database and from there it is loaded into the data warehouse.

1. SOURCE FILES

The source consists of five CSV files with data related to customers, sales rep, products, promotions and orders.

1. customer_export.csv
2. salesrep_export.csv
3. products_export.csv
4. promotions_export.csv
5. orders_export.csv



customers_export	04-10-2020 21:22	Microsoft Excel Com...	54 KB
orders_export	04-10-2020 21:22	Microsoft Excel Com...	31 KB
products_export	04-10-2020 21:22	Microsoft Excel Com...	81 KB
promotions_export	04-10-2020 21:22	Microsoft Excel Com...	1 KB
salesrep_export	04-10-2020 21:22	Microsoft Excel Com...	3 KB

Fig 1. Source csv format files

2. STAGING DATABASE

A staging database was designed which will act as the source for the data warehouse. A staging database serves a lot purposes.

1. It is used to perform data cleansing and validation before finally loading into the data warehouse.
2. It reduces redundancy from the source data by normalizing the tables suitably for loading into target.

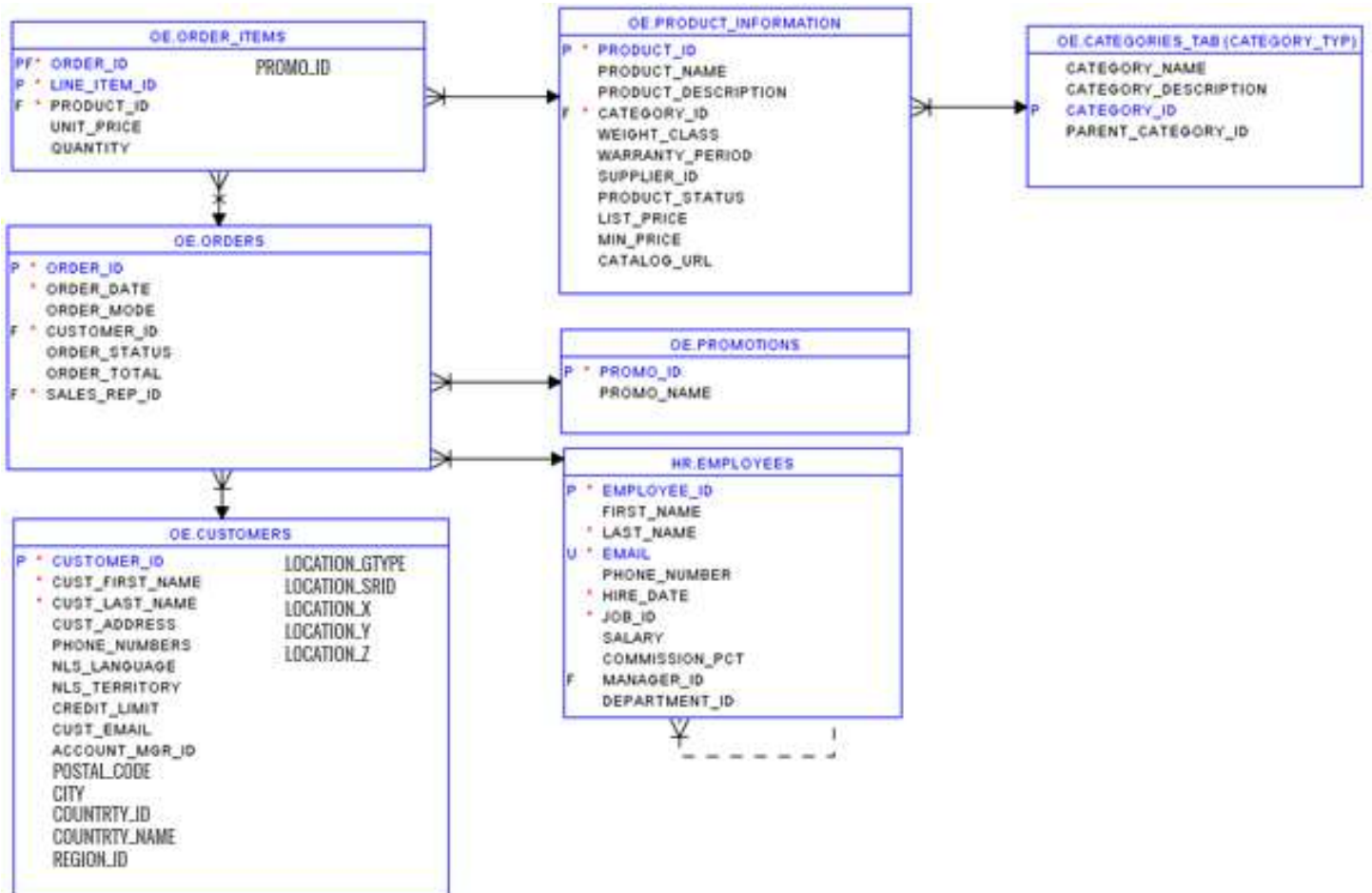


Fig 2. Staging Database Model

3. DATA WAREHOUSE

The data warehouse schema used is star schema. Each dimension is represented with only one dimension table and thus there are five dimension tables. There is a fact table at the center which is surrounded by dimension tables.

Fields like CUSTOMER_DIM_ID, SALESREP_DIM_ID, PRODUCT_DIM_ID, PROMOTION_DIM_ID, DATE_DIM_ID are included in the respective dimension tables as

primary keys. All these keys are included in the fact table and combined they form the composite key of the fact table.

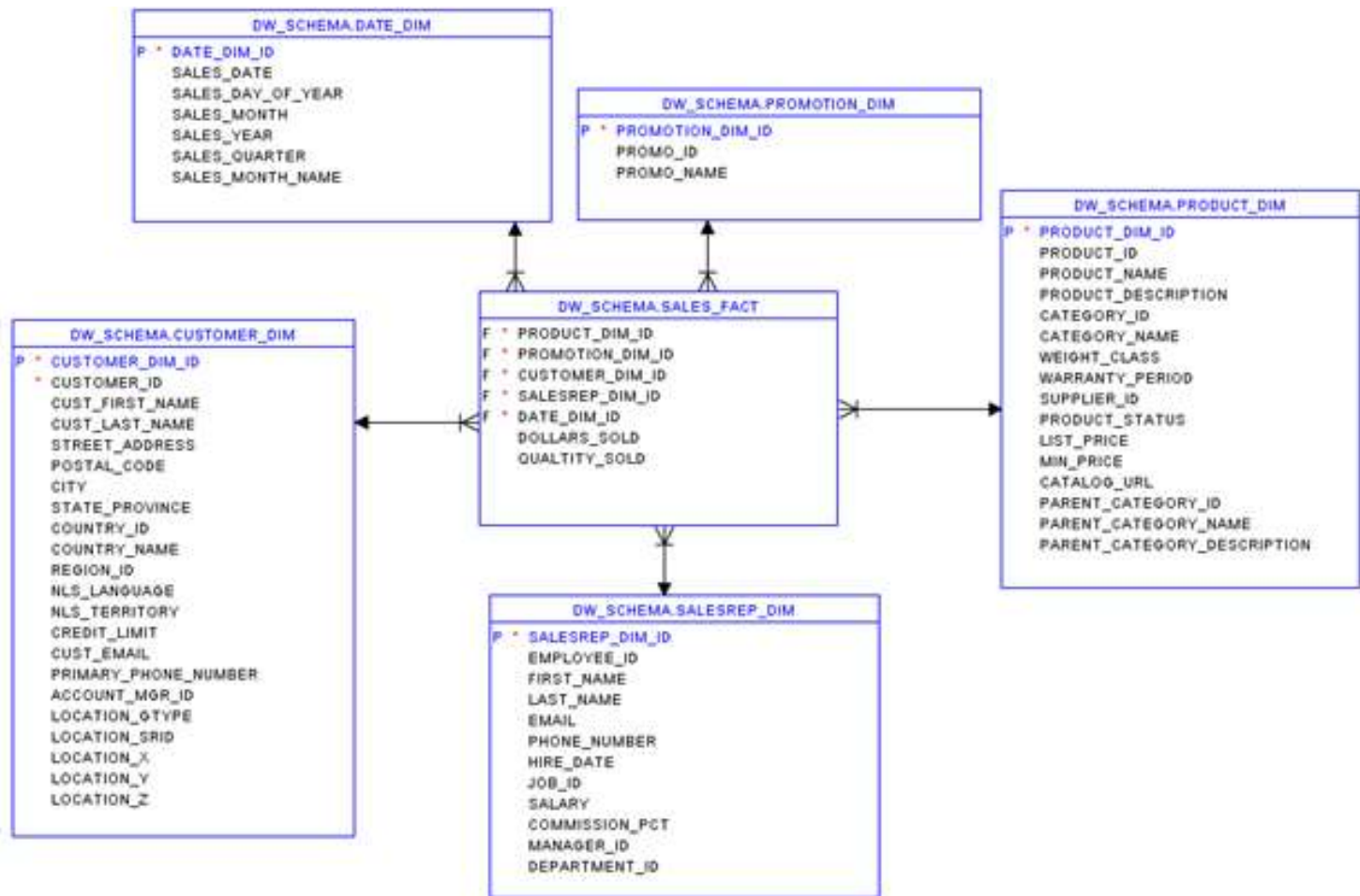


Fig 3. Data Warehouse Model

ETL PROCESS

1. FROM CSV FILES TO STAGING DATABASE

All the csv files are loaded into the staging database by creating respective tables.

Source File	Staging Table
customer_export.csv	customers
salesrep_export.csv	employees
products_export.csv	product_information & categories_tab
promotions_export.csv	promotions
orders_export.csv	orders & order_items

* For products and orders files the data is normalized into two different tables to reduce redundancy.

K	Name	Datatype	L
	CUSTOMER_ID	number	3
	CUST_FIRST	string	256
	CUST_LAST_N	string	256
	STREET_ADD	string	256
	POSTAL_CODE	number	6
	CITY	string	256
	STATE_PROVI	string	256
	COUNTRY_ID	string	256
	COUNTRY_NA	string	256
	REGION_ID	number	1
	NLS_LANGUAGE	string	256
	NLS_TERRITO	string	256
	CREDIT_LIMIT	number	4
	CUST_EMAIL	string	256
	PRIMARY_PHO	string	256
	PHONE_NUMB	string	256
	ACCOUNT_MG	number	3
	LOCATION_GT	number	4
	LOCATION_SRID	number	4
	LOCATION_X	number	10
	LOCATION_Y	number	9

K	Name	Datatype	L
	PRODUCT_ID	number	4
	PRODUCT_NA	string	27
	LANGUAGE_ID	string	2
	MIN_PRICE	number	4
	LIST_PRICE	number	4
	PRODUCT_ST	string	17
	SUPPLIER_ID	number	6
	WARRANTY_P	number	2
	WEIGHT_CLASS	number	1
	PRODUCT_DE	string	829
	CATEGORY_ID	number	2
	CATALOG_URL	string	48
	SUB_Catego	string	9
	SUB_Catego	string	62
	PARENT_CATE	number	2
	CATEGORY_N	string	16

K	Name	Datatype	L
	EMPLOYEE_ID	number	3
	FIRST_NAME	string	256
	LAST_NAME	string	256
	EMAIL	string	256
	PHONE_NUMB	string	256
	HIRE_DATE	string	256
	JOB_ID	string	256
	SALARY	number	5
	COMMISSION	number	4
	MANAGER_ID	number	3
	DEPARTMENT	number	2

K	Name	Datatype	L
	ORDER_ID	number	4
	ORDER_DATE	string	29
	CUSTOMER_ID	number	3
	ORDER_STATUS	number	2
	ORDER_TOTAL	number	8
	SALES_REP_ID	number	256
	PROMO_ID	string	256
	LINE_ITEM_ID	number	2
	PRODUCT_ID	number	4
	UNIT_PRICE	number	6
	QUANTITY	number	3

K	Name	Datatype	L
	ORDER_ID	number	4
	ORDER_DATE	string	256
	CUSTOMER_ID	number	3
	ORDER_STATUS	number	2
	ORDER_TOTAL	number	8
	SALES_REP_ID	number	3
	PROMO_ID	number	1
	LINE_ITEM_ID	number	1
	PRODUCT_ID	number	4
	UNIT_PRICE	number	5
	QUANTITY	number	2

K	Name	Datatype	L
	PROMO_ID	number	1
	PROMO_NAME	string	256

Fig 4. csv sources metadata

K	Name	Datatype	L
	CUSTOMER_ID	number	15
	CUST_FIRST	nvarchar2	25
	CUST_LAST_N	nvarchar2	25
	STREET_ADD	nvarchar2	50
	POSTAL_CODE	number	15
	CITY	nvarchar2	25
	STATE_PROVI	nvarchar2	25
	COUNTRY_ID	nvarchar2	5
	COUNTRY_NA	nvarchar2	50
	REGION_ID	number	15
	NLS_LANGUAGE	nvarchar2	25
	NLS_TERRITO	nvarchar2	25
	CREDIT_LIMIT	number	15
	CUST_EMAIL	nvarchar2	60
	PRIMARY_PHO	nvarchar2	25
	PHONE_NUMB	nvarchar2	25
	ACCOUNT_MG	number	15
	LOCATION_GT	number	15
	LOCATION_SRID	number	15
	LOCATION_X	number	15
	LOCATION_Y	number	15

K	Name	Datatype	L
	EMPLOYEE_ID	number	15
	FIRST_NAME	nvarchar2	25
	LAST_NAME	nvarchar2	25
	EMAIL	nvarchar2	60
	PHONE_NUMB	nvarchar2	25
	HIRE_DATE	nvarchar2	15
	JOB_ID	nvarchar2	10
	SALARY	number	15
	COMMISSION	number	15
	MANAGER_ID	number	15
	DEPARTMENT	number	15

K	Name	Datatype	L
	PROMO_ID	number	15
	PROMO_NAME	nvarchar2	30

K	Name	Datatype	L
	PRODUCT_ID	number	15
	PRODUCT_NAME	nvarchar2	50
	LANGUAGE_ID	varchar2	10
	PRODUCT_DESCRIPTION	nvarchar2	1
	CATEGORY_ID	number	15
	WEIGHT_CLASS	number	15
	WARRANTY_PERIOD	number	15
	SUPPLIER_ID	number	15
	PRODUCT_STATUS	nvarchar2	30
	LIST_PRICE	number	15
	MIN_PRICE	number	15
	CATLOG_URL	nvarchar2	100

K	Name	Datatype	L
	CATEGORY_ID	number	15
	SUB_CATEGORY_NAME	varchar2	20
	SUB_CATEGORY_DESCRIP	varchar2	100
	CATEGORY_NAME	nvarchar2	50
	PARENT_CATEGORY_ID	number	15

ORDERS (Crude)			
K. Name	Datatype	L.	
ORDER_ID	number	15	
ORDER_DATE	varchar	19	
ORDER_MODE	nvarchar2	25	
CUSTOMER_ID	number	15	
ORDER_STATUS	number	15	
ORDER_TOTAL	number	15	
SALES_REP_ID	number	15	

ORDER_ITEMS (Crude)			
K. Name	Datatype	L.	
ORDER_ID	number	15	
LINE_ITEM_ID	number	15	
PRODUCT_ID	number	15	
UNIT_PRICE	number	15	
QUANTITY	number	15	
PROMO_ID	varchar	20	

REG_DATE (Crude)			
K. Name	Datatype	L.	
CALENDAR_DATE	date	19	

Fig 5. Staging Target Metadata

Mappings from csv source to staging database

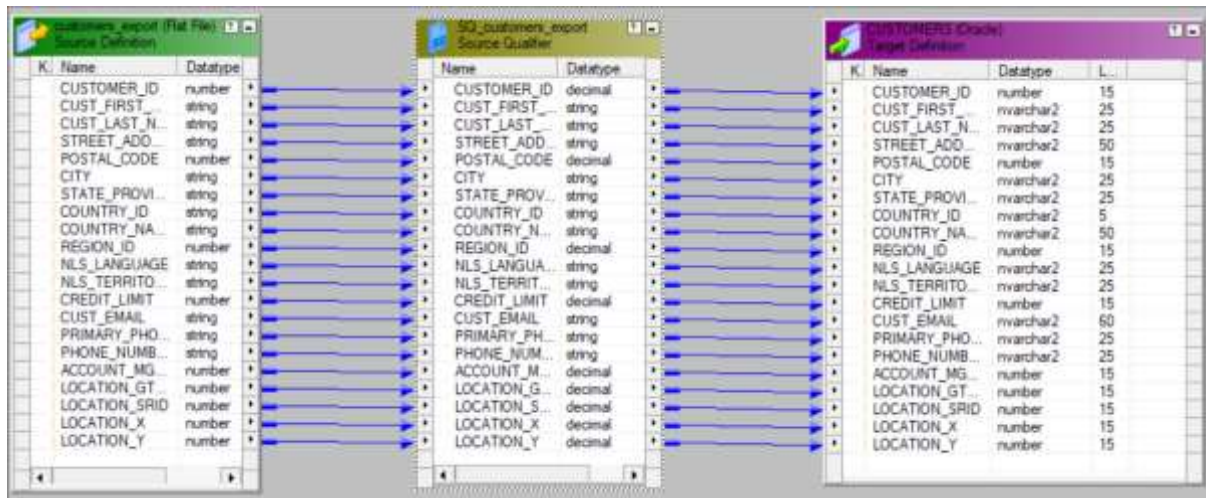


Fig 6. CUSTOMERS MAPPING

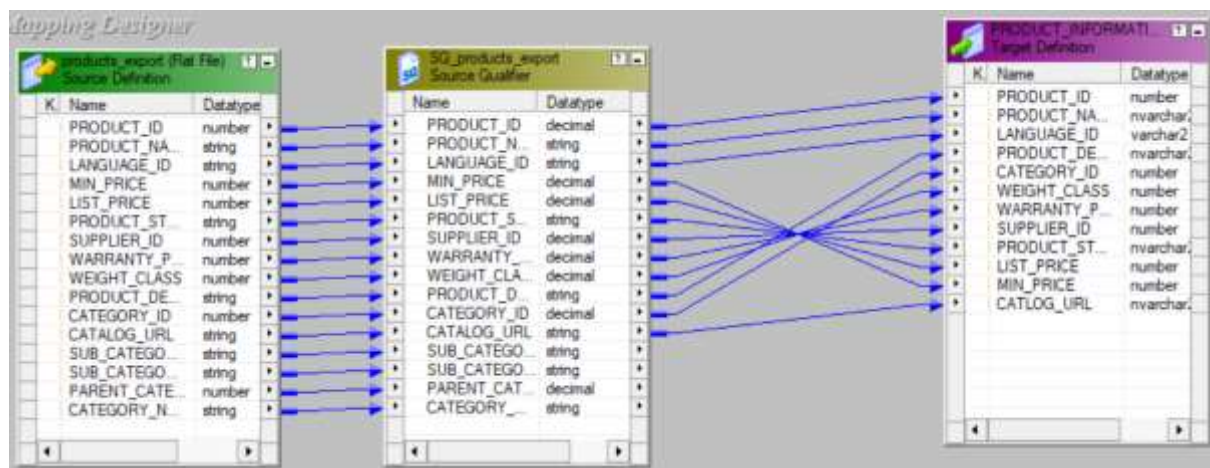


Fig 7. PRODUCT_INFORMATION MAPPING

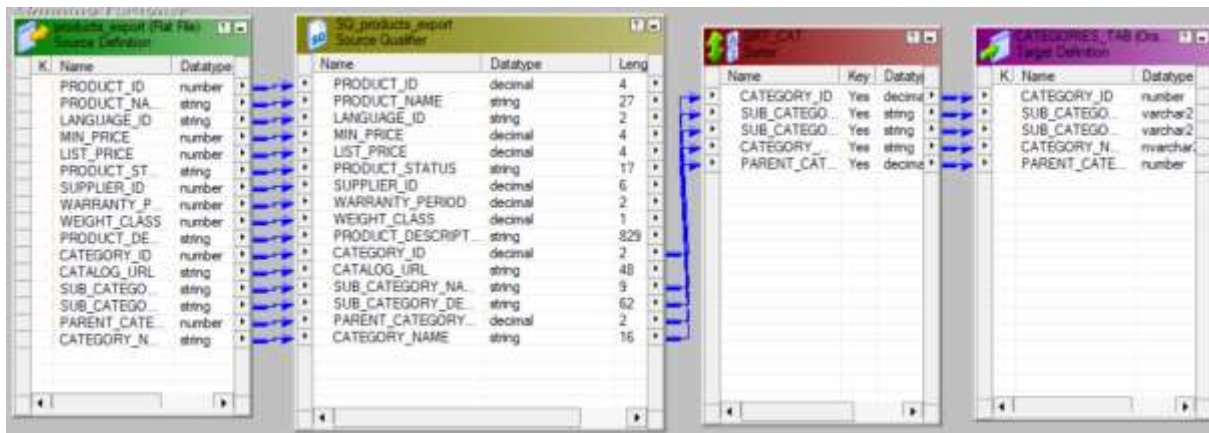


Fig 8. CATEGORIES_TAB MAPPING

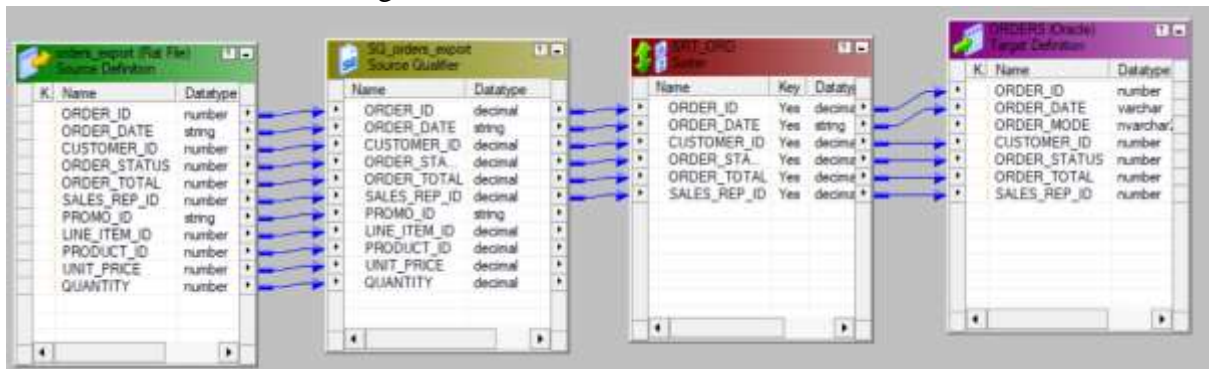


Fig 9. ORDERS MAPPING

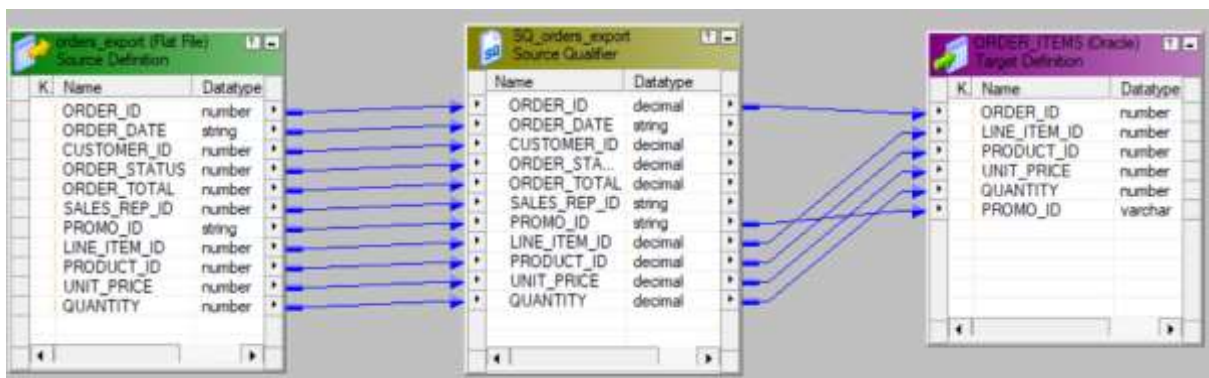


Fig 10. ORDER_ITEMS MAPPING

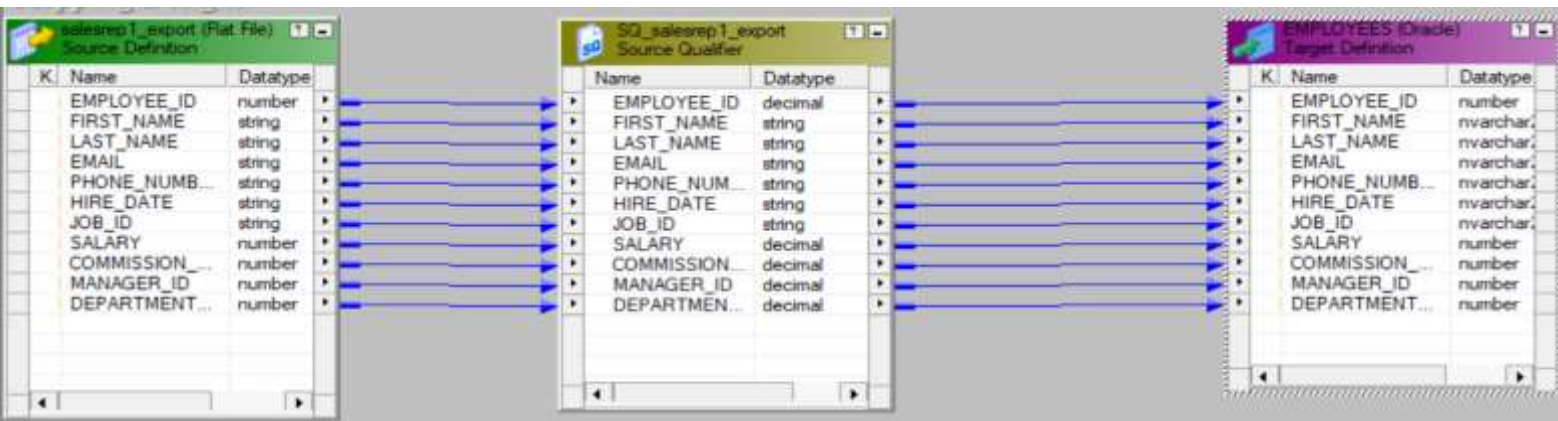


Fig 11. EMPLOYEES MAPPING

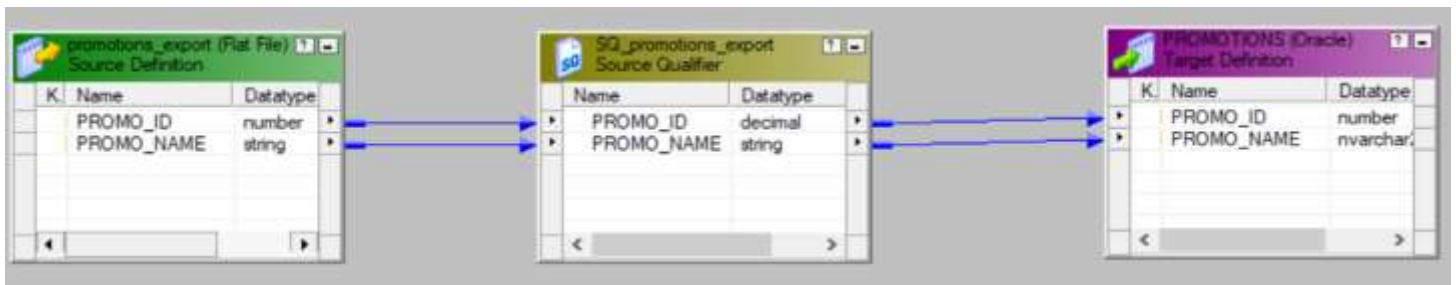


Fig 12. PROMOTIONS MAPPING

```
SQL> desc STG_DATE;
Name                               Null?    Type
-----
CALENDAR_DATE                      DATE

SQL>

SQL> INSERT INTO STG_DATE(CALENDAR_DATE) SELECT to_date('01-JAN-2000','DD-MON-YYYY') + level-1 calendar_date
2 FROM dual connect by level <=
3 (
4   to_date('31-DEC-2025','DD-MON-YYYY') -
5   to_date('01-JAN-2000','DD-MON-YYYY') + 1
6 );

9497 rows created.

SQL> select count(*) from stg_date;

COUNT(*)
-----
9497

SQL>
```

Fig 13. STG_DATE TABLE CREATION IN STAGING DATABASE

A table with date values is created for using as a source to create a date dimension table in data warehouse.



Fig 14. WORKFLOW FOR STAGING DATABASE

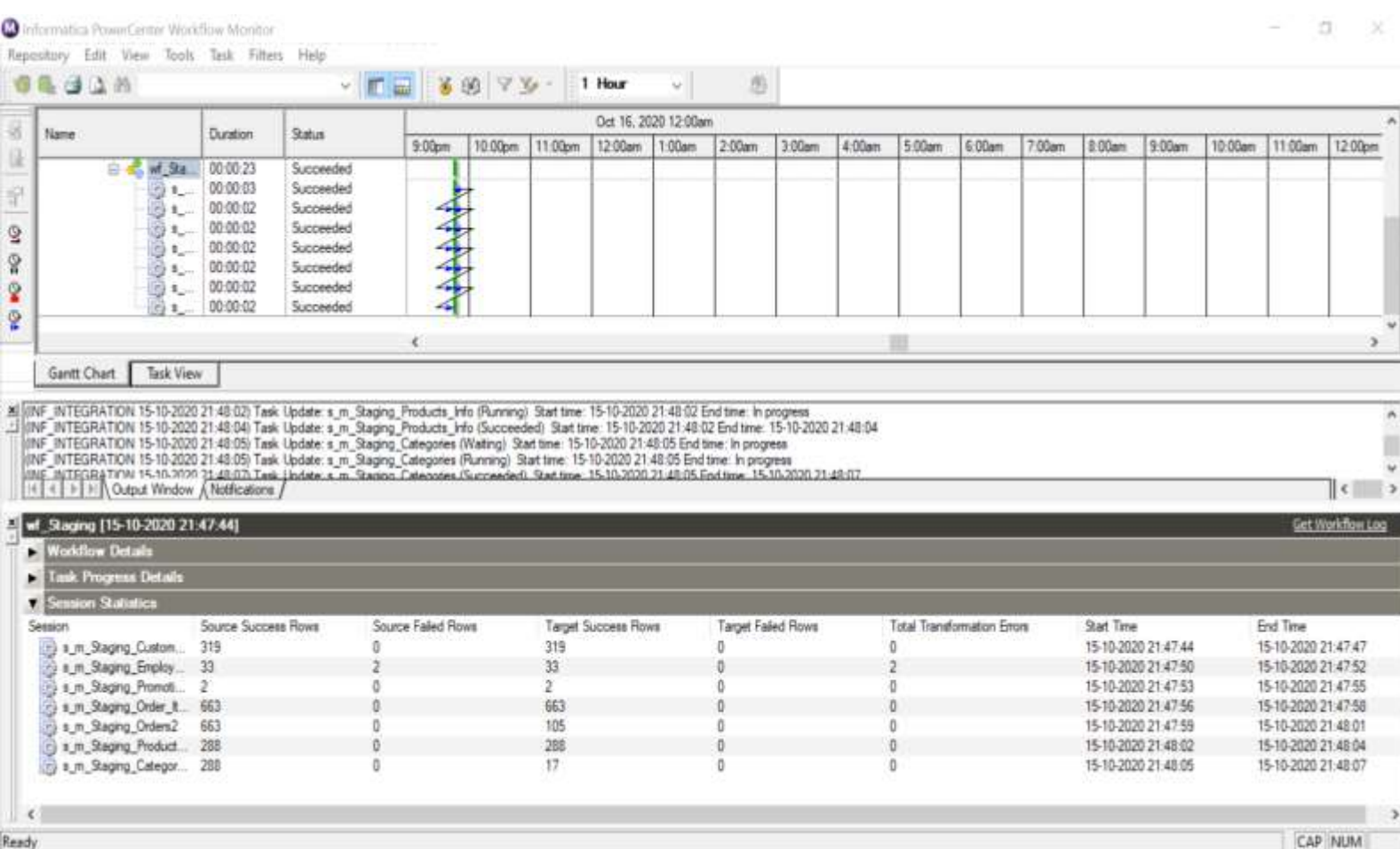


Fig 15. WORKFLOW MONITOR OF STAGING DATABASE

From the above picture of workflow monitor the number of rows in source and target are visible.

	Source Success Rows	Target Success Rows
customer	319	319
salesrep	33	33
promotions	2	2
orders	663	105
order_items	663	663
product_information	288	288
categories_tab	288	17

Here it can be seen that the number of target success rows are lesser for the ORDERS and CATEGORIES_TAB tables. This is because we are passing only distinct values to these tables from source. Because in staging database we want to remove the redundancy from the source.

From the source data it can be seen that the fields included in ORDERS and CATEGORIES_TAB tables are getting repeated. So we need to normalize the data to reduce redundancy.

Here from complete orders details data is divided into two tables ORDER_ITEMS and ORDERS. All repeating value fields are added to the ORDERS table and we load only distinct values into this table. The same happens in the case of product details, data is divided into two tables PRODUCT_INFORMATION and CATEGORIES_TAB and only distinct values get loaded to CATEGORIES_TAB table.

```
SQL> select count(*) from customers;
COUNT(*)
-----
319
SQL> select count(*) from employees;
COUNT(*)
-----
33
SQL> select count(*) from product_information;
COUNT(*)
-----
288
SQL> select count(*) from categories_tab;
COUNT(*)
-----
17
SQL> select count(*) from orders;
COUNT(*)
-----
105
SQL> select count(*) from order_items;
COUNT(*)
-----
663
SQL> select count(*) from promotions;
COUNT(*)
-----
2
```

Fig 16. COUNT IN STAGING DATABASE

2. FROM STAGING DATABASE TO DATA WAREHOUSE

Staging Table	DWH Table
customers	customer_dim_scd2
employees	salesrep_dim_scd2
product_information & categories_tab	products_dim
promotions	promotions_dim
orders & order_items	sales_fact
stg_date	date_dim

In data warehouse customer and salesrep tables are done in SCD2 and products and promotions tables are done in SCD1. Date_dim table is directly mapped with some extra fields created from an expression table. Sales_fact table is created by taking in fields from orders and order_items table along with primary keys from other dimension tables.

The screenshot displays the Source Analyzer tool with several tables open, showing their metadata. The tables and their columns are as follows:

K	Name	Datatype	L
	CUSTOMER_ID	number	15
	CUST_FIRST	varchar2	25
	CUST_LAST_N	varchar2	25
	STREET_ADD	varchar2	50
	POSTAL_CODE	number	15
	CITY	varchar2	25
	STATE_PROVI	varchar2	25
	COUNTRY_ID	varchar2	5
	COUNTRY_NA	varchar2	50
	REGION_ID	number	15
	NLS_LANGUAGE	varchar2	25
	NLS_TERRITO	varchar2	25
	CREDIT_LIMIT	number	15
	CUST_EMAIL	varchar2	60
	PRIMARY_PHO	varchar2	25
	PHONE_NUMB	varchar2	25
	ACCOUNT_MG	number	15
	LOCATION_GT	number	15
	LOCATION_SRID	number	15
	LOCATION_X	number	15
	LOCATION_Y	number	15

K	Name	Datatype	L
	EMPLOYEE_ID	number	15
	FIRST_NAME	nvarchar2	25
	LAST_NAME	nvarchar2	25
	EMAIL	nvarchar2	60
	PHONE_NUMBER	nvarchar2	25
	HIRE_DATE	nvarchar2	15
	JOB_ID	nvarchar2	10
	SALARY	number	15
	COMMISSION_PCT	number	15
	MANAGER_ID	number	15
	DEPARTMENT_ID	number	15

K	Name	Datatype	Length/Precision
	PRODUCT_ID	number	15
	PRODUCT_NAME	varchar2	50
	LANGUAGE_ID	nvarchar2	10
	PRODUCT_DESCRIPTION	varchar2	1000
	CATEGORY_ID	number	15
	WEIGHT_CLASS	number	15
	WARRANTY_PERIOD	number	15
	SUPPLIER_ID	number	15
	PRODUCT_STATUS	varchar2	30
	LIST_PRICE	number	15
	MIN_PRICE	number	15
	CATALOG_URL	varchar2	100

K	Name	Datatype	Length/Precision
	CATEGORY_ID	number	15
	SUB_CATEGORY_NAME	nvarchar2	20
	SUB_CATEGORY_DESC	nvarchar2	100
	CATEGORY_NAME	varchar2	50
	PARENT_CATEGORY_ID	number	15

K	Name	Datatype	L
	ORDER_ID	number	15
	ORDER_DATE	nvarchar2	19
	ORDER_MODE	varchar2	25
	CUSTOMER_ID	number	15
	ORDER_STATUS	number	15
	ORDER_TOTAL	number	15
	SALES_REP_ID	number	15

K	Name	Datatype	L
	PROMO_ID	number	15
	PROMO_NAME	nvarchar2	30

K	Name	Datatype	L
	ORDER_ID	number	15
	LINE_ITEM_ID	number	15
	PRODUCT_ID	number	15
	UNIT_PRICE	number	15
	QUANTITY	number	15
	PROMO_ID	number	15

K	Name	Datatype	L
	CALENDAR_DA	date	19

Fig 17. Staging Source Metadata

CUSTOMER_DIM_SCD2 (Oracle)			
K	Name	Datatype	Length/P
	CUSTOMER_DIM_ID	number	15
	CUSTOMER_ID	number	15
	CUST_FIRST_NAME	nvarchar2	25
	CUST_LAST_NAME	nvarchar2	25
	STREET_ADDRESS	nvarchar2	50
	POSTAL_CODE	number	15
	CITY	nvarchar2	25
	STATE_PROVINCE	nvarchar2	25
	COUNTRY_ID	nvarchar2	5
	COUNTRY_NAME	nvarchar2	50
	REGION_ID	number	15
	NLS_LANGUAGE	nvarchar2	25
	NLS_TERRITORY	nvarchar2	25
	CREDIT_LIMIT	number	15
	CUST_EMAIL	nvarchar2	60
	PRIMARY_PHONE_N	nvarchar2	25
	PHONE_NUMBER_2	nvarchar2	25
	ACCOUNT_MGR_ID	number	15
	LOCATION_GTYPE	number	15
	LOCATION_SRID	number	15
	LOCATION_X	number	15
	LOCATION_Y	number	15
	START_DATE	timestamp	26
	END_DATE	timestamp	26
	VERSION	number	15
	FLAG	nvarchar2	10
	INSERTED_DATE	timestamp	26
	INSERTED_BY	nvarchar2	20
	UPDATED_DATE	timestamp	26
	UPDATED_BY	nvarchar2	20

SALESREP_DIM_SCD2 (Oracle)			
K	Name	Datatype	Length/Precision
	SALESREP_DIM_ID	number	15
	EMPLOYEE_ID	number	15
	FIRST_NAME	nvarchar2	25
	LAST_NAME	nvarchar2	25
	EMAIL	nvarchar2	60
	PHONE_NUMBER	nvarchar2	25
	HIRE_DATE	nvarchar2	15
	JOB_ID	nvarchar2	10
	SALARY	number	15
	COMMISSION_PCT	number	15
	MANAGER_ID	number	15
	DEPARTMENT_ID	number	15
	START_DATE	timestamp	26
	END_DATE	timestamp	26
	VERSION	number	15
	FLAG	nvarchar2	10
	INSERTED_DATE	timestamp	26
	INSERTED_BY	nvarchar2	20
	UPDATED_DATE	timestamp	26
	UPDATED_BY	nvarchar2	20

PRODUCTS_DIM (Oracle)			
K	Name	Datatype	Leng...
	PRODUCT_DIM_ID	number	15
	PRODUCT_ID	number	15
	PRODUCT_NAME	nvarchar2	50
	LANGUAGE_ID	nvarchar2	2
	MIN_PRICE	number	15
	LIST_PRICE	number	15
	PRODUCT_STATUS	nvarchar2	30
	SUPPLIER_ID	number	15
	WARRANTY_PERIOD	number	15
	WEIGHT_CLASS	number	15
	PRODUCT_DESCRIPTION	nvarchar2	1000
	CATEGORY_ID	number	15
	CATALOG_URL	nvarchar2	100
	SUB_CATEGORY_NAME	nvarchar2	25
	SUB_CATEGORY_DESCRI...	nvarchar2	1000
	PARENT_CATEGORY_ID	number	15
	CATEGORY_NAME	nvarchar2	25
	INSERTED_DATE	timestamp	26
	INSERTED_BY	nvarchar2	15
	UPDATED_DATE	timestamp	26
	UPDATED_BY	nvarchar2	15

PROMOTIONS_DIM (Oracle)			
K	Name	Datatype	Length
	PROMOTION_DIM_ID	number	15
	PROMO_ID	number	15
	PROMO_NAME	nvarchar2	30
	INSERTED_DATE	timestamp	26
	INSERTED_BY	nvarchar2	15
	UPDATED_DATE	timestamp	26
	UPDATED_BY	nvarchar2	15

SALES_FACT (Oracle)			
K	Name	Datatype	Length...
	PK_SALES_FACT_KEY	number	15
	ORDER_ID	number	15
	CUSTOMER_DIM_ID	number	15
	SALESREP_DIM_ID	number	15
	PRODUCT_DIM_ID	number	15
	PROMOTION_DIM_ID	number	15
	DATE_DIM_ID	number	15
	DOLLARS_SOLD	number	15
	QUANTITY_SOLD	number	15
	INSERT_DATE	timestamp	26
	INSERT_BY	nvarchar2	20
	UPDATE_DATE	timestamp	26
	UPDATE_BY	nvarchar2	30

DATE_DIM (Oracle)			
K	Name	Datatype	Length/Pr...
	DATE_DIM_ID	number	15
	SALES_DATE	date	19
	SALES_DAY_OF_THE_YEAR	number	15
	SALES_MONTH	number	15
	SALES_YEAR	number	15
	SALES_QUARTER	number	15
	SALES_MONTH_NAME	nvarchar2	25

Fig 18. Data Warehouse Target Metadata

TYPES OF SLOWLY CHANGING DIMENSIONS IMPLEMENTED FOR DIMENSION TABLES

- 1) SLOWLY CHANGING DIMENSION TYPE 1 (SCD1)
- 2) SLOWLY CHANGING DIMENSION TYPE 2 (SCD2)

1. SLOWLY CHANGING DIMENSION TYPE 1

In SCD1 dimension table only current values are stored there is no record of previous values i.e, only new rows are inserted and updates are done on the existing rows itself.

In this case study PRODUCT_DIM and PROMOTIONS_DIM tables are implemented in SCD1.

There are some columns which we add extra in the target tables of data warehouse(can be seen in the product target metadata Fig 18.). These are for better analyzing the data.

INSERTED_DATE	date/time	29	9	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	SYSDATE
INSERTED_BY	string	20	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$PMRepositoryUserName
UPDATED_DATE	date/time	29	9	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	SYSDATE
UPDATED_BY	string	20	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$PMRepositoryUserName
FLAG	string	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	IIF(ISNULL(PROMOTION...

Fig 19. Extra columns in expression table of a SCD1 mapping

2. SLOWLY CHANGING DIMENSION TYPE 2

In SCD2 dimension table we keep the current value as well as the historical value in the data warehouse. So we add two columns to keep track of the history called VERSION AND FLAG. We also have all columns except FLAG from Fig 19. We also add two columns called START_DATE and END_DATE which helps to keep track of history.

(All the above mentioned columns can be seen in customer target metadata in Fig 18.)

We also use some columns in the expression table for various purposes about which we will see in detail.

START_DATE	date/time	29	9	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	SYSDATE
LKP_CHECKSUM	string	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	MD5(TO_CHAR(LKP_CU...
SRC_CHECKSUM	string	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	MD5(TO_CHAR(SRC_CU...
INS_UPD_FLAG	string	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	IIF(ISNULL(LKP_CUSTO...
ACTIVE_FLAG	string	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'Y'
INACTIVE_FLAG	string	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'N'
VERSION	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1
REINSERT_VERSION	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	LKP_VERSION+1
INSERTED_DATE	date/time	29	9	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	SYSDATE
INSERTED_BY	string	20	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$PMRepositoryUserName
UPDATED_DATE	date/time	29	9	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	SYSDATE
UPDATED_BY	string	20	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$PMRepositoryUserName

Fig 20. Extra columns in expression table of a SCD2 mapping

START_DATE

Denotes the start date of the record.

LKP_CHECKSUM

This field is made as a variable. It uses MD5() hash function which takes in a string of any length and returns a 128 bit hashed value. We pass in the fields from lookup table into this function for generating a unique value for that particular record.

SRC_CHECKSUM

Same function as of LKP_CHECKSUM here instead of lookup fields we pass in source fields to generate the unique value.

INS_UPD_FLAG

This field determines whether the row from source should be treated as a new row or as an update to an existing row. First it checks whether the primary key from lookup is null or not.

If it is null then field is set to 'TRUE' if not null and the
LKP_CHECKSUM != SRC_CHECKSUM then field is set to 'FALSE'.

ACTIVE_FLAG

Value is set to 'Y'. Used to map to the final FLAG in target table. FLAG='Y' if it is the latest version.

INACTIVE_FLAG

Value is set to 'N'. Used to map to the final FLAG in target table. FLAG='N' if it is not the latest version.

VERSION

Field is always set to 1. Used to map to the VERSION field of target if the row is new.

REINSERT_VERSION

Field is VERSION+1. Used to map to the VERSION field of target if the row is getting updated.

INSERTED_DATE

Denotes the inserted date of the row into the data warehouse.

INSERTED_BY

Denotes the name of the person or entity who inserted the row into the data warehouse.

UPDATED_DATE

Denotes the updated date of the row in the data warehouse.

UPDATED_BY

Denotes the name of the person or entity who updated the row in the data warehouse.

MAPPINGS FROM STAGING DATABASE TO DATA WAREHOUSE

1. CUSTOMER_DIM_SCD2

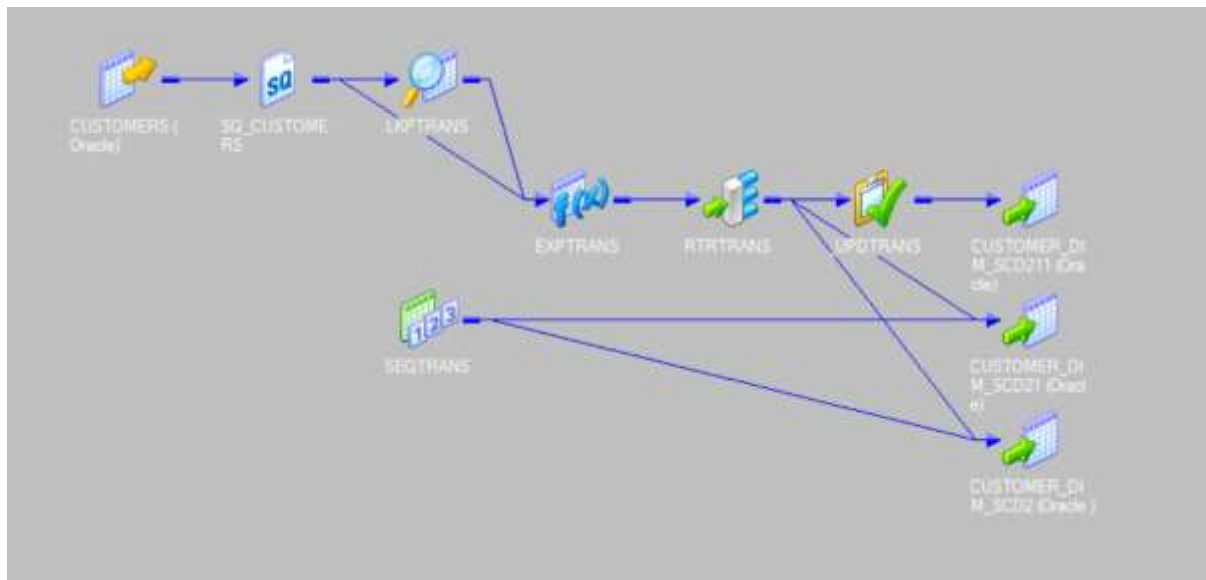


Fig 21. CUSTOMER_DIM_SCD2 MAPPING

Process Flow

The input is from CUSTOMERS table of staging database. The business key from source, here it is the CUSTOMER_ID is passed to the lookup table. The lookup object is CUSTOMER_DIM_SCD2 table in the data warehouse. In lookup it checks whether the CUSTOMER_ID from source matches with CUSTOMER_ID in target. In lookup SQL Override is done to select the primary key only with FLAG='Y' (denotes latest version of the record). The primary key CUSTOMER_DIM_ID is connected from lookup to the next transformation which is an expression. Inside expression transformation we add as well as compute all fields shown in Fig 20. From expression, fields are mapped to next transformation which is a router transformation. Inside router two groups (INSERT,

UPDATE) are created. In INSERT group the condition is INS_UPD_FLAG='TRUE' and in UPDATE group the condition is INS_UPD_FLAG='FALSE'. A sequence generator transformation is used to create the running sequence needed for the primary key.

From INSERT group of router, fields are mapped to the target table. The primary key will be from sequence generator.

From UPDATE group of router, fields are mapped to the target table. Here also the primary key is from sequence generator the change is REINSERT_VERSION is mapped to the VERSION of target table. This inserts the update in a new row with different version and flag.

From UPDATE group of router, only CUSTOMER_DIM_ID, INACTIVE_FLAG, UPDATED_DATE, UPDATED_BY are mapped to an update strategy which are then mapped to the target table accordingly. This is for changing the old records flag and updated columns.

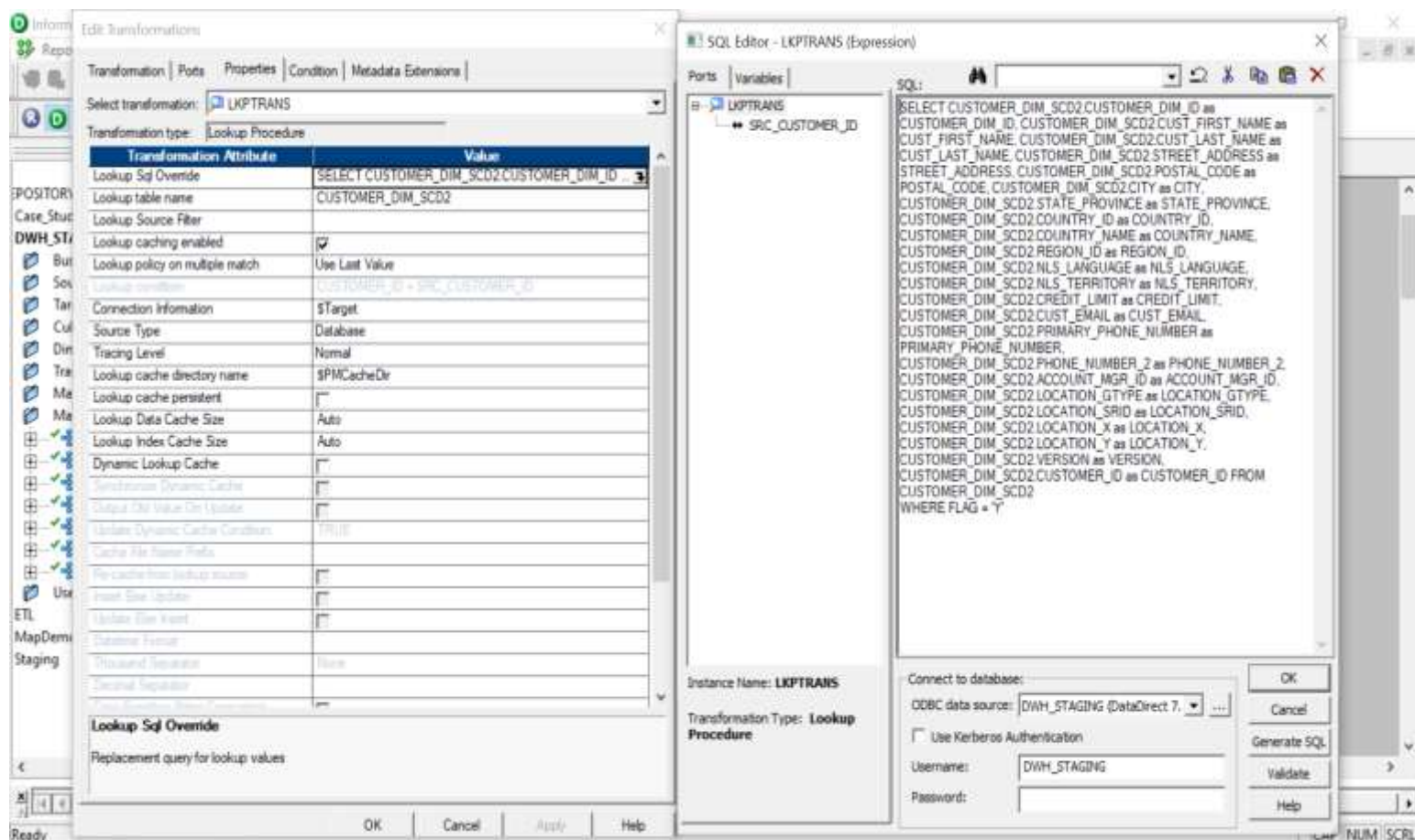


Fig 22. Lookup SQL OVERRIDE in CUSTOMER MAPPING

```

MD5(TO_CHAR(LKP_CUSTOMER_ID) ||
LKP_CUST_FIRST_NAME || LKP_CUST_LAST_NAME ||
LKP_STREET_ADDRESS || TO_CHAR
(LKP_POSTAL_CODE) || LKP_CITY ||
LKP_STATE_PROVINCE || LKP_COUNTRY_ID ||
LKP_COUNTRY_NAME || TO_CHAR(LKP_REGION_ID) ||
LKP_NLS_LANGUAGE || LKP_NLS_TERRITORY ||
TO_CHAR(LKP_CREDIT_LIMIT) || LKP_CUST_EMAIL ||
LKP_PRIMARY_PHONE_NUMBER ||
LKP_PHONE_NUMBER_2 || TO_CHAR
(LKP_ACCOUNT_MGR_ID) || TO_CHAR
(LKP_LOCATION_GTYPE) || TO_CHAR
(LKP_LOCATION_SRID) || TO_CHAR(LKP_LOCATION_X) ||
TO_CHAR(LKP_LOCATION_Y))

```

Fig 23. LKP_CHECKSUM for CUSTOMER MAPPING

```

MD5(TO_CHAR(SRC_CUSTOMER_ID) ||
SRC_CUST_FIRST_NAME || SRC_CUST_LAST_NAME ||
SRC_STREET_ADDRESS || TO_CHAR
(SRC_POSTAL_CODE) || SRC_CITY ||
SRC_STATE_PROVINCE || SRC_COUNTRY_ID ||
SRC_COUNTRY_NAME || TO_CHAR(SRC_REGION_ID) ||
SRC_NLS_LANGUAGE || SRC_NLS_TERRITORY ||
TO_CHAR(SRC_CREDIT_LIMIT) || SRC_CUST_EMAIL ||
SRC_PRIMARY_PHONE_NUMBER ||
SRC_PHONE_NUMBER_2 || TO_CHAR
(SRC_ACCOUNT_MGR_ID) || TO_CHAR
(SRC_LOCATION_GTYPE) || TO_CHAR
(SRC_LOCATION_SRID) || TO_CHAR(SRC_LOCATION_X) ||
TO_CHAR(SRC_LOCATION_Y))

```

Fig 24. SRC_CHECKSUM for CUSTOMER MAPPING

```

IIF(ISNULL(LKP_CUSTOMER_DIM_ID),'TRUE',IIF(NOT
ISNULL(LKP_CUSTOMER_DIM_ID) AND LKP_CHECKSUM
<>SRC_CHECKSUM,'FALSE'))

```

Fig 25. INS_UPD_FLAG for CUSTOMERS MAPPING

Testing the CUSTOMER_DIM_SCD2 table

For testing the working of SCD2 implemented, in the source file (customer_export.csv)

the last row is updated and a new row is also added at the end. Below is the screen shot of last few rows of source files before updating.

317	627 Sivaji	Gielgud	1667 2010	61311 Batavia	Ker	IN	India	3 hi	INDIA	500 Sivaji.Gielg	+91 80 012 4931	148
318	715 Malcolm	Field	Piazza Sviz	361187 Roma		IT	Italy	1 i	ITALY	2400 Malcolm.F	+39 6 012 +39 6 083	147
319	727 Margaret	Ustinov	Via Dello C	361193 Roma		IT	Italy	1 i	ITALY	1200 Margaret.	+39 6 012 4531	147
320	755 Kevin	Cleveland	Via Notori	361235 Ventimiglia		IT	Italy	1 i	ITALY	700 Kevin.Clev	+39 10 012 4387	147
321												

Fig 26. CUSTOMER_EXPORT.CSV BEFORE UPDATING

317	627 Sivaji	Gielgud	1667 2010	61311 Batavia	Ker	IN	India	3 hi	INDIA	500 Sivaji.Gielg	+91 80 012 4931	148
318	715 Malcolm	Field	Piazza Sviz	361187 Roma		IT	Italy	1 i	ITALY	2400 Malcolm.F	+39 6 012 4507 +39 6 083	147
319	727 Margaret	Ustinov	Via Dello C	361193 Roma		IT	Italy	1 i	ITALY	1200 Margaret.	+39 6 012 4531	147
320	755 Kevin	123 Cleveland	Via Notori	361235 Ventimiglia		IT	Italy	1 i	ITALY	700 Kevin.Clev	+39 10 012 4387	147
321	999 Akash	J	ABCD	678678 SDFSS		YY	SDFP	2 hi	DSFS	786 ASDA	67567	567
322												

Fig 27. CUSTOMER_EXPORT.CSV AFTER UPDATING

Now after loading this data to the staging database and then loading that to the data warehouse the SCD2 was working as expected. Below image is taken from SQL DEVELOPER by connecting to the data warehouse.

315	1645	607 Sharmila	Fonda	1649 Anamika St		361168 Cochin	Ker	IN	India	3 hi	INDIA	500 Sharm
316	1646	627 Sivaji	Gielgud	1667 2010 St		61311 Batavia	Ker	IN	India	3 hi	INDIA	500 Sivaji
317	1647	715 Malcolm	Field	Piazza Svinpera		361187 Roma	(null)	IT	Italy	1 i	ITALY	2400 Malco
318	1648	727 Margaret	Ustinov	Via Dello Croce 93		361193 Roma	(null)	IT	Italy	1 i	ITALY	1200 Marga
319	1649	755 Kevin	Cleveland	Via Notoriosa 1943		361235 Ventimiglia	(null)	IT	Italy	1 i	ITALY	700 Kevin
320	1650	755 Kevin123	Cleveland	Via Notoriosa 1943		361235 Ventimiglia	(null)	IT	Italy	1 i	ITALY	700 Kevin
321	1651	999 Akash	J	ABCD		678678 SDFSS	(null)	YY	SDFP	2 hi	DSFS	786 ASDA

Fig 28. LAST ROWS OF CUSTOMER_DIM_SCD2 TABLE IN DATA WAREHOUSE

In CUSTOMER_DIM_SCD2 as SCD2 was implemented, the table got inserted with 2 rows one was the update made to the last row other was a completely new record(new row added at the end in source).

2. SALESREP_DIM_SCD2

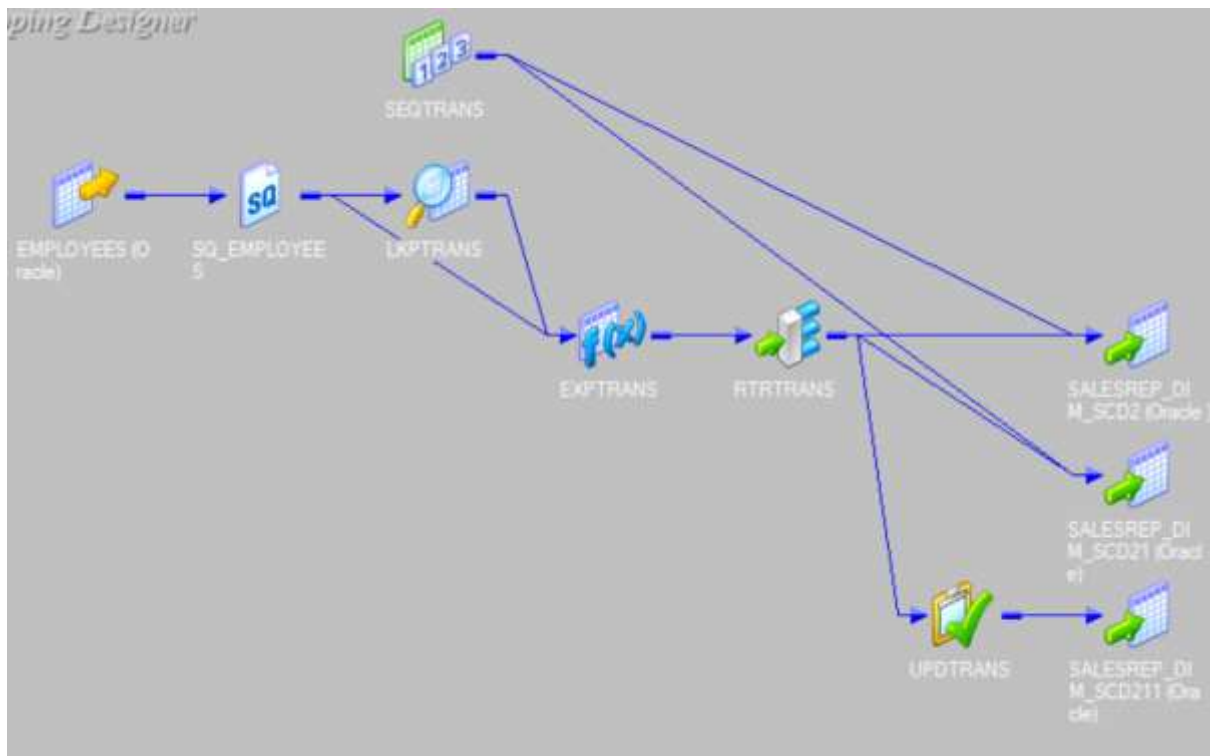


Fig 29. SALESREP MAPPING

Process Flow

The input is from EMPLOYEES table of staging database. The business key from source, here it is the EMAIL is passed to the lookup table. The lookup object is SALESREP_DIM_SCD2 table in the data warehouse. In lookup it checks whether the EMAIL from source matches with EMAIL in target. In lookup SQL Override is done to select the primary key only with FLAG='Y' (denotes latest version of the record). The primary key SALESREP_DIM_ID is connected from lookup to the next transformation which is an expression. Inside expression transformation we add as well as compute all fields shown in Fig 20. From expression, fields are mapped to next transformation which is a router transformation. Inside router two groups (INSERT, UPDATE) are created. In INSERT group the condition is INS_UPD_FLAG='TRUE' and in UPDATE group the condition is INS_UPD_FLAG='FALSE'. A sequence generator transformation is used to create the running sequence needed for the primary key.

From INSERT group of router, fields are mapped to the target table. The primary key will be from sequence generator.

From UPDATE group of router, fields are mapped to the target table. Here also the primary key is from sequence generator the change is REINSERT_VERSION is mapped to the VERSION of target table. This inserts the update in a new row with different version and flag.

From UPDATE group of router, only SALESREP_DIM_ID, INACTIVE_FLAG, UPDATED_DATE, UPDATED_BY are mapped to an update strategy which are then mapped to the target table accordingly. This is for changing the old records flag and updated columns.

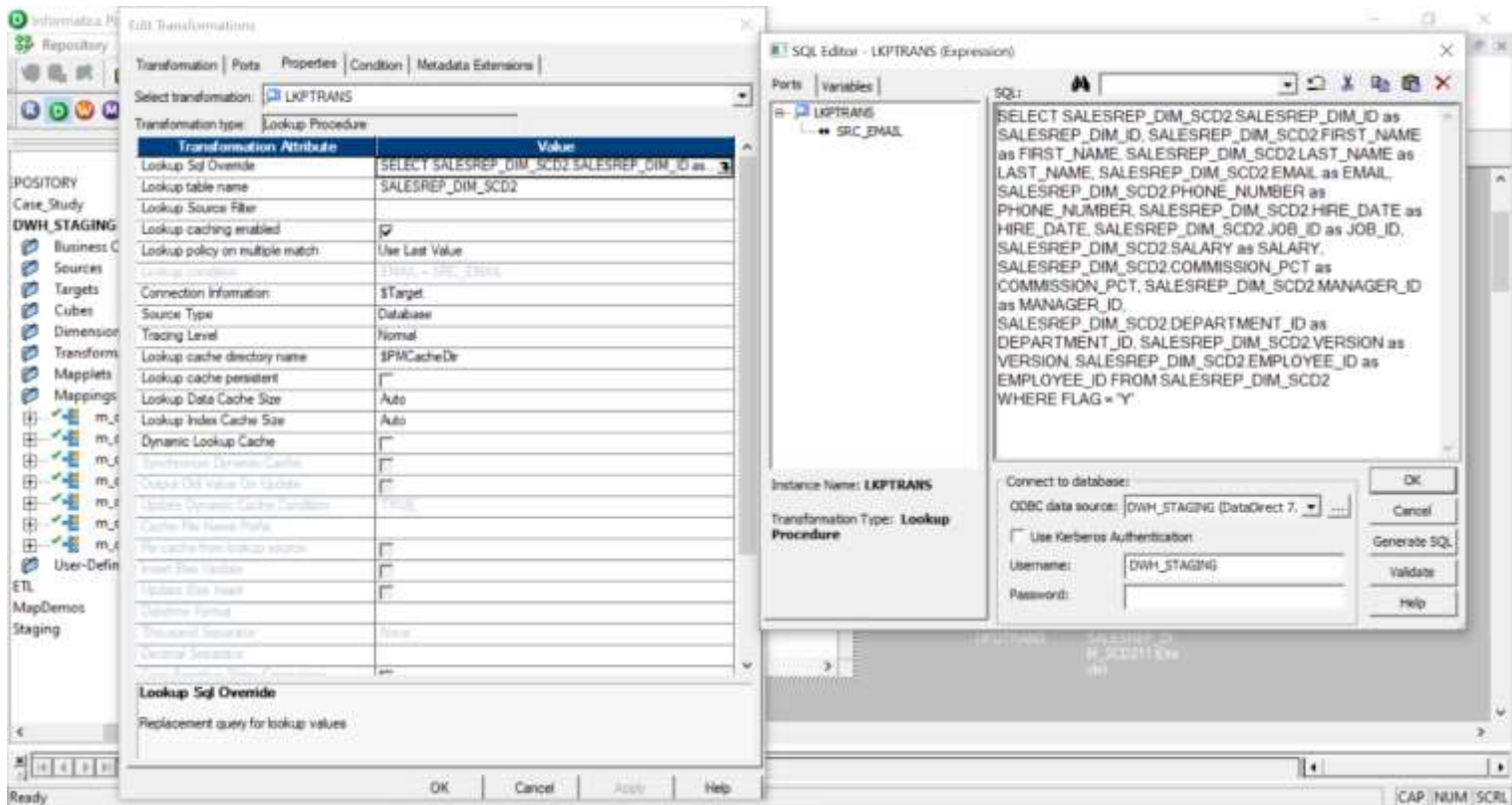


Fig 30. SALESREP LOOKUP SQL OVERRIDE

```
MD5(TO_CHAR(LKP_EMPLOYEE_ID) || LKP_FIRST_NAME
|| LKP_LAST_NAME || LKP_EMAIL || LKP_PHONE_NUMBER ||
LKP_HIRE_DATE || LKP_JOB_ID || TO_CHAR
(LKP_SALARY) || TO_CHAR(LKP_COMMISSION_PCT) ||
TO_CHAR(LKP_MANAGER_ID) || TO_CHAR
(LKP_DEPARTMENT_ID))
```

Fig 31. SALESREP LKP_CHECKSUM

```
MD5(TO_CHAR(SRC_EMPLOYEE_ID) ||
SRC_FIRST_NAME || SRC_LAST_NAME || SRC_EMAIL ||
SRC_PHONE_NUMBER || SRC_HIRE_DATE || SRC_JOB_ID
|| TO_CHAR(SRC_SALARY) || TO_CHAR
(SRC_COMMISSION_PCT) || TO_CHAR
(SRC_MANAGER_ID) || TO_CHAR
(SRC_DEPARTMENT_ID))
```

Fig 32. SALESREP SRC_CHECKSUM

```
IIF(ISNULL(LKP_SALESREP_DIM_ID),'TRUE',IIF(NOT
ISNULL(LKP_SALESREP_DIM_ID) AND LKP_CHECKSUM<>
SRC_CHECKSUM,'FALSE'))
```

Fig 33. SALESREP INS_UP_FLAG

Testing the SALESREP_DIM_SCD2 table

For testing the working of SCD2 implemented, in the source file (salesrep_export.csv)

the last row is updated and a new row is also added at the end. Below is the screen shot of last few rows of source files before updating.

30	175	Alyssa	Hutton	AHUTTON	011.44.16.#####	SA_REP	8800	0.25	149	80
31	176	Jonathon	Taylor	JTAYLOR	011.44.16.#####	SA_REP	8600	0.2	149	80
32	177	Jack	Livingston	JLIVINGS	011.44.16.#####	SA_REP	8400	0.2	149	80
33	178	Kimberely	Grant	KGRANT	011.44.16.#####	SA_REP	7000	0.15	149	
34	179	Charles	Johnson	CJOHNSO	011.44.16.04-Jan-08	SA_REP	6200	0.1	149	80
35										

Fig 34. SALESREP_EXPORT.CSV BEFORE UPDATING

30	175	Alyssa	Hutton	AHUTTON	011.44.16.#####	SA_REP	8800	0.25	149	80
31	176	Jonathon	Taylor	JTAYLOR	011.44.16.#####	SA_REP	8600	0.2	149	80
32	177	Jack	Livingston	JLIVINGS	011.44.16.#####	SA_REP	8400	0.2	149	80
33	178	Kimberely	Grant	KGRANT	011.44.16.#####	SA_REP	7000	0.15	149	
34	179	Charles12	Johnson	CJOHNSO	011.44.16.04-Jan-08	SA_REP	6200	0.1	149	80
35	200	Akash	J	asdad	234	asd	32	0.8	345	78
36										

Fig 35. SALESREP_EXPORT.CSV AFTER UPDATING

Now after loading this data to the staging database and then loading that to the data warehouse the SCD2 was working as expected. Below image is taken from SQL DEVELOPER by connecting to the data warehouse.

30	307	176	Jonathon Taylor	JTAYLOR	011.44.1644.429365	24-Mar-06	SA_REP	8600	0.2	149	80	15-10-20	10:05:54.000000000	PM	(null)	17	15	
31	308	177	Jack Livingston	JLIVINGS	011.44.1644.429364	13-Apr-06	SA_REP	8400	0.2	149	80	15-10-20	10:05:54.000000000	PM	(null)	17	15	
32	309	178	Kimberely Grant	KGRANT	011.44.1644.429363	14-May-07	SA_REP	7000	0.15	149	(null)	15-10-20	10:05:54.000000000	PM	(null)	17	15	
33	310	179	Charles Johnson	CJOHNSON	011.44.1644.429362	04-Jan-08	SA_REP	6200	0.1	149	80	15-10-20	10:05:54.000000000	PM	15-10-20 11:18:05.000000100	PM	17	15
34	311	179	Charles123 Johnson	CJOHNSON	011.44.1644.429362	04-Jan-08	SA_REP	6200	0.1	149	80	15-10-20	11:18:05.000000000	PM	(null)	17	15	
35	312	200	Akash J	asdad	234	(null)	asd	32	0.8	345	78	15-10-20	11:18:05.000000000	PM	(null)	17	15	

Fig 36. LAST ROWS OF SALESREP_DIM_SCD2 TABLE IN DATA WAREHOUSE

In SALESREP_DIM_SCD2 as SCD2 was implemented, the table got inserted with 2 rows one was the update made to the last row other was a completely new record(new row added at the end in source).

3. PRODUCTS_DIM

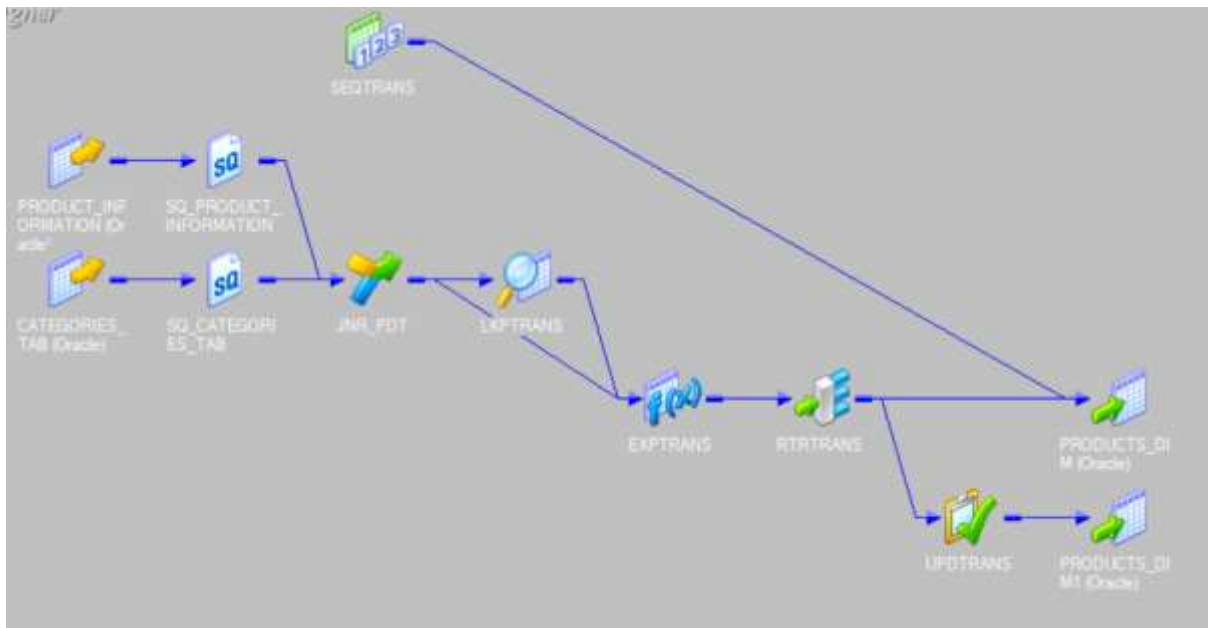


Fig 37. PRODUCTS MAPPING

Process Flow

The input is PRODUCT_INFORMATION and CATEGORIES_TAB tables from the staging database. Both the tables are inner joined on CATEGORY_ID. The resulting table is the source for the mapping. A lookup table with PRODUCTS_DIM as lookup object is created. The PRODUCT_ID which is the business key here is taken into the lookup. In lookup a condition is made to check whether the PRODUCT_ID from source exists in target. The primary key PRODUCT_DIM_ID is then mapped to the expression transformation. In expression transformation all extra fields (Fig 19.) are added. FLAG is set to 'TRUE' or 'FALSE' depending on the primary key.

All fields from the expression transformation is then passed to the router transformation. Inside router two groups (INSERT, UPDATE) are created. In INSERT group the condition is FLAG='TRUE' and in UPDATE group the condition is FLAG='FALSE'. A sequence generator transformation is used to create the running sequence needed for the primary key.

From INSERT group of router, fields are mapped to the target table. The primary key will be from sequence generator.

From UPDATE group of router, fields are mapped to the target table. Here the primary key is from lookup and all fields are mapped from update strategy accordingly.


```
IF(ISNULL(PRODUCT_DIM_ID),'TRUE','FALSE')
```

Fig 38. FLAG IN PRODUCTS MAPPING

Testing the PRODUCT_DIM table

For testing the working of SCD1 implemented, in the source file (product_export.csv) the last row is updated and a new row is also added at the end. Below is the screen shot of last few rows of source files before updating.

285	2470 SPNIX4.0 - US	70	80 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
286	2471 SPNIX3.3 US	439	500 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
287	2492 SPNIX3.3 /US	38	45 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
288	2493 SPNIX3.3 C US	22	25 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
289	2494 SPNIX3.3 I US	20	25 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
290									
291									

Fig 39. PRODUCT_EXPORT.CSV BEFORE UPDATING

286	2471 SPNIX3.3 US	439	500 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
287	2492 SPNIX3.3 US	38	45 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
288	2493 SPNIX3.3 US	22	25 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
289	2494 SPNIX3.3 IN	999	25 orderable	103092	1	1 Operating	24 http://www.software4 operating	20 software	
290	3000 SDF RE	534	32 ASDA	324242	3	3 ASDA	100 SDAD SDAD ASD	66 ASDASD	
291									

Fig 40. PRODUCT_EXPORT.CSV AFTER UPDATING

Now after loading this data to the staging database and then loading that to the data warehouse the SCD1 was working as expected. Below image is taken from SQL DEVELOPER by connecting to the data warehouse.

284	3747	2470 SPNIX4.0 - US	US	70	80 orderable	103092	1	1 Operating System Software: SPNIX V4.0 - Additional network access license.
285	3748	2471 SPNIX3.3 US	US	439	500 orderable	103092	1	1 Operating System Software: SPNIX V3.3 - Base Server License Upgrade to V4.0.
286	3749	2492 SPNIX3.3 /US	US	38	45 orderable	103092	1	1 Operating System Software: SPNIX V3.3 - V4.0 upgrade; class A user.
287	3750	2493 SPNIX3.3 C/DU	US	22	25 orderable	103092	1	1 Operating System Software: SPNIX V3.3 - V4.0 upgrade; class C or D user.
288	3751	2494 SPNIX3.3 IN	IN	999	25 orderable	103092	1	1 Operating System Software: SPNIX V3.3 - V4.0 upgrade; network access license.
289	3752	3000 SDF RE	RE	534	32 ASDA	324242	3	3 ASDA

Fig 41. LAST ROWS OF PRODUCT_DIM TABLE IN DATA WAREHOUSE

4. PROMOTIONS_DIM

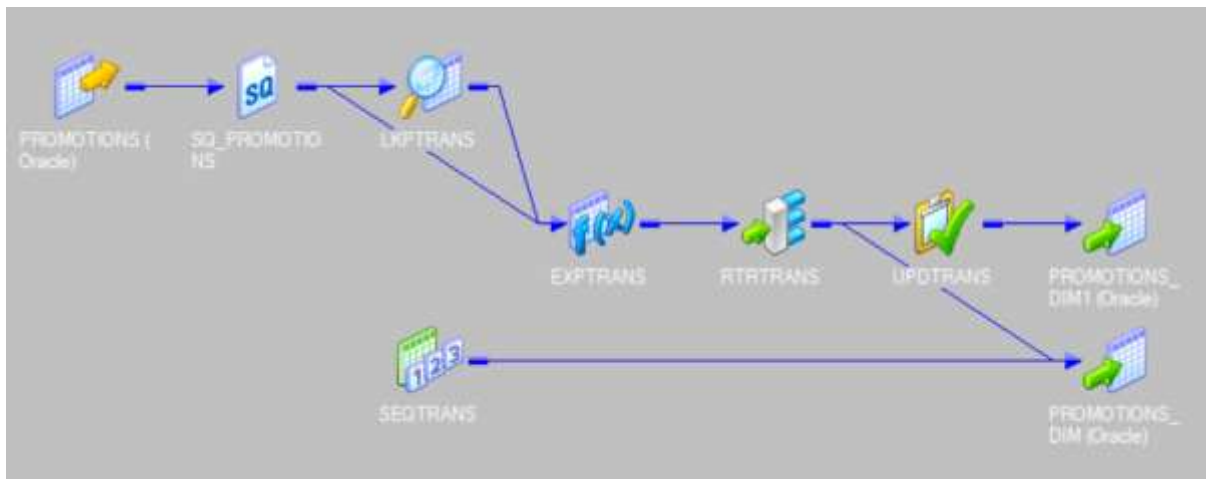


Fig 42. PROMOTIONS MAPPING

Process Flow

The input is PROMOTIONS table from the staging database. A lookup table with PROMOTIONS_DIM as lookup object is created. The PROMO_ID which is the business key here is taken into the lookup. In lookup a condition is made to check whether the PROMO_ID from source exists in target. The primary key PROMOTION_DIM_ID is then mapped to the expression transformation. In expression transformation all extra fields (Fig 19.) are added. FLAG is set to 'TRUE' or 'FALSE' depending on the primary key.

All fields from the expression transformation is then passed to the router transformation. Inside router two groups (INSERT, UPDATE) are created. In INSERT group the condition is FLAG='TRUE' and in UPDATE group the condition is FLAG='FALSE'. A sequence generator transformation is used to create the running sequence needed for the primary key.

From INSERT group of router, fields are mapped to the target table. The primary key will be from sequence generator.

From UPDATE group of router, fields are mapped to the target table. Here the primary key is from lookup and all fields are mapped from update strategy accordingly.

```
IIF(ISNULL(PRODUCT_DIM_ID),'TRUE','FALSE')
```

Fig 43. FLAG IN PROMOTIONS MAPPING

Testing the PROMOTIONS_DIM table

For testing the working of SCD1 implemented, in the source file (promotions_export.csv) the last row is updated and a new row is also added at the end. Below is the screen shot of last few rows of source files before updating.

	A	B	C	D
1	PROMO_ID	PROMO_NAME		
2	1	everyday low price		
3	2	blowout sale		
4				

Fig 44. PROMOTIONS_EXPORT.CSV BEFORE UPDATING

	A	B	
1	PROMO_ID	PROMO_NAME	
2	1	everyday low price	
3	2	blowout sale123	
4	3	HAHA	
5			

Fig 45. PROMOTIONS_EXPORT.CSV AFTER UPDATING

Now after loading this data to the staging database and then loading that to the data warehouse the SCD1 was working as expected. Below image is taken from SQL DEVELOPER by connecting to the data warehouse.

PROMOTION_DIM_ID	PROMO_ID	PROMO_NAME	INSERTED_DATE	INSERTED_BY	UPDATED_DATE	UPDATED_BY
1	23	1 everyday low price	15-10-20 10:06:00.000000000	PM Administrator	15-10-20 11:18:14.000000000	PM Administrator
2	24	2 blowout sale123	15-10-20 10:06:00.000000000	PM Administrator	15-10-20 11:18:14.000000000	PM Administrator
3	25	3 HAHA	15-10-20 11:18:14.000000000	PM Administrator	15-10-20 11:18:14.000000000	PM Administrator

Fig 46. LAST ROWS OF PRODUCT_DIM TABLE IN DATA WAREHOUSE

5. DATE_DIM

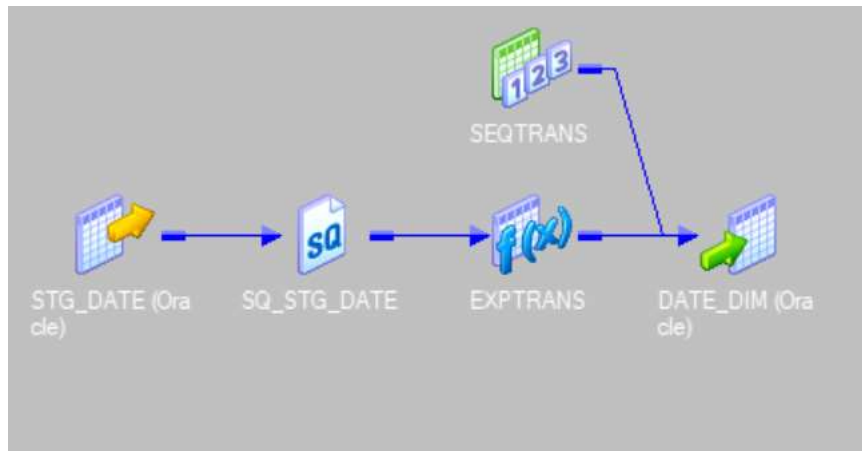


Fig 47. DATE MAPPING

Process Flow

The STG_DATE table in staging database acts as the source. The fields are connected to an expression transformation where six extra columns are added.

Columns added:

1. SALES_DATE
2. SALES_DAY_OF_THE_MONTH
3. SALES_MONTH
4. SALES_YEAR
5. SALES_QUARTER
6. SALES_MONTH_NAME

SALES_DATE is mapped directly to CALENDAR_DATE which is the input field from source.

```
GET_DATE_PART(CALENDAR_DATE,'DD')
```

Fig 48. SALES_DAY_OF_THE_MONTH

```
GET_DATE_PART(CALENDAR_DATE,'MM')
```

Fig 49. SALES_MONTH

```
GET_DATE_PART(CALENDAR_DATE,'YYYY')
```

Fig 50. SALES_YEAR

`TO_INTEGER(TO_CHAR(CALENDAR_DATE,'Q'))`

Fig 51. SALES_QUARTER

`TO_CHAR(CALENDAR_DATE,'MONTH')`

Fig 52. SALES_MONTH_NAME

9483	85459	17-12-25	17	12	2025	4 December
9484	85460	18-12-25	18	12	2025	4 December
9485	85461	19-12-25	19	12	2025	4 December
9486	85462	20-12-25	20	12	2025	4 December
9487	85463	21-12-25	21	12	2025	4 December
9488	85464	22-12-25	22	12	2025	4 December
9489	85465	23-12-25	23	12	2025	4 December
9490	85466	24-12-25	24	12	2025	4 December
9491	85467	25-12-25	25	12	2025	4 December
9492	85468	26-12-25	26	12	2025	4 December
9493	85469	27-12-25	27	12	2025	4 December
9494	85470	28-12-25	28	12	2025	4 December
9495	85471	29-12-25	29	12	2025	4 December
9496	85472	30-12-25	30	12	2025	4 December
9497	85473	31-12-25	31	12	2025	4 December

Fig 53. LAST FEW ROWS OF DATE_DIM TABLE CREATED IN DATA WAREHOUSE

6. SALES_FACT

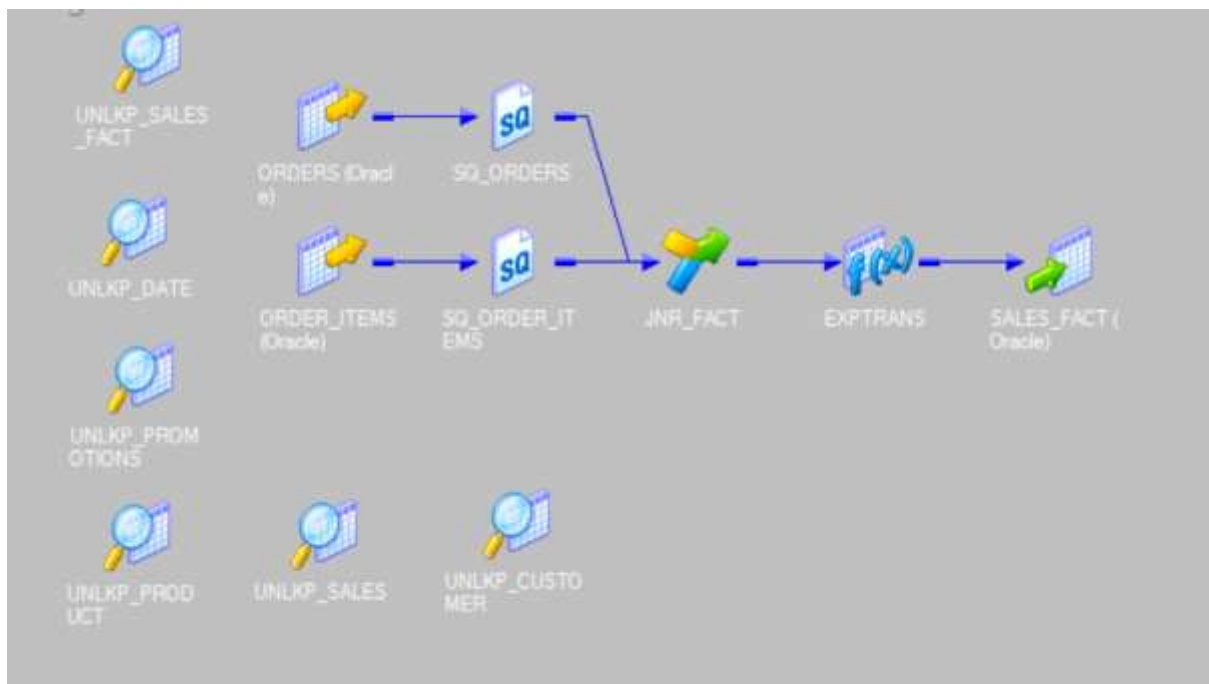


Fig 54. SALES_FACT TABLE MAPPING

Process Flow

Here ORDERS and ORDER_ITEMS tables from the staging database acts as the source.

The tables are inner joined on ORDER_ID. From joiner all fields are passed to an expression transformation. In expression transformation primary keys of other dimension tables are found using their unconnected lookups and added into the fact table. The business keys of the respective dimension tables are passed to their unconnected lookups which in turn returns the corresponding primary key.

Here unconnected lookup is used instead of sequence generator for generating the primary key for fact table. This generation of running sequence using unconnected lookup uses extra fields which are developed in the expression table.

Extra fields used for creation of running sequence primary key for fact table

V_CNT	integer	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	V_CNT+1
V_MAX_VAL	integer	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	IIF(V_CNT=1,,:LKP.UNLK...
V_SEQ	integer	10	0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	IIF(ISNULL(V_MAX_VAL)... 3
O_PK_SALES_FACT_KEY	integer	10	0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	V_SEQ

V_CNT

A value which is used for calculating the running sequence. It is always increasing.

V_MAX_VAL

A value which is used for calculating the running sequence. It will be always 1.

V_SEQ

Denotes the final running sequence generated.

```
SELECT MAX(SALES_FACT.PK_SALES_FACT_KEY) as  
PK_SALES_FACT_KEY FROM SALES_FACT
```

Fig 55. SQL OVERRIDE GIVEN IN THE UNCONNECTD LOOKUP OF SALES_FACT TABLE

```
V_CNT+1
```

Fig 56. VALUE OF V_CNT FIELD IN EXPRESSION

```
IIF(V_CNT=1,:LKP.UNLKP_SALES_FACT(1).IIF(ISNULL
(V_MAX_VAL) OR V_MAX_VAL =0,1,V_MAX_VAL))
```

Fig 57. VALUE OF V_MAX_VAL FIELD IN EXPRESSION

```
IIF(ISNULL(V_MAX_VAL) OR V_MAX_VAL=
0,1,V_CNT+V_MAX_VAL)
```

Fig 58. VALUE OF V_SEQ FIELD IN EXPRESSION

```
SELECT CUSTOMER_DIM_SCD2.CUSTOMER_DIM_ID as
CUSTOMER_DIM_ID,
CUSTOMER_DIM_SCD2.CUSTOMER_ID as CUSTOMER_ID
FROM CUSTOMER_DIM_SCD2
WHERE FLAG = 'Y'
```

Fig 59. SQL OVERRIDE GIVEN IN UNCONNECTED LOOKUP OF CUSTOMER TABLE

```
SELECT SALESREP_DIM_SCD2.SALESREP_DIM_ID as
SALESREP_DIM_ID, SALESREP_DIM_SCD2.EMAIL as
EMAIL FROM SALESREP_DIM_SCD2
WHERE FLAG = 'Y'
```

Fig 60. SQL OVERRIDE GIVEN IN UNCONNECTED LOOKUP OF SALESREP TABLE

Testing the SALES_FACT table

	A	B	C	D	E	F	G	H	I	J	K
1	ORDER_ID	ORDER_D	CUSTOMER_ID	ORDER_SEQ	ORDER_TOTAL	SALES_REP	PROMO_ID	LINE_ITEM	PRODUCT	UNIT_PRICE	QUANTITY
2	2491	25-Oct-08	107	3	31574	160	0	1	3106	46	36
3	2520	#####	146	3	29249.1		0	1	2322	22	22
4	2531	#####	169	8	15760.5	156	1	1	3112	72	5
5	2563	#####	107	3	31574	160	0	1	3114	99	30
6	2601	#####	159	2	69286.4	161	0	1	2986	123	3
7	2615	27-Oct-08	143	3	27132.6		0	1	3187	2.2	25
8	2642	#####	144	6	62303	159	0	1	2311	86.9	5
9	2689	07-Oct-08	101	8	33893.6	161	1	1	2308	54	30
10	2724	#####	169	8	15760.5	156	0	1	3124	84	14
11	2743	#####	107	3	31574	160	0	1	3150	17	45
12	2764	#####	109	1	77727.2	155	0	1	3165	37	71
13	2799	#####	150	4	282694.3		2	1	2308	56	41
14	2824	07-Oct-08	119	9	16447.2		0	1	3163	30	13
15	2859	28-Oct-08	170	9	66816	158	0	1	3167	54	42
16	2876	27-Oct-08	147	3	1636	159	1	1	3197	44	3
17	2899	#####	101	8	33893.6	161	0	1	2264	199.1	15
18	2941	#####	158	0	25270.3	161	2	1	2289	44	15
19	2980	#####	148	10	21116.9		0	1	2365	77	9
20											

Fig 61. ORDERS_EXPORT_NEW.CSV IS ADDED TO STAGING DATABASE ORDERS AND ORDER_ITEMS TABLE FOR TESTING FACT TABLE

The above csv file is loaded to the staging database ORDERS and ORDER_ITEMS table.

From there it is loaded to SALES_FACT table. The count in fact table increased from 663 to 681. Below image is taken from SQL DEVELOPER by connecting to the data warehouse.

668	669	2601	3542	(null)	3336	(null)	79167	369	3 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
669	670	2615	3725	(null)	3300	(null)	79199	55	25 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
670	671	2642	3531	(null)	3419	(null)	79220	434.5	5 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
671	672	2699	3688	(null)	3351	23	79179	1620	30 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
672	673	2714	3554	(null)	3411	(null)	79220	1176	14 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
673	674	2743	3694	(null)	3329	(null)	79217	765	45 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
674	675	2764	3696	(null)	3338	(null)	79165	2627	71 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
675	676	2799	3537	(null)	3351	24	79218	2296	41 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
676	677	2814	3706	(null)	3337	(null)	79179	390	13 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
677	678	2859	3555	(null)	3339	(null)	79200	2268	42 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
678	679	2876	3534	(null)	3438	23	79199	132	3 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
679	680	2899	3688	(null)	3371	(null)	79171	2986.5	15 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
680	681	2941	3545	(null)	3355	24	79164	660	15 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		
681	682	2980	3535	(null)	3292	(null)	79216	693	9 15-10-20 11:18:18.000000000	PM Administrator	15-10-20 11:18:18.000000000	PM Administrator		

Fig 62. LAST FEW ROWS OF FACT TABLE

```
SQL> SELECT SUM(QUANTITY) , SUM(QUANTITY*UNIT_PRICE) FROM ORDER_ITEMS;

SUM(QUANTITY) SUM(QUANTITY*UNIT_PRICE)
-----
30373          3684777.3
```

Fig 63. SUM OF QUANTITY AND (UNIT_PRICE * QUANTITY) FROM ORDER_ITEMS TABLE OF STAGING DATABASE

```
SQL> SELECT SUM(QUANTITY_SOLD), SUM(DOLLARS_SOLD) FROM SALES_FACT;

SUM(QUANTITY_SOLD) SUM(DOLLARS_SOLD)
-----
30373          3684777.3
```

Fig 64. SUM OF QUANTITY AND DOLLARS_SOLD FROM SALES_FACT TABLE

In SALES_FACT table the new records from ORDERS_EXPORT_NEW.CSV was added and a comparison of SUM(QUANTITY) and SUM(UNIT_PRICE * QUANTITY) was made between ORDER_ITEMS and SALES_FACT table to check whether all records were updated in fact table. Both the results were exactly matching proving that the ETL is working.



Fig 65. WORKFLOW OF DATE_DIM TABLE



Fig 66. WORKFLOW OF ALL TABLES EXCEPT DATE_DIM

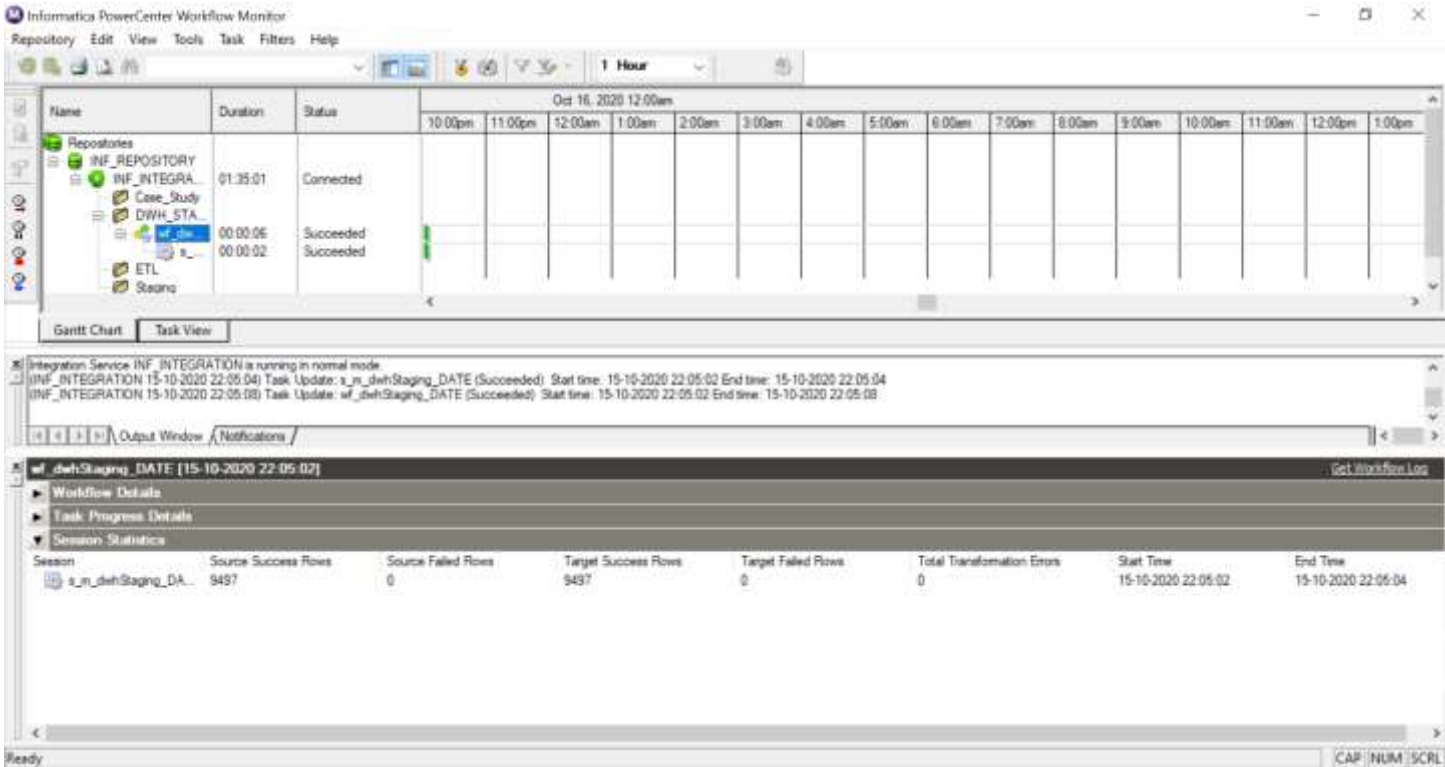


Fig 67. WORKFLOW MONITOR OF DATE_DIM TABLE

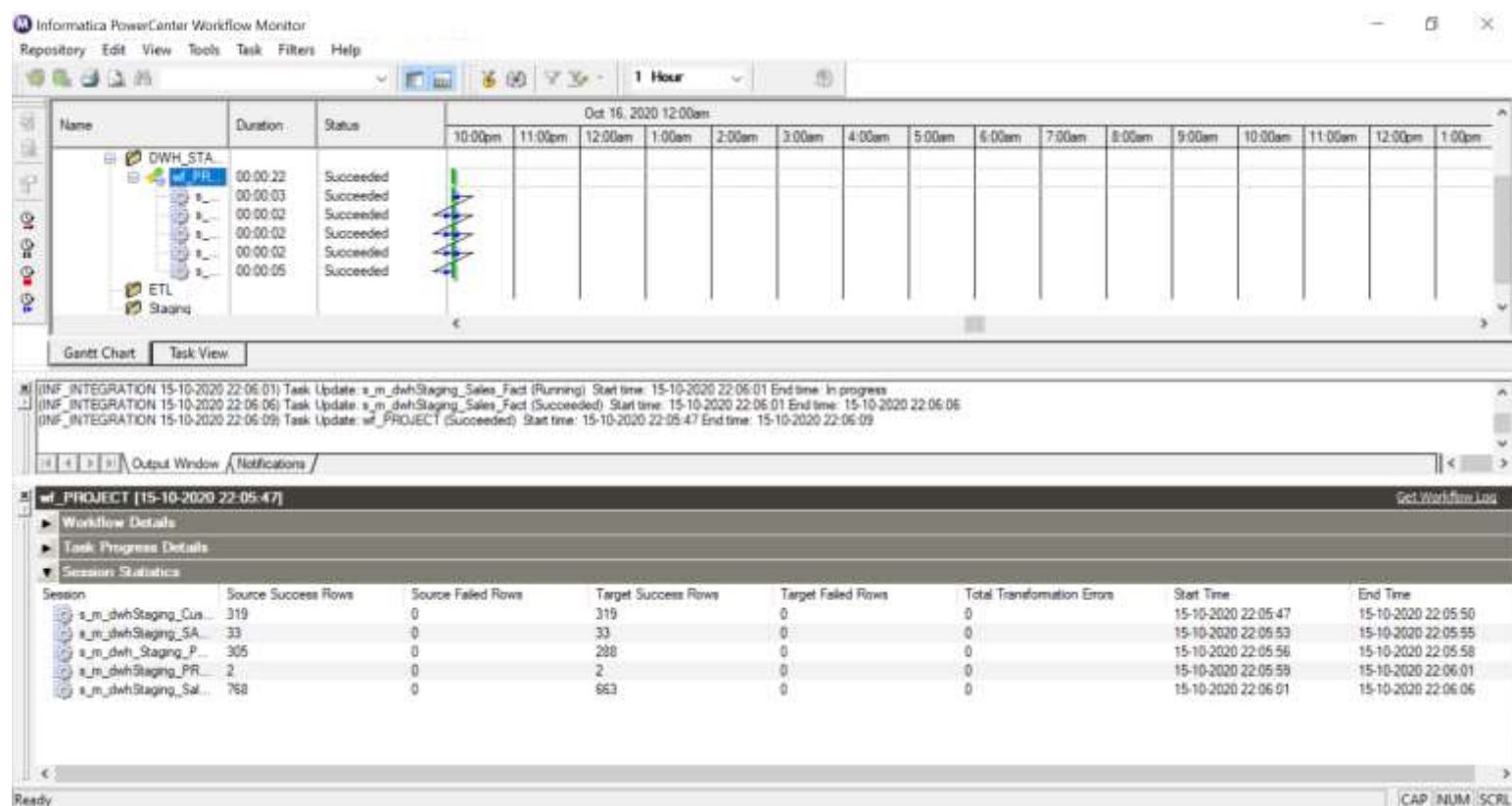


Fig 68. WORKFLOW MONITOR OF ALL TABLES EXCEPT DATE_DIM

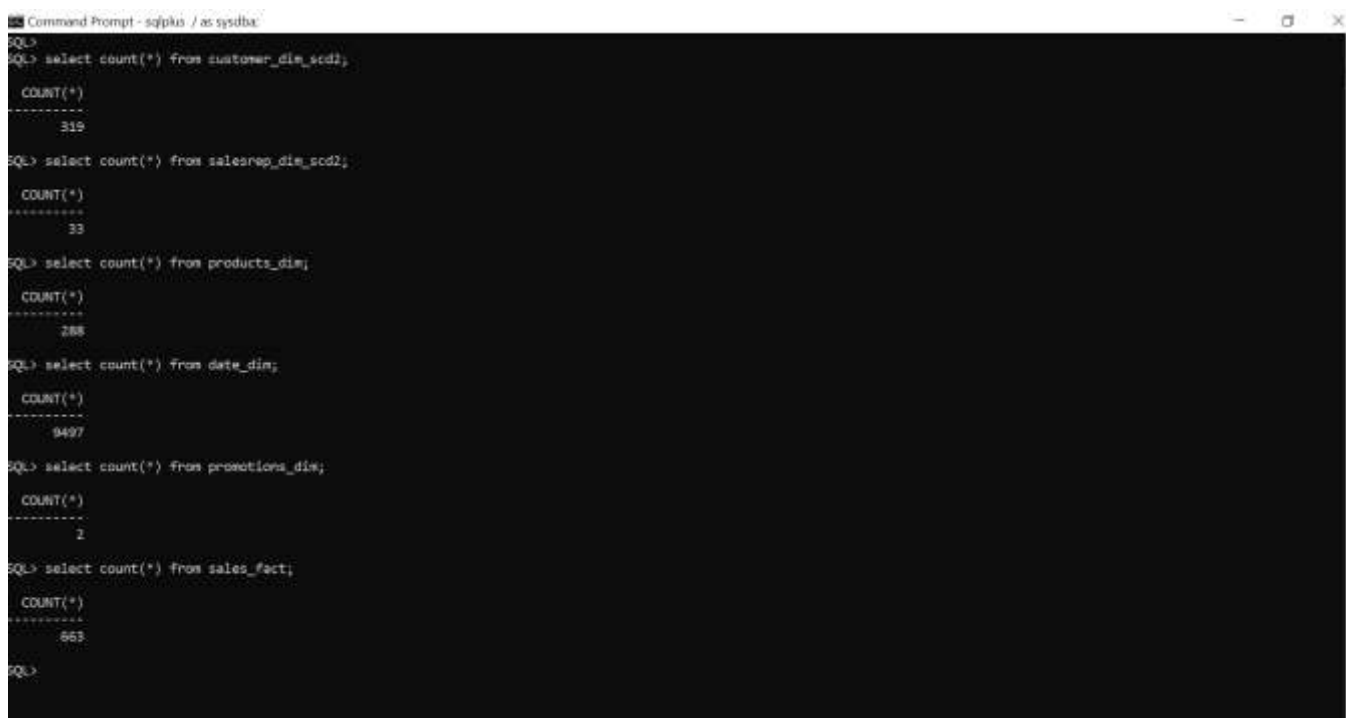


Fig 69. COUNT OF ALL TABLES IN DATABASE BEFORE RUNNING TESTS


```
Command Prompt - sqlplus / as sysdba

SQL> select count(*) from customer_dim_scd2;

COUNT(*)
-----
321

SQL> select count(*) from salesrep_dim_scd2;

COUNT(*)
-----
35

SQL> select count(*) from products_dim;

COUNT(*)
-----
289

SQL> select count(*) from date_dim;

COUNT(*)
-----
9497

SQL> select count(*) from promotions_dim;

COUNT(*)
-----
3

SQL> select count(*) from sales_fact;

COUNT(*)
-----
681

SQL>
```

Fig 70. COUNT OF ALL TABLES IN DATA WAREHOUSE AFTER RUNNING TESTS

CONCLUSION

All the transactional data from various sources was successfully loaded into a staging database which follows a certain model to reduce the redundancy in the incoming data. This data was further loaded into a star schema data warehouse where it was divided into various dimension tables following different slowly changing dimensions and into a fact table.

Thus the data warehouse has been successfully loaded with proper data which can be further used for all analytical purposes.

SCRIPTS USED FOR TABLE CREATION IN STAGING DATABASE AND IN DATA WAREHOUSE

CREATE TABLE CUSTOMERS

```
(
    CUSTOMER_ID          number,
    CUST_FIRST_NAME      nvarchar2(25),
    CUST_LAST_NAME       nvarchar2(25),
    STREET_ADDRESS       nvarchar2(50),
    POSTAL_CODE          number,
    CITY                 nvarchar2(25),
    STATE_PROVINCE       nvarchar2(25),
    COUNTRY_ID           nvarchar2(5),
    COUNTRY_NAME         nvarchar2(50),
    REGION_ID           number,
    NLS_LANGUAGE         nvarchar2(25),
    NLS_TERRITORY        nvarchar2(25),
    CREDIT_LIMIT         number,
    CUST_EMAIL           nvarchar2(60),
    PRIMARY_PHONE_NUMBER nvarchar2(25),
    PHONE_NUMBER_2       nvarchar2(25),
    ACCOUNT_MGR_ID       number,
    LOCATION_GTYPE       number,
    LOCATION_SRID        number,
    LOCATION_X           number,
    LOCATION_Y           number
);
```

CREATE TABLE EMPLOYEES

```
(
    EMPLOYEE_ID  number,
    FIRST_NAME   nvarchar2(25),
    LAST_NAME    nvarchar2(25),
    EMAIL        nvarchar2(60),
    PHONE_NUMBER nvarchar2(25),
    HIRE_DATE    nvarchar2(15),
    JOB_ID       nvarchar2(10),
    SALARY       number,
```

```

        COMMISSION_PCT number,
        MANAGER_ID      number,
        DEPARTMENT_ID   number
    );

CREATE TABLE PRODUCT_INFORMATION
(
    PRODUCT_ID          number,
    PRODUCT_NAME        nvarchar2(50),
    LANGUAGE_ID         varchar2(10),
    PRODUCT_DESCRIPTION nvarchar2(1000),
    CATEGORY_ID         number,
    WEIGHT_CLASS        number,
    WARRANTY_PERIOD     number,
    SUPPLIER_ID         number,
    PRODUCT_STATUS      nvarchar2(30),
    LIST_PRICE          number,
    MIN_PRICE           number,
    CATLOG_URL          nvarchar2(100)
);

CREATE TABLE CATEGORIES_TAB
(
    CATEGORY_ID          number,
    SUB_CATEGORY_NAME    varchar2(20),
    SUB_CATEGORY_DESCRIPTION varchar2(100),
    CATEGORY_NAME        nvarchar2(50),
    PARENT_CATEGORY_ID   number
);

CREATE TABLE PROMOTIONS
(
    PROMO_ID   number,
    PROMO_NAME nvarchar2(30)
);

CREATE TABLE ORDERS
(
    ORDER_ID   number,
    ORDER_DATE varchar(19),

```

```

ORDER_MODE    nvarchar2(25),
CUSTOMER_ID   number,
ORDER_STATUS  number,
ORDER_TOTAL   number,
SALES_REP_ID  number
);

CREATE TABLE ORDER_ITEMS
(
    ORDER_ID      number,
    LINE_ITEM_ID  number,
    PRODUCT_ID    number,
    UNIT_PRICE    number,
    QUANTITY      number,
    PROMO_ID      varchar(20)
);

CREATE TABLE STG_DATE
(
    CALENDAR_DATE DATE
);

CREATE TABLE CUSTOMER_DIM_SCD2
(
    CUSTOMER_DIM_ID      number NOT NULL PRIMARY KEY,
    CUSTOMER_ID          number,
    CUST_FIRST_NAME      nvarchar2(25),
    CUST_LAST_NAME       nvarchar2(25),
    STREET_ADDRESS       nvarchar2(50),
    POSTAL_CODE          number,
    CITY                 nvarchar2(25),
    STATE_PROVINCE       nvarchar2(25),
    COUNTRY_ID           nvarchar2(5),
    COUNTRY_NAME         nvarchar2(50),
    REGION_ID           number,
    NLS_LANGUAGE         nvarchar2(25),
    NLS_TERRITORY        nvarchar2(25),
    CREDIT_LIMIT         number,
    CUST_EMAIL           nvarchar2(60),

```

```

PRIMARY_PHONE_NUMBER nvarchar2(25),
PHONE_NUMBER_2       nvarchar2(25),
ACCOUNT_MGR_ID       number,
LOCATION_GTYPE         number,
LOCATION_SRID          number,
LOCATION_X             number,
LOCATION_Y             number,
START_DATE           timestamp(6) NOT NULL,
END_DATE             timestamp(6),
VERSION              number NOT NULL,
FLAG                 nvarchar2(10) NOT NULL,
INSERTED_DATE        timestamp(6) NOT NULL,
INSERTED_BY          nvarchar2(20) NOT NULL,
UPDATED_DATE         timestamp(6) NOT NULL,
UPDATED_BY           nvarchar2(20) NOT NULL
);

```

```

CREATE TABLE SALESREP_DIM_SCD2
(
    SALESREP_DIM_ID number NOT NULL PRIMARY KEY,
    EMPLOYEE_ID     number,
    FIRST_NAME      nvarchar2(25),
    LAST_NAME       nvarchar2(25),
    EMAIL           nvarchar2(60),
    PHONE_NUMBER    nvarchar2(25),
    HIRE_DATE       nvarchar2(15),
    JOB_ID          nvarchar2(10),
    SALARY          number,
    COMMISSION_PCT  number,
    MANAGER_ID      number,
    DEPARTMENT_ID   number,
    START_DATE      timestamp(6) NOT NULL,
    END_DATE        timestamp(6),
    VERSION         number NOT NULL,
    FLAG            nvarchar2(10) NOT NULL,
    INSERTED_DATE   timestamp(6) NOT NULL,
    INSERTED_BY     nvarchar2(20) NOT NULL,
    UPDATED_DATE    timestamp(6) NOT NULL,

```



```

        UPDATED_BY          nvarchar2(20) NOT NULL
    );

CREATE TABLE PRODUCTS_DIM
(
    PRODUCT_DIM_ID          number NOT NULL PRIMARY KEY,
    PRODUCT_ID              number,
    PRODUCT_NAME             nvarchar2(50),
    LANGUAGE_ID             nvarchar2(2),
    MIN_PRICE                number,
    LIST_PRICE               number,
    PRODUCT_STATUS          nvarchar2(30),
    SUPPLIER_ID              number,
    WARRANTY_PERIOD          number,
    WEIGHT_CLASS             number,
    PRODUCT_DESCRIPTION      nvarchar2(1000),
    CATEGORY_ID              number,
    CATALOG_URL              nvarchar2(100),
    SUB_CATEGORY_NAME        nvarchar2(25),
    SUB_CATEGORY_DESCRIPTION nvarchar2(1000),
    PARENT_CATEGORY_ID       number,
    CATEGORY_NAME            nvarchar2(25),
    INSERTED_DATE            timestamp(6),
    INSERTED_BY              nvarchar2(15),
    UPDATED_DATE             timestamp(6),
    UPDATED_BY               nvarchar2(15)
);

```

```

CREATE TABLE PROMOTIONS_DIM
(
    PROMOTION_DIM_ID number NOT NULL PRIMARY KEY,
    PROMO_ID          number,
    PROMO_NAME         nvarchar2(30),
    INSERTED_DATE      timestamp(6),
    INSERTED_BY        nvarchar2(15),
    UPDATED_DATE       timestamp(6),
    UPDATED_BY         nvarchar2(15)
);

```

```

CREATE TABLE DATE_DIM
(
    DATE_DIM_ID          number NOT NULL PRIMARY KEY,
    SALES_DATE           date,
    SALES_DAY_OF_THE_YEAR number,
    SALES_MONTH          number,
    SALES_YEAR           number,
    SALES_QUARTER        number,
    SALES_MONTH_NAME     nvarchar2(25)
);

```

```

CREATE TABLE SALES_FACT
(
    PK_SALES_FACT_KEY number NOT NULL PRIMARY KEY,
    ORDER_ID          number,
    CUSTOMER_DIM_ID   number,
    SALESREP_DIM_ID   number,
    PRODUCT_DIM_ID    number,
    PROMOTION_DIM_ID  number,
    DATE_DIM_ID       number,
    DOLLARS_SOLD       number,
    QUANTITY_SOLD      number,
    INSERT_DATE        timestamp(6) NOT NULL,
    INSERT_BY          nvarchar2(20) NOT NULL,
    UPDATE_DATE        timestamp(6) NOT NULL,
    UPDATE_BY          nvarchar2(30) NOT NULL
);

```