INCREMENTAL LOAD/CREATION OF DIMENSION TABLE LOGIC AT MAYO CLINIC

In this particular document we shall be looking at the incremental load logic at Mayo Clinic. This document is being prepared because of the following purposes.

- Data Integrator does not have a dimension creation wizard. Thus using the transforms within the Data Integrator we need to make the required type-1 and type-2 dimensions.
- This document will serve as a template for all new developers to populate the data warehouse and the data mart as part of the PAMDSS project at Mayo Clinic.
- The document is attached with the required templates which shall further help in understanding of the processes. With this document we seek to systematize the development of history, base and the dimension tables within the project.
- Prevent some basic mistakes that developers can make while populating the history and the dimension tables.

Note: Along with this document we shall be attaching/referring to the required *atl* data flows. As part of the understanding we shall be attaching the screen shots of the required *atl* and sending the required *atl.*

Methodology for populating the Data Warehouse

Scenario for loading the data warehouse: depending upon the project requirements a client within the Data warehouse can have the following tables:

- Base tables
- History tables.

In this document we shall be looking at populating the History tables. The methodology for populating the base tables is simpler and does not require special treatment.

We have history making events and non-History making events. We shall have to track inserts, updates and deletes for both the events. The scenario can be expressed by the use of the following matrix.

Event	**Insert**	**Update**	**Delete**
History making attributes	1	2	3
Non-history making attributes	4	5	6

Also many times we have last update time stamps in the source. We shall be looking at a specific case where we have the time stamp in the source and have to populate the target in such a scenario.

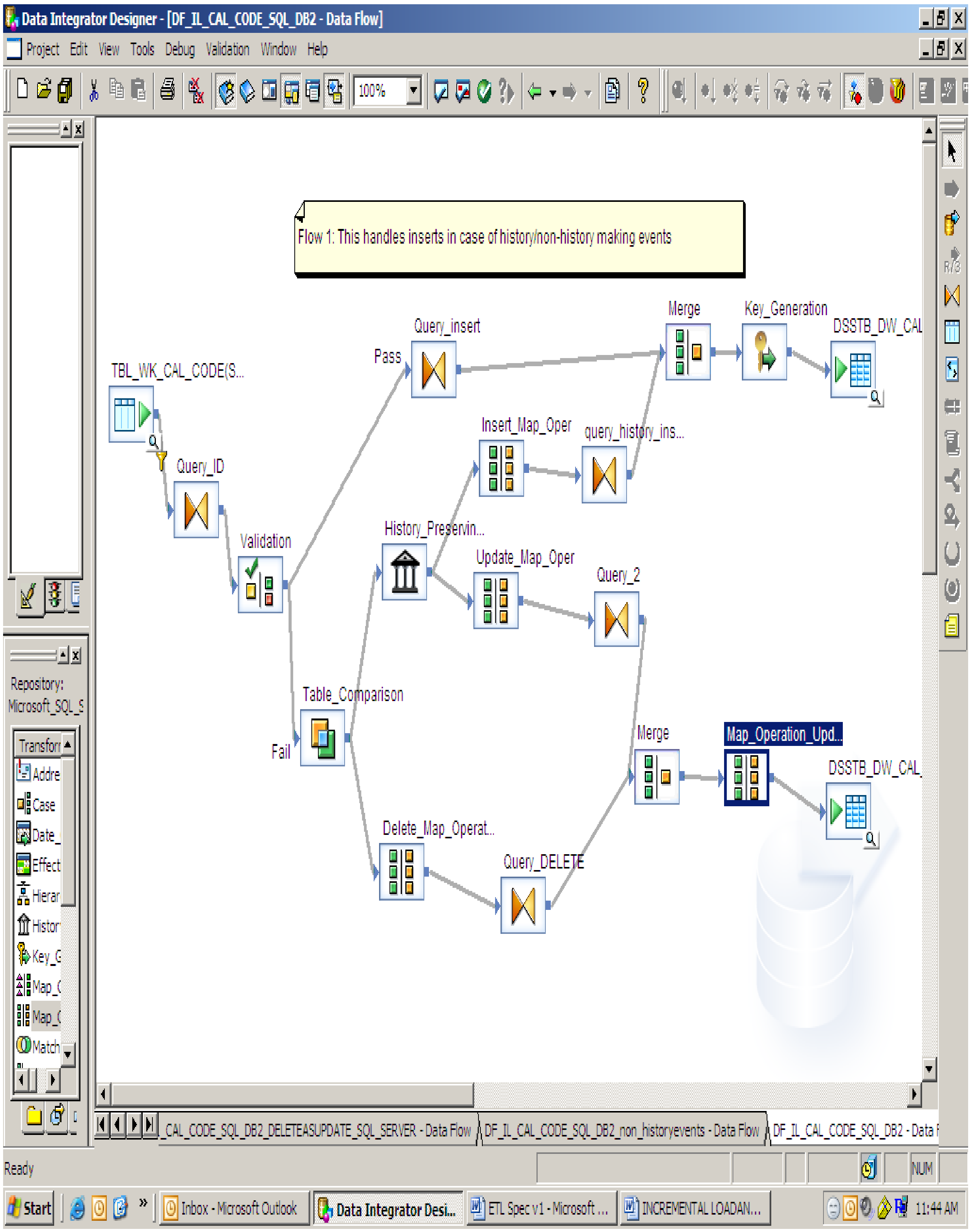
Methodology for handling Inserts: That is in this case we shall be looking at scenarios 1 and 4.

There shall be two kinds of inserts within this scenario.

- Those records whose business key in not present in the data warehouse.
- Those history making records that have been updated. We shall generate an insert row for each one of them.

Methodology for handling those records whose business key is not present in the Data ware house

For this purpose we propose that you follow activities within Flow 1. In the following diagram Flow1 includes Query_ID, Validation, Query_Insert, Merge and Surrogate key generation.


Data Integrator Designer - [DF_IL_CAL_CODE_SQL_DB2 - Data Flow]
Project Edit View Tools Debug Validation Window Help
100%
Flow 1: This handles inserts in case of history/non-history making events
TBL_WK_CAL_CODE(S...
Query_ID
Validation
Pass
Query_insert
Merge
Key_Generation
DSSTB_DW_CAL
Insert_Map_Oper
query_history_ins...
History_Preservin...
Update_Map_Oper
Query_2
Table_Comparison
Fail
Merge
Map_Operation_Upd...
DSSTB_DW_CAL
Delete_Map_Operat...
Query_DELETE
Repository:
Microsoft_SQL_S
Transform
Addre
Case
Date_
Effect
Hierar
Histor
Key_G
Map_(
Map_(
Match
_CAL_CODE_SQL_DB2_DELETEASUPDATE_SQL_SERVER - Data Flow
DF_IL_CAL_CODE_SQL_DB2_non_historyevents - Data Flow
DF_IL_CAL_CODE_SQL_DB2 - Data F
Ready
NUM
Start
Inbox - Microsoft Outlook
Data Integrator Desi...
ETL Spec v1 - Microsoft ...
INCREMENTAL LOADAN...
11:44 AM

Steps in the process are as following:

1. Lookup for the business key in the data warehouse table.
2. Validate whether the Business key is Null or not. Within the Validation transform put the “Business key is NULL” as the custom condition.
3. In case Business Key is NULL then it is straight a case of an insert.
4. Set the Insert Time Stamp, Update Time Stamp, Effective From (History Start) Time Stamp as today’s date [Query_Insert].
5. Set the Effective through(History End time stamp) as the end of time.

Note: This shall handle Insert operation for all non-history making events also.

Handling Inserts from History Making Records: for the records that are to be updated we shall be handling separately within the updates section. However for the new row to be inserted after the history preserving transform, we perform the following operations.

- Map the Insert row as normal (discard all the other row types) through the use of Map Transform
- Merge the output with the Inserts from Flow1.
- Generate a surrogate key after union operation.

Handling Updates for History Making attributes (scenario 3): In this particular case we have the following scenario:

- The Lookup value of the business key(from Query_ID) is not NULL.
- We have History Start and History End Time stamps in the target table.
- We know the attributes on which we propose to preserve history.

Then the combination of activities used shall be as following:

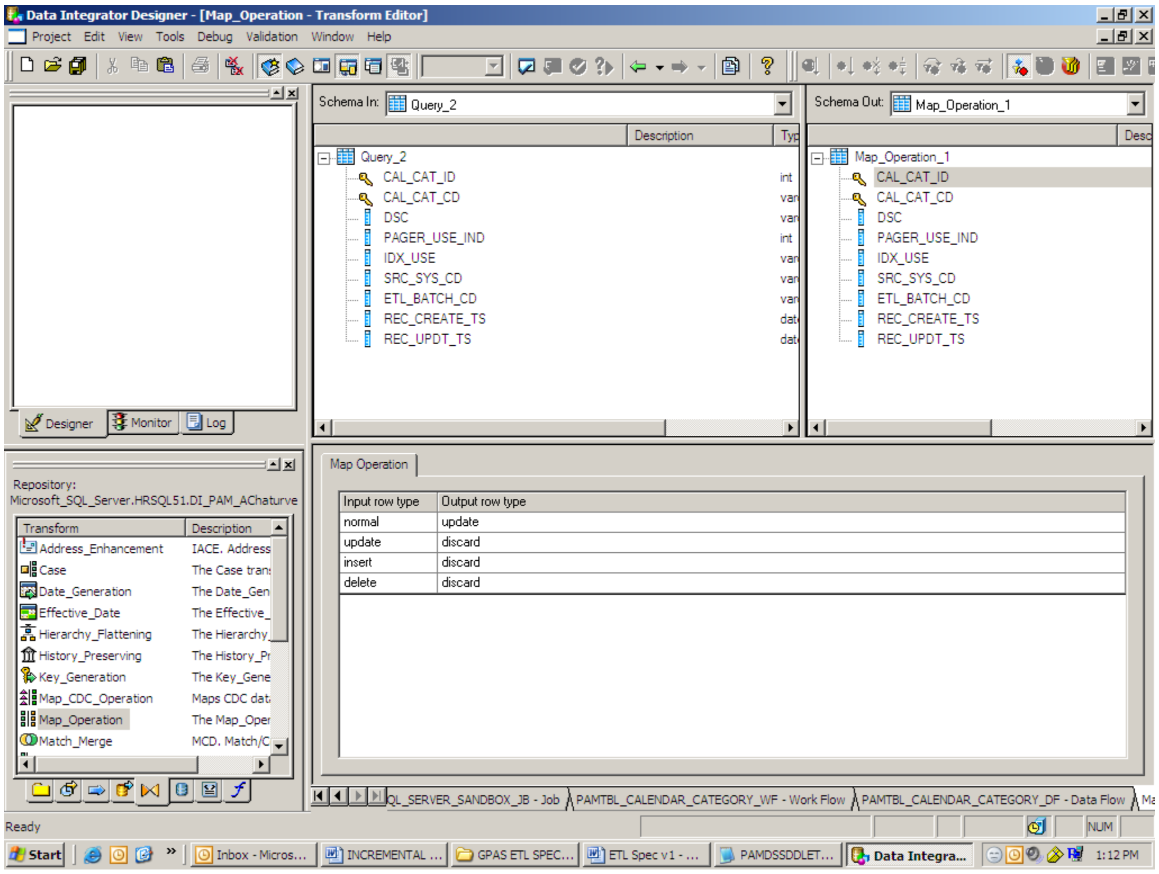
- Table Comparison: this shall help determine all the required updates within the target table. All the output rows from this transform shall be marked as ‘U’. The following care should be taken when we use this transform:
  - The Compare column should not have the Target table Load Time Stamps and any of the primary key columns.

    - The compare column should have only the history tracking attributes from the target table.
- History Preserving transform: The History preserve transform shall have input rows marked as 'Update'. Within the History preserve transform we need to take the following care:
    - The history preserve columns are the same as in Table Comparison.
    - If there is a row status in the target table then its *Reset Status* should be changed.
- Map the update rows: this is done using the MAP transform and only marking the Update rows output as Normal. The other row types should be discarded.

Handling Deletes: within Data Integrator we have an option for marking Deletes as Update rows within the History Preserve transform. Our experience with the same has not been very convincing. The alternate methodology is as following:

- In the Table Comparison, use the option Preserve deleted rows.
- Map the deleted rows using MAP transform(discard all input types except the deletes).
- Set the last update time stamp and the History end date as today's date.
- Set the row status as 'delete'.
- Merge it with the update rows.

Explanation for the transform Map_Operation_Update: It is important to consider why we need to add one last Map transform before we load the data into the target table. For case of *Normal*(we converted the update rows into Normal rows) rows mark the respective row as *Update*(as given in the following diagram). The reason for doing the given step is that in case the rows are marked as normal then they shall be inserted as a separate row in the target table. In case we have primary key constraints in the target it shall then result in an error.


Data Integrator Designer - [Map_Operation - Transform Editor]
Project Edit View Tools Debug Validation Window Help
Designer
Monitor
Log
Ready
NUM
Start
1:12 PM

Schema In: Query_2

	Description	Typ
Query_2		
CAL_CAT_ID		int
CAL_CAT_CD		var
DSC		var
PAGER_USE_IND		int
IDX_USE		var
SRC_SYS_CD		var
ETL_BATCH_CD		var
REC_CREATE_TS		dat
REC_UPDT_TS		dat

Schema Out: Map_Operation_1

	Desc
Map_Operation_1	
CAL_CAT_ID	
CAL_CAT_CD	
DSC	
PAGER_USE_IND	
IDX_USE	
SRC_SYS_CD	
ETL_BATCH_CD	
REC_CREATE_TS	
REC_UPDT_TS	

Repository:
Microsoft_SQL_Server.HRSQL51.DI_PAM_AChaturve

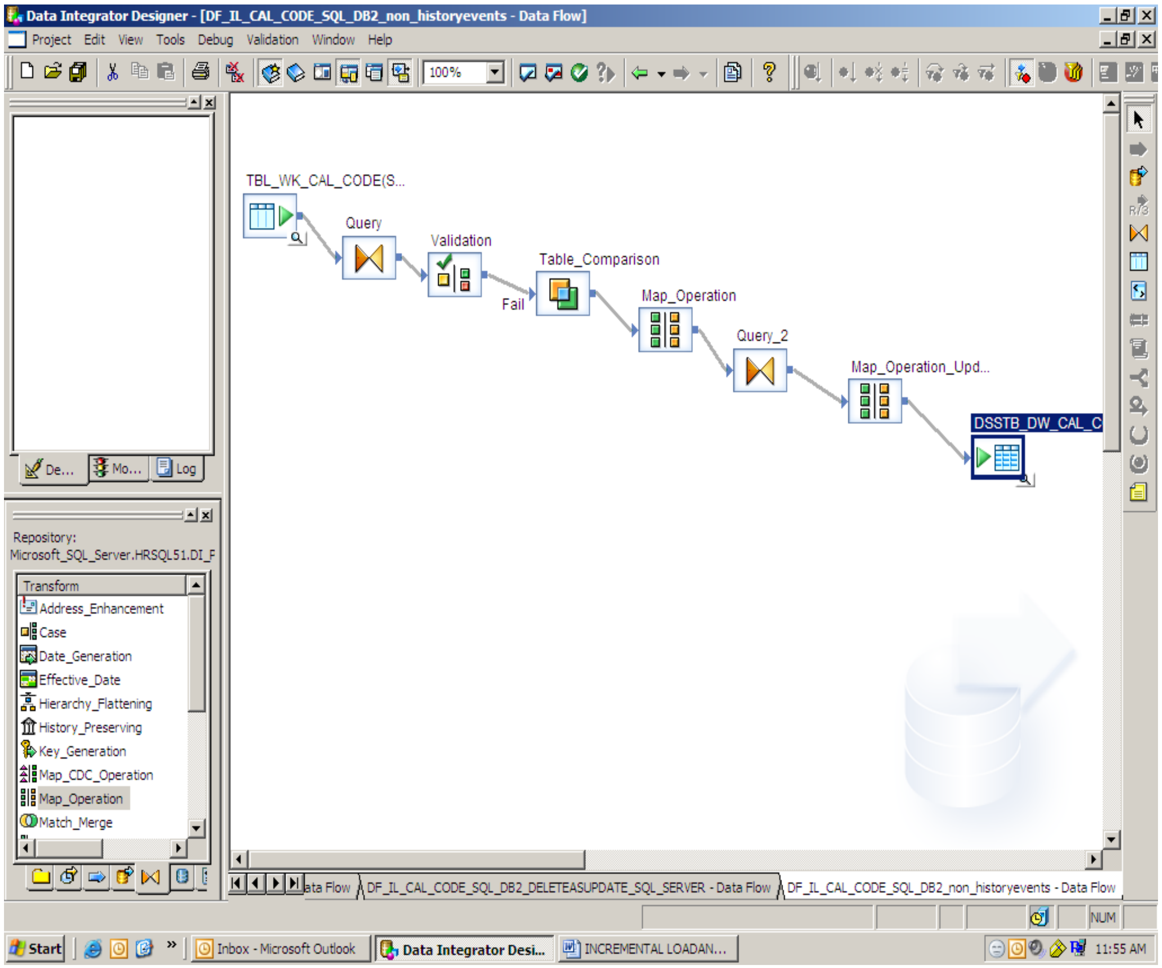
Transform	Description
Address_Enhancement	IACE. Address
Case	The Case tran:
Date_Generation	The Date_Gen
Effective_Date	The Effective_
Hierarchy_Flattening	The Hierarchy
History_Preserving	The History_Pr
Key_Generation	The Key_Gene
Map_CDC_Operation	Maps CDC dat.
Map_Operation	The Map_Oper
Match_Merge	MCD. Match/C

Map Operation

Input row type	Output row type
normal	update
update	discard
insert	discard
delete	discard

QL_SERVER_SANDBOX_JB - Job | PAMTBL_CALENDAR_CATEGORY_WF - Work Flow | PAMTBL_CALENDAR_CATEGORY_DF - Data Flow | Ma

Handling Updates for the Non-history making attributes (Scenario 5): Unfortunately we cannot handle this case within the same data flow. For this we designed another data flow whose implementation shall follow the execution of the previous data flow. In this case we simply track the rows flagged as *Updates* from the Table Comparison Transform in the following manner:



In the Query_2 step we shall set the Last Update Time Stamp to today's date.

(Note we should take the following care while implementing the above steps)

- In the Compare column category of Table Comparison include only the non-History making attributes from the target table. Do not include the target table load Time Stamps and the primary key columns in the Compare column category.
- In the last Map operation Map_Operation_Update set the Input *Normal* rows as output type *update*.

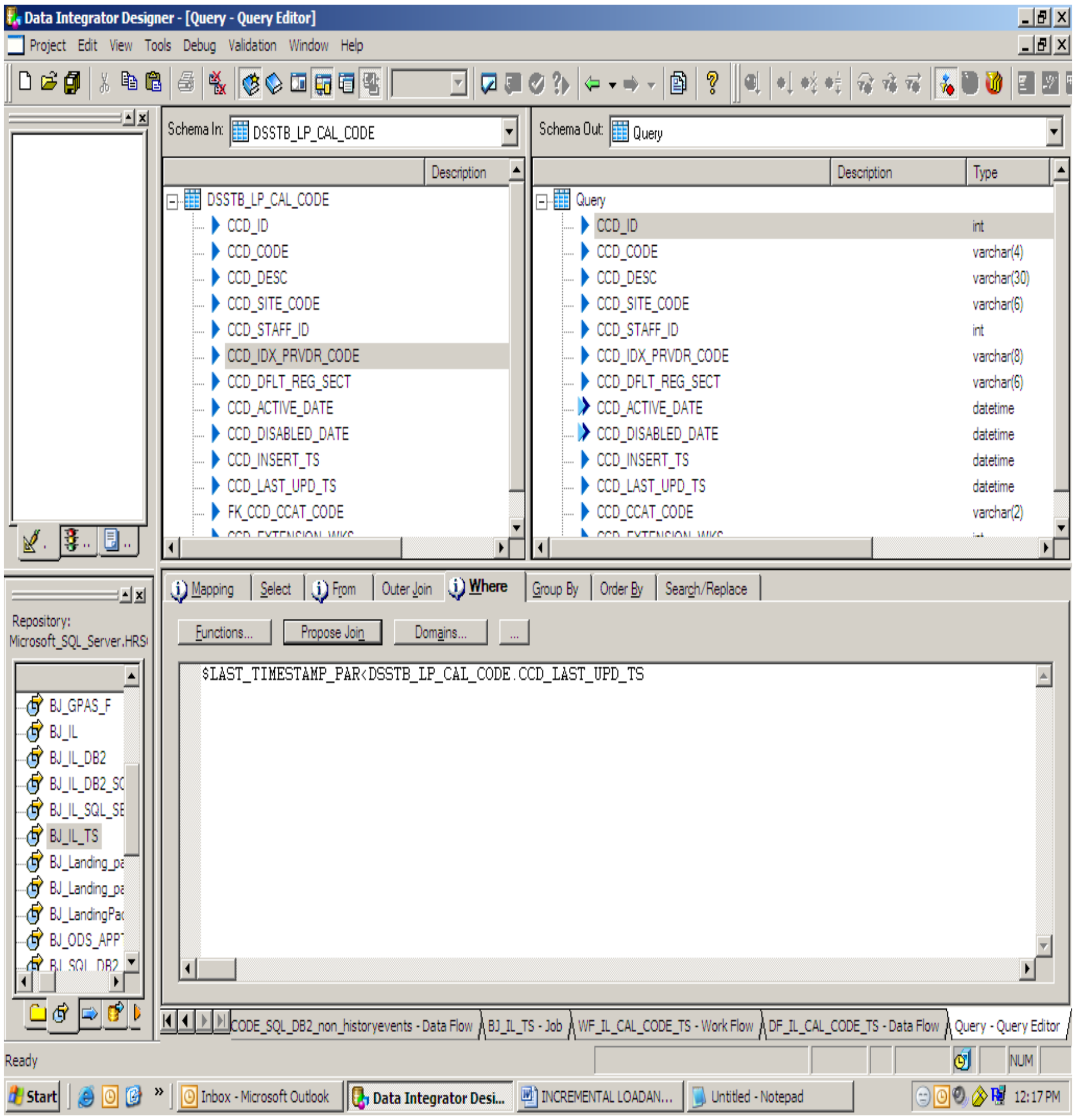
Input Tables have a Last Update Time stamp: In case the input tables have a last update time stamp then we can limit the number of rows to be scanned. This is done with the help of the following steps:

- Generate a script before the dataflow in which you capture the Maximum value of the **Last Update Time stamp from the target table(see the marked rows in red in the attachment below).** This can be done by writing a simple script as following:

```
$FULLUPLOAD=’NO’;
IF ($FULLUPLOAD='YES')
begin
        print( 'The process of full upload is set to begin');
        sql( 'SQL_SERVER_SANDBOX', 'TRUNCATE TABLE DSSTB_DW_CAL_CODE_TS');
        sql( 'SQL_SERVER_SANDBOX','INSERT INTO
DSSTB_DW_CAL_CODE_TS(CCD_DW_ID,CCD_DW_ROW_STATUS,CCD_DW_INSERT_TS,
CCD_DW_LAST_UPD_TS,CCD_ID,CCD_CODE,CCD_DESC,CCD_SITE_CODE,CCD_STAF
F_ID,CCD_IDX_PRVDR_CODE,CCD_DFLT_REG_SECT,CCD_ACTIVE_DATE,CCD_DISAB
LED_DATE,CCD_INSERT_TS,CCD_LAST_UPD_TS,CCD_CCAT_CODE,CCD_EXTENSION_
WKS) VALUES(0,\'\',\'1900-01-01 00:00:00.000\',\'1900-01-01
00:00:00.000\',0,\'N\A\',\'\',\'\',0,\'\',\'\',\'1900-01-01 00:00:00.000\',\'1900-01-01
00:00:00.000\',\'1900-01-01 00:00:00.000\',\'1900-01-01 00:00:00.000\',\'\',0)');
   $Last_Timestamp_var='1900-01-01 00:00:00.000';
end
else
begin
        $Last_Timestamp_var=SQL('SQL_SERVER_SANDBOX','SELECT
MAX(CCD_LAST_UPD_TS) FROM DSSTB_DW_CAL_CODE_TS');
        print('the process of incremental load has started');
end
```

Thereafter we do the following steps:

- Declare the variable as a parameter.
- Pass the parameter into the dataflow.
- In the ‘where’ clause (as given in the attachment below) select the time stamps from the **source table with values greater than the parameter value.**



Schema Out: Query	Description	Type
CCD_ID		int
CCD_CODE		varchar(4)
CCD_DESC		varchar(30)
CCD_SITE_CODE		varchar(6)
CCD_STAFF_ID		int
CCD_IDX_PRVDR_CODE		varchar(8)
CCD_DFLT_REG_SECT		varchar(6)
CCD_ACTIVE_DATE		datetime
CCD_DISABLED_DATE		datetime
CCD_INSERT_TS		datetime
CCD_LAST_UPD_TS		datetime
CCD_CCAT_CODE		varchar(2)

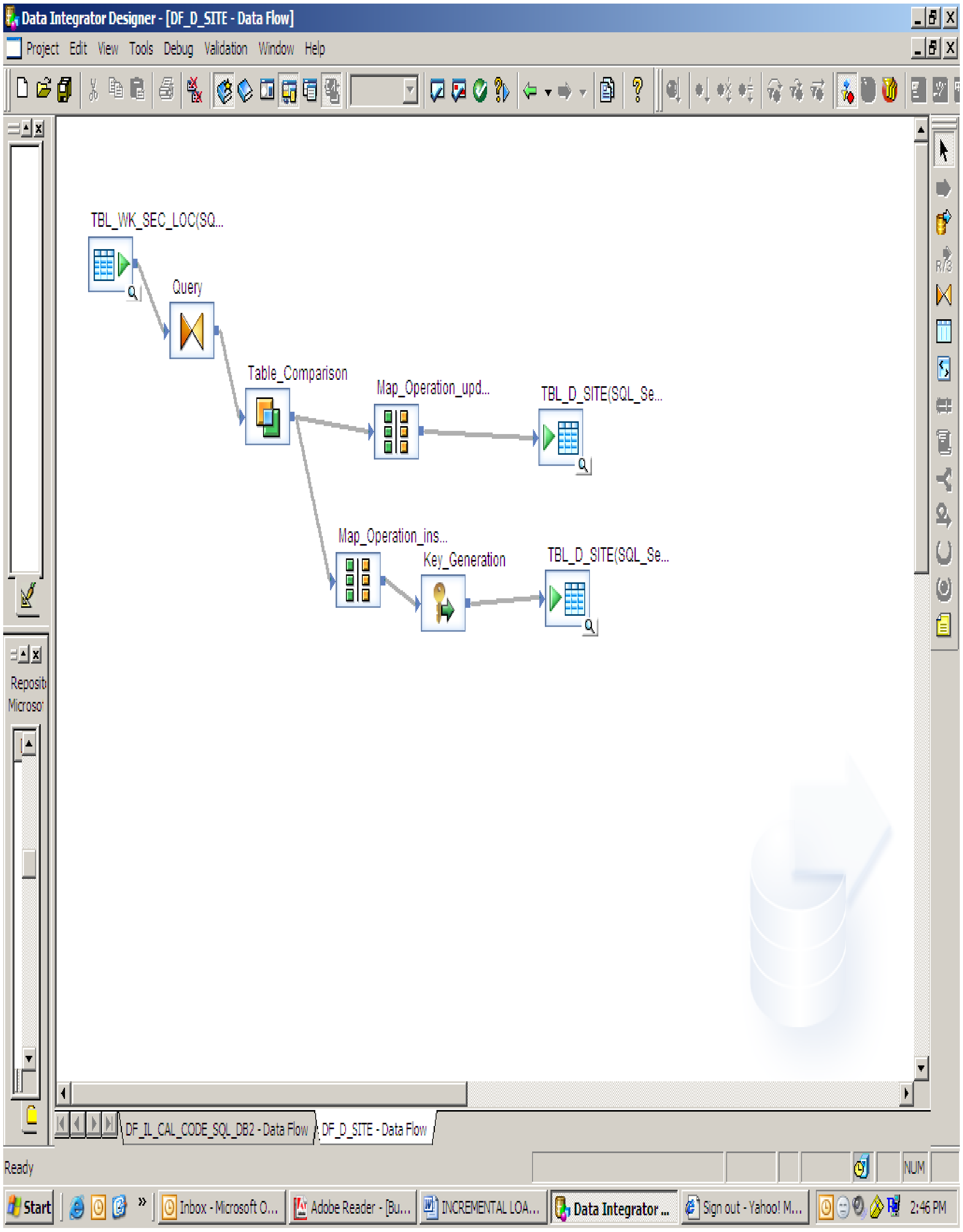
```
$LAST_TIMESTAMP_PAR<DSSTB_LP_CAL_CODE.CCD_LAST_UPD_TS
```

Dimension Tables: In this case we shall look at both Type-1 and Type-2 dimension tables.

How to work with Type-1 Dimension tables: In this case we wish to carry out the following steps:

- In case of an update keep the latest record. Do not increment the dimension key.
- However in case of an insert we wish to increment the dimension key.

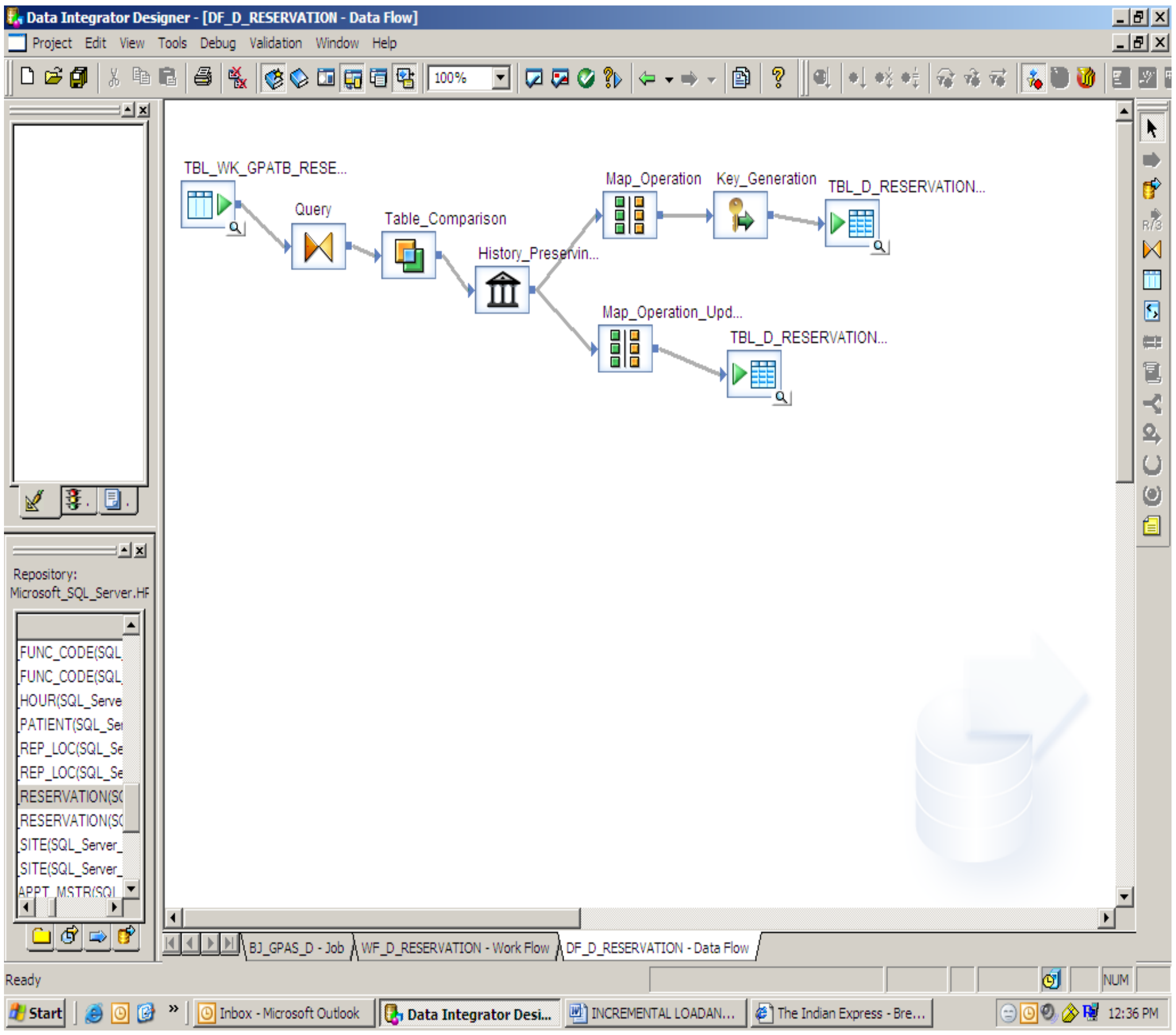
We carry the above using the following steps:



The logic: the logic for the above data flow is as following.

From the source tables we shall get records marked as Insert(new records) and updates(old records changed). In case the record is an update then load it directly onto the target table. This is done by passing it through a map transform(*the output type for the Update being 'Update' with other rows types being 'discarded'*) .However in case it is an insert, then we generate the surrogate key.

Methodology for populating type-2 Dimension Tables: The methodology for populating the type-2 dimension tables is the same as populating the history tables with history making attributes. The sequence of operation and the logic are as following:


Data Integrator Designer - [DF_D_RESERVATION - Data Flow]
Project Edit View Tools Debug Validation Window Help
100%
TBL_WK_GPATB_RESE...
Query
Table_Comparison
History_Preservin...
Map_Operation
Key_Generation
TBL_D_RESERVATION...
Map_Operation_Upd...
TBL_D_RESERVATION...
Repository:
Microsoft_SQL_Server.HF
FUNC_CODE(SQL
FUNC_CODE(SQL
HOUR(SQL_Serve
PATIENT(SQL_Se
REP_LOC(SQL_Se
REP_LOC(SQL_Se
RESERVATION(S
RESERVATION(S
SITE(SQL_Server_
SITE(SQL_Server_
APPT_MSTR(SQL
BJ_GPAS_D - Job
WF_D_RESERVATION - Work Flow
DF_D_RESERVATION - Data Flow
Ready
NUM
Start
Inbox - Microsoft Outlook
Data Integrator Desi...
INCREMENTAL LOADAN...
The Indian Express - Bre...
12:36 PM