# TPC-DI

TPC-Di using PostgreSQL and Pentaho Data Integration Services

**Table creation:**

Every table that has EffectiveDate and EndDate fields is created with a UNIQUE INDEX on the Natural key and the EndDate. In this way, we are ensuring that we do not violate the Primary key constraints of the OLTP database.

**Historical-tracking dimension tables:**

Some tables like DimCompany are populated 1 time with data. Hovewer, in the historial file, it is possible to find multiple records for the same natural key (business key). This means that we need to keep track of the changes, as it is described in chapter 4.4.1 of the specification document. In this implementation, we achieve this by comparing the natural keys of every row that is read from the source files with the current records of the table.

If there is no matching, the record is marked as "new" and it is inserted in the database.

If there is a matching, then the old record is closed (enddate = effectivedate of the new recods && iseffective = 0) and the new record is inserted into the database.

**Historical load description:**

**DimCustomer and DimAcccount:**

Both tables are filled with table from CustomerMgmt file. As it is mentioned, this file was transformed to csv and as a result the package uses this file as input.

For DimCustomer, we are using only the rows that have ActionType in (‘NEW’, ‘UPDCUST’, ‘INACT’). Firstly, we are inserting only the New customers to the target table. Then, for every customer that has at least one Update or Inactive row, we are closing all the old records. We are defining here that by saying “closing records” we mean that we set to the records that have EndDate = ‘9999-12-31’ and IsCurrent = ‘True’ the EffectiveDate of the new records and ‘False’, respectively. After that, we are calculating in memory the order of the new records. This means that if a customer has more than one updates, we are sorting the records by ActionDate. In this way, we are achieving to keep track of the changes during the time. We also set the EndDate of the update records of each customer that have a forthcoming update to the EffectiveDate of the next record and the IsCurrent to ‘False’, which means that this record will not be the active one for this customer. All this process is done by using the Analytical query component of Pentaho DI that allows us to Partition the dataset by CustomerID (C\_ID) and use the LEAD and LAG rows.

Below, there is an example of a Customer with 1 record with ActionType ‘NEW’, 2 records with ‘UPDCUST’ and 1 record with ‘INACT’ :

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **sk\_customerid** | **customerid** | **taxid** | **status** | **lastname** | **firstname** | **…** | **marketingnameplate** | **iscurrent** | **batchid** | **effectivedate** | **enddate** |
| 1934 | 21 | 494-50-2485 | ACTIVE | Neely | Herman | … | NULL | FALSE | 1 | 2007-07-15 | 2007-09-04 |
| 4760 | 21 | 494-50-2485 | ACTIVE | Neely | Herman | … | NULL | FALSE | 1 | 2007-09-04 | 2008-05-08 |
| 4761 | 21 | 494-50-2485 | ACTIVE | Neely | Herman | … | NULL | FALSE | 1 | 2008-05-08 | 2008-09-27 |
| 4762 | 21 | 494-50-2485 | INACTIVE | Neely | Herman | … | NULL | TRUE | 1 | 2008-09-27 | 9999-12-31 |

The update records of DimCustomer contain only the values of the updated field. The inserted rows should maintain the value from the previous rows for the field that are not updated and assign the new values for the updated fields. In our code we achieve this by initially inserting every update record with nulls to the non-updated field and with the new value to the updated one (we have created the table with nullable fields and in the end of the initial load we alter the fields to not nullable). After the table is fully loaded, we are executing the [sql query 1](#_Query_1) of the Apendix A. For every customer we are recursively update the records from the oldest to the earliest one, by setting to the null fields the value from the previous record.

Finally, we are updating DimCustomer to assign the right values to the prospect fields. This is done by getting all the records of DimCustomer and joining the rows with Prospect.csv source file on LastName, FistName, Address 1, Address 2 and Postcode.

The initial load of DimAccount is happening in the same Job with DimCustomer but after the Initial Load of DimCustomer because the first one has the Surrogate key of DimCustomer in each fields. As a result, a change to DimCustomer triggers 1 or more change(s) to DimAccount.

We are following the same strategy as DimCustomer. Firstly, we are inserting only the new accounts, then we close the old records and we are proceeding to the update phase. The difference here is that a row of DimAccount may be update because of an update to the account fields (ActionType in ‘UPDACC’, ‘CLOSEACCT’) or to the customer that holds this account (ActionType in ‘UPDCUST’, ‘INACT’).

Regarding execution path of the updates to the customer, we should first join with the DimCustomer table to get the old surrogate key that will be used later to join with the records of DimAccount that should be closed. After that, we union the two path and we do similar process with the update of DimCustomer (Analytical query etc).

Finally, the view of the account of customer 21 is:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **sk\_accountid** | **accountid** | **sk\_brokerid** | **sk\_customerid** | **status** | **…** | **iscurrent** | **batchid** | **effectivedate** | **enddate** |
| 98 | 21 | 2261 | 1934 | ACTIVE | **…** | FALSE | 1 | 2007-07-15 | 2007-08-12 |
| 9248 | 21 | 2261 | 1934 | INACTIVE | **…** | FALSE | 1 | 2007-08-12 | 2007-09-04 |
| 9249 | 21 | 2261 | 4760 | INACTIVE | **…** | FALSE | 1 | 2007-09-04 | 2008-05-08 |
| 9250 | 21 | 2261 | 4761 | INACTIVE | **…** | FALSE | 1 | 2008-05-08 | 2008-09-27 |
| 9251 | 21 | 2261 | 4762 | INACTIVE | **…** | TRUE | 1 | 2008-09-27 | 9999-12-31 |
| 100 | 559 | 3785 | 4760 | ACTIVE | **…** | FALSE | 1 | 2008-02-19 | 2008-05-08 |
| 9252 | 559 | 3785 | 4761 | ACTIVE | **…** | FALSE | 1 | 2008-05-08 | 2008-09-27 |
| 9253 | 559 | 3785 | 4762 | INACTIVE | **…** | FALSE | 1 | 2008-09-27 | 2009-01-18 |
| 9255 | 559 | 1724 | 4762 | INACTIVE | **…** | TRUE | 1 | 2009-01-18 | 9999-12-31 |
| 99 | 1003 | 3208 | 4761 | ACTIVE | **…** | FALSE | 1 | 2008-08-18 | 2008-09-27 |
| 9246 | 1003 | 3208 | 4762 | INACTIVE | **…** | FALSE | 1 | 2008-09-27 | 2013-08-09 |
| 9254 | 1003 | 350 | 4762 | INACTIVE | **…** | FALSE | 1 | 2013-08-09 | 2015-06-15 |
| 9247 | 1003 | 350 | 4762 | INACTIVE | **…** | TRUE | 1 | 2015-06-15 | 9999-12-31 |

The last transform we need to do is to delete the rows that triggered more than 1 update during the same day. This is done by executing [query 2](#_Query_2).

**Transformation to files:**

1. CustomerMgmt.xml to csv using XSLT
2. FINWIRE\* merge to FINWIRE\_MERGED

**Apendix A**

# Query 1

1. **with** RECURSIVE **temp** **as**(
2. **select** ROW\_NUMBER () OVER (
3. PARTITION **BY** customerid
4. **ORDER** **BY** effectivedate
5. ) **as** rn,
6. \*
7. **from** dimcustomer
8. ),new\_dimcustomer **as**(
9. **select** \* **from** **temp**
10. **where** rn = 1
11. **union**
12. **select**
13. b.rn
14. ,b.SK\_CustomerID
15. ,b.customerid
16. ,case **when** b.TaxID **is** null **then** a.TaxID **else** b.TaxID **end** **as** TaxID
17. ,case **when** b.Status **is** null **then** a.Status **else** b.Status **end** **as** Status
18. ,case **when** b.LastName **is** null **then** a.LastName **else** b.LastName **end** **as** LastName
19. ,case **when** b.FirstName **is** null **then** a.FirstName **else** b.FirstName **end** **as** FirstName
20. ,case **when** b.MiddleInitial **is** null **then** a.MiddleInitial **else** b.MiddleInitial **en d** **as** MiddleInitial
21. ,case **when** b.Gender **is** null **then** a.Gender **else** b.Gender **end** **as** Gender
22. ,case **when** b.Tier **is** null **then** a.Tier **else** b.Tier **end** **as** Tier
23. ,case **when** b.DOB **is** null **then** a.DOB **else** b.DOB **end** **as** DOB
24. ,case **when** b.AddressLine1 **is** null **then** a.AddressLine1 **else** b.AddressLine1 **end** **as** AddressLine1
25. ,case **when** b.AddressLine2 **is** null **then** a.AddressLine2 **else** b.AddressLine2 **end** **as** AddressLine2
26. ,case **when** b.PostalCode **is** null **then** a.PostalCode **else** b.PostalCode **end** **as** PostalCode
27. ,case **when** b.City **is** null **then** a.City **else** b.City **end** **as** City
28. ,case **when** b.StateProv **is** null **then** a.StateProv **else** b.StateProv **end** **as** StateProv
29. ,case **when** b.Country **is** null **then** a.Country **else** b.Country **end** **as** Country
30. ,case **when** b.Phone1 **is** null **then** a.Phone1 **else** b.Phone1 **end** **as** Phone1
31. ,case **when** b.Phone2 **is** null **then** a.Phone2 **else** b.Phone2 **end** **as** Phone2
32. ,case **when** b.Phone3 **is** null **then** a.Phone3 **else** b.Phone3 **end** **as** Phone3
33. ,case **when** b.Email1 **is** null **then** a.Email1 **else** b.Email1 **end** **as** Email1
34. ,case **when** b.Email2 **is** null **then** a.Email2 **else** b.Email2 **end** **as** Email2
35. ,case **when** b.NationalTaxRateDesc **is** null **then** a.NationalTaxRateDesc **else** b.NationalTaxRateDesc **end** **as** NationalTaxRateDesc
36. ,case **when** b.NationalTaxRate **is** null **then** a.NationalTaxRate **else** b.NationalTaxRate **end** **as** NationalTaxRate
37. ,case **when** b.LocalTaxRateDesc **is** null **then** a.LocalTaxRateDesc **else** b.LocalTaxRateDesc **end** **as** LocalTaxRateDesc
38. ,case **when** b.LocalTaxRate **is** null **then** a.LocalTaxRate **else** b.LocalTaxRate **end** **as** LocalTaxRate
39. ,case **when** b.AgencyID **is** null **then** a.AgencyID **else** b.AgencyID **end** **as** AgencyID
40. ,case **when** b.CreditRating **is** null **then** a.CreditRating **else** b.CreditRating **end** **as** CreditRating
41. ,case **when** b.NetWorth **is** null **then** a.NetWorth **else** b.NetWorth **end** **as** NetWorth
42. ,case **when** b.MarketingNameplate **is** null **then** a.MarketingNameplate **else** b.MarketingNameplate **end** **as** MarketingNameplate
43. ,b.IsCurrent
44. ,b.BatchID
45. ,b.effectivedate
46. ,b.enddate
47. **from** **temp** b **inner** join new\_dimcustomer a **on** a.rn = b.rn-1 and a.customerid = b.customerid
48. )
50. **update** dimcustomer **as** old\_cust
51. **set** TaxID = new\_cust.TaxID  ,
52. Status = new\_cust.Status  ,
53. LastName = new\_cust.LastName  ,
54. FirstName = new\_cust.FirstName  ,
55. MiddleInitial = new\_cust.MiddleInitial  ,
56. Gender = new\_cust.Gender  ,
57. Tier = new\_cust.Tier  ,
58. DOB = new\_cust.DOB  ,
59. AddressLine1 = new\_cust.AddressLine1  ,
60. AddressLine2 = new\_cust.AddressLine2  ,
61. PostalCode = new\_cust.PostalCode  ,
62. City = new\_cust.City  ,
63. StateProv = new\_cust.StateProv  ,
64. Country = new\_cust.Country  ,
65. Phone1 = new\_cust.Phone1  ,
66. Phone2 = new\_cust.Phone2  ,
67. Phone3 = new\_cust.Phone3  ,
68. Email1 = new\_cust.Email1  ,
69. Email2 = new\_cust.Email2  ,
70. NationalTaxRateDesc = new\_cust.NationalTaxRateDesc  ,
71. NationalTaxRate = new\_cust.NationalTaxRate  ,
72. LocalTaxRateDesc     = new\_cust.LocalTaxRateDesc  ,
73. LocalTaxRate = new\_cust.LocalTaxRate  ,
74. AgencyID = new\_cust.AgencyID  ,
75. CreditRating  = new\_cust.CreditRating  ,
76. NetWorth = new\_cust.NetWorth  ,
77. MarketingNameplate = new\_cust.MarketingNameplate
78. **from**  new\_dimcustomer **as** new\_cust
79. **where** new\_cust.sk\_customerid = old\_cust.sk\_customerid
80. --order by new\_cust.customerid, new\_cust.effectivedate;

# Query 2

1. --Delete the records for the accounts that changed more than once in a batch date
2. --Example: Accountid = 138,159 and 270
3. **with** tmp **as**(
4. **select** a.accountid, **max**(a.sk\_accountid) **as** sk\_accountid
5. ,**min**(a.sk\_accountid) **as** sk\_accountid\_new, **max**(enddate) **as** enddate\_new
6. **from** dimaccount a
7. **inner** join(
8. **select** accountid, effectivedate **from** dimaccount **where**
9. accountid in
10. (
11. **select** accountid **from** dimaccount **where** effectivedate = enddate
12. )
13. **group** **by** accountid, effectivedate
14. **having** count(1)>1
15. ) b **on** a.accountid = b.accountid and a.effectivedate = b.effectivedate
16. **group** **by** a.accountid)
17. ,deleted **as**(
18. **delete** **from** dimaccount a
19. using tmp b **where** a.sk\_accountid = b.sk\_accountid
20. returning a.\*
21. )
23. **update** dimaccount a
24. **set** iscurrent = b.iscurrent
25. ,enddate = b.enddate
26. **from** deleted b, tmp c
27. **where** c.sk\_accountid = b.sk\_accountid and a.sk\_accountid = c.sk\_accountid\_new;
29. --recreate the unique index
30. **CREATE** **UNIQUE** **INDEX** UI\_DimAccount **ON** DimAccount (AccountID, EndDate);