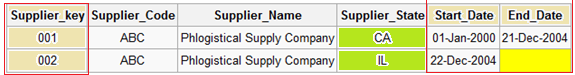
The **Type 1** methodology overwrites old data with new data, and therefore does not track historical data at all.

|  |  |
| --- | --- |
| Previous | After : The Supplier\_State was changed |
| [clip_image002[4]](http://blogs.msdn.com/cfs-file.ashx/__key/CommunityServer-Blogs-Components-WeblogFiles/00-00-01-14-37-metablogapi/7450.clip_5F00_image0024_5F00_7D53A4CF.png) |  |

The obvious disadvantage to this method of managing SCDs is that there is no historical record kept in the data warehouse. But an advantage to this is that these are very easy to maintain.

|  |  |
| --- | --- |
| The **Type 2** method tracks historical data by creating multiple records in the dimensional tables with separate keys. With Type 2, we have unlimited history preservation as a new record is inserted each time a change is made. | The supplier moves to Illinois, we have version (to know the current) and we include a unique key (sequential most of the time) |

Another option instead of using **version**, is to use **Start\_Date and End\_Date**:



Null **End\_Date** signifies current tuple version. In some cases, a standardized surrogate high date (e.g. 9999-12-31) may be used as an end date, so that the field can be included in an index.

Transactions that reference this Surrogate Key (Supplier\_Key) are then permanently bound to these time slices defined by each row in the slowly changing dimension table. If there are retrospective changes made to the contents of the dimension, or if a new set of attributes are added to the dimension (for example a Sales Rep column) which have different effective dates to those already defined, then this can result in the existing transactions needing to be updated to reflect the new situation. This can be an expensive database operation, so Type 2 SCD is not a good choice if the dimensional model is subject to change.

The **Type 3** method tracks changes using separate columns. Whereas Type 2 had unlimited history preservation, Type 3 has limited history preservation, as it's limited to the number of columns we designate for storing historical data. Where the original table structure in Type 1 and Type 2 was very similar, Type 3 will add additional columns to the tables:



Note that this record cannot track all historical changes, such as when a supplier moves twice. Although, one possible version of this type is to create the field Previous\_Supplier\_State instead of Original\_Supplier\_State which will then track all historical changes.

|  |  |
| --- | --- |
|  | **Note: Type 3, keeps separate** **columns for both the old and new attribute values—sometimes called “alternate** **realities.” Type 3 is less common because it involves** **changing the physical tables and is not very scalable.**  Source Table: .**SourceEmployees**  Destination Table: **DestinationEmployee** |
| Source | Execute Package for the first time and will make a copy of the source table into the destination |

Now let us fire following statement. Then execute the package and look the difference:

**update SourceEmployees**

**set managerid=1002**

**where empname='G'**

|  |  |
| --- | --- |
| Destination |  |
|  | EmpBusinessID for 1007 is duplicated.  **The old row has:**  -EmpSurrogateID (sequence) 7  -EndDate value, which means until when this data is valid  **The newest one has:**  -EmpSurrogateID of 11 (last one)  -EndDate Null, means the active one. |

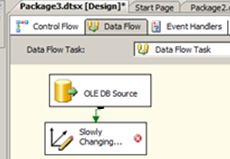
**EmpBusinessID**  will be the **Business Key** must be allowed to be duplicated

Step of Implementing SCD from SSIS

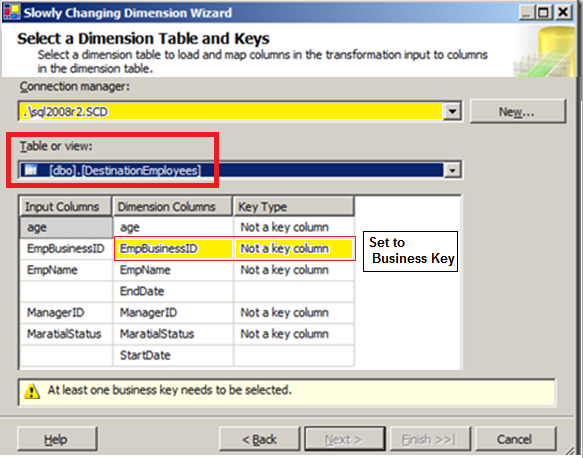
1. Got to Data Tab and drop one OLE DB Source & Configure it for database ->Source Table

|  |  |
| --- | --- |
| Control | Configure Connection |
|  |  |

2) From Toolbox drag & drop in Data Flow, connect OLE DB Source created in step 2 to SCD



1. Double Click on SCD will initiate SCD Wizard. Select a **Dimension Table and Keys**, here you need to specify the destination table.  **DestinationEmployees**



* First select the destination connection manager
* Then the table
* Then map the source input column to the target dimension column.
* Lastly, select one key to be your business key (the primary key from the source system is sometimes called the alternate key).

Specify **EmpBusinessID** as **Business Key** this field **IS** **NOT** unique

The unique field is **EmpSurrogateID**

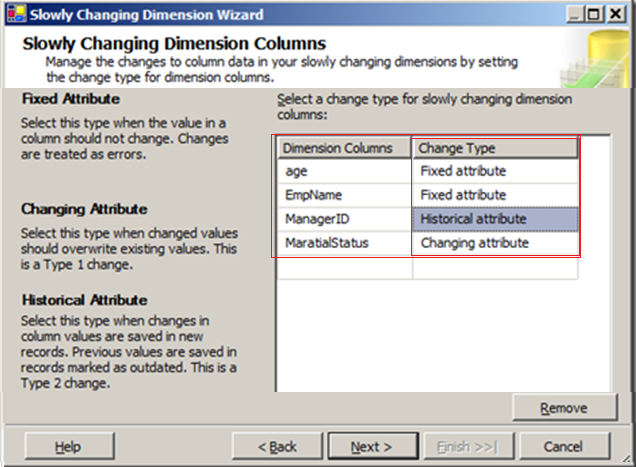
4) Click on **next tab**, this will allow you to specify the Change Type.

There are three kinds of Attributes:

* **Fixed Attribute** - For which value is not changing (**Type 0**)
* **Changing Attribute** - For which value will be over-written (**Type 1**)
* **Historical Attribute**- For which value is saved as a new record. Previous values are saved in the records marked as outdated (**Type 2**)

Include 4 fields: **age, empName, ManagerID, MaratialStatus**:

* Type 0: **age** is never allow to be changed
* Type 0 **EmpName** is never allow to be changed
* Type 1 **ManagerID** is allowed to be changed (sets the actual manager, but it will not generate history
* Type 2 **MaratialStatus** is allowed to be changed, and it will generate a historical information (new row)



5) You can check or uncheck as per your requirement, I am un-checking these two

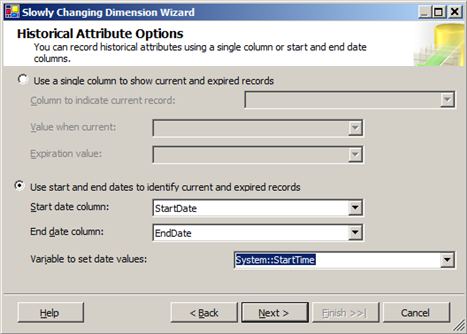
|  |  |
| --- | --- |
|  | Uncheck “Fail the transformation…. “ means, don’t fail if you find that the data has changed |

6) Specify how you want record changed value, using single column or start / end date

If any of columns are set to a **Historical Attribute** (Type 2), then in a few screens you are asked how you want to expire the row and create a new row.

There are 2 options to manage:

1. The top section allows you to have an **active-current column** where you just set a value defined by you.
2. The bottom section sets a **start date and an end date column** to a date system or user variable.



7) While loading data in Fact tables we usually see a scenario where the fact data is available but there is no corresponding business key in the related dimension.

In this case we choose **multiple options** to resolve the issue.

Ignore that fact

Insert the associated business key in dimension table and return the newly generated surrogate key from dimension table. And now store the data in Fact table with the surrogate key.

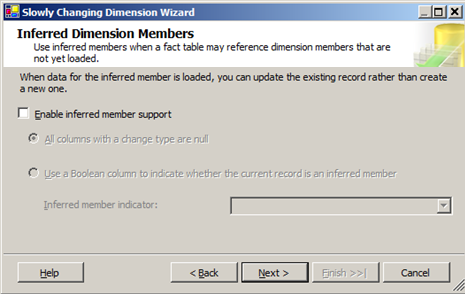
The second approach relates to a term called “**Inferred members**”. All the other attributes of that dimension will also be updated in next run of dimension load (usually nightly load).

For further information refers below mentioned links:-

[SSIS approach to handle Inferred Members](http://mohitnayyar.blogspot.com/2008/05/ssis-approach-to-handle-inferred.html)

[Inferred Members Implementation Best Practices](http://www.bp-msbi.com/2009/08/inferred-members-implementation-best/)

For this demo, I am not checking **Enable Inferred** member support



Click Next and Finish

LOAD 4

SELECT Event\_Id,Emp\_Id,AI\_Ticket\_iD AS AI\_TICKET,Event\_Type\_Id,Event\_Date

FROM EVENT

WHERE EVENT\_ID IN (1340867,1340868,1340869,1412347)

LOAD 6

SELECT Event\_Id,Emp\_Id,AI\_Ticket\_iD AS AI\_TICKET,Event\_Type\_Id,Event\_Date

FROM EVENT

WHERE EVENT\_ID IN (1340867,1340868,1340869,1412347,1472500,1477952)

WHERE EVENT\_ID IN (1340867,1340868,1340869,1412347,1472500,1477952,1485009,1501807,1522301,1604367)

Update SCD\_TEST

set StartDate = getdate()-1

Update SCD\_TEST

set AI\_Ticket = 1,

Emp\_Id = 2

where Event\_ID = 1340867

UPDATE SCD\_TEST

SET EVENT\_DATE = GETDATE()

WHERE EVENT\_ID = 1340868

Update SCD\_TEST

set Event\_Type\_Id = 'X'

where Event\_ID = 1412347

Update SCD\_TEST

set AI\_Ticket = 2

where Event\_ID = 1340868

Update SCD\_TEST

set Emp\_Id = 3

where Event\_ID = 1340869

emp\_id type 0

ai\_ticket type 1

Event\_Type\_Id 2

Event\_date no definido pero modifiquelo