Backdating Scenarios in Type 2 Dimensions

By Anil Jacob, Senior Consultant at iOLAP, Inc.

Recently, I had an interesting conversation with my project's data architect regarding possible back-dated changes for a primary dimension -- Employee -- in the data warehouse. In our existing data model, Employee was maintained as a Type 2 slowly changing dimension.

Six months into deployment, it was confirmed by business management that employee changes could be back-dated. This conversation reminded me of a project that I was part of three years earlier where such back-dated scenarios happened frequently.

Many of us are familiar with design sessions where business leaders confirm the chances of a certain scenario happening is rare. The architect then designs a solution without considering this supposedly rare scenario at all. After one month of deployment, it is noticed that this scenario occurs and breaks the existing design. Back-dated changes are an example of such "rare scenarios."

In addition to the challenge we faced in designing a solution that tackles back dated changes, such scenarios bring in additional complexities such as possible insertion of records within historical date ranges, surrogate keys of dimensions getting out of sequence, the need to update historical fact data with new surrogate keys, and altering of summarized data in fact tables/aggregate fact tables.

After the code was deployed to the production environment the team realized that this "rare scenario" happened more than anticipated. For the first few occurrences of this scenario, manual updates were performed to clean the data. Soon it was decided to re-design the architecture to handle back-dated scenarios.

With rapid technological innovations in the area of business intelligence, business management expects solutions to all their issues as a return for their investment. Often overlooked are the complexities that scenarios like back-dating presents. I am aware that Oracle BI handles such back-dated changes as part of their solution, but not all systems do. As the BI field becomes more mature, vendor systems will accommodate occurrences of such back-dated scenarios as a normal situation in any data warehousing design. Everyone's goals is to have a data warehousing solution that is committed to keeping the data as clean as possible.

In the remainder of this article, I will present a logical approach that you can use to tackle back-dated scenarios.

A conventional type 2 slowly changing dimension is modeled with an educated assumption that all changes coming from the source occur going forward. The following diagram depicts the conventional design of a type 2 dimension. To illustrate a scenario, I have given an example below.

Employee XYZ resides in California, and was hired on 1/1/2010.

Exhibit 1

Emp_No	Emp_Name	Location	Hire Date	End Date
1	XYZ	California	1/1/2010	

On 5/1/2013, we were notified from the source system that this employee has moved to Texas. The source table src_Employee looks as follows:

Exhibit 2

Emp_No	Emp_Name	Location	Hire Date	End Date
1	XYZ	Texas	1/1/2010	

In a conventional type 2 slowly changing dimension -- Dim_Employee, the following records are created.

Exhibit 3

Emp_ID	Emp_No	Emp_Name	Location	Eff_Start_Date	Eff_End_Date
100	1	XYZ	California	1/1/2010	4/30/2013
200	1	XYZ	Texas	5/1/2013	

On 7/1/2013, we are informed by the user that this employee had relocated to Texas on 3/1/2013. In a scenario in which 'location' is the only column to trigger historical employee records, the start and end dates in 'DIM_Employee' could be updated for each record.

Exhibit 4

Emp_ID	Emp_No	Emp_Name	Location	Eff_Start_Date	Eff_End_Date
100	1	XYZ	California	1/1/2010	2/28/2013
200	1	XYZ	Texas	3/1/2013	

However, in a real life scenario, there will be multiple columns that trigger historical records for a dimension. In the previous example, let us consider Position to be another critical attribute for Employee.

Employee was hired on 1/1/2010 as a Software Engineer. He resides in California.

Exhibit 5

Emp_No	Emp_Name	Location	Position	Hire Date	End Date
1	XYZ	California	Software Engineer	1/1/2010	

On 4/1/2013, he gets promoted to the position of a Senior Software Engineer.

Exhibit 6

Emp_No	Emp_Name	Location	Position	Hire Date	End Date
1	XYZ	California	Senior Software Engineer	1/1/2010	

On 5/1/2013, we get a trigger from the source system that this employee has relocated to Texas.

Exhibit 7

Emp_No	Emp_Name	Location	Position	Hire Date	End Date
1	XYZ	Texas	Senior Software Engineer	1/1/2010	

In the conventional type 2 slowly changing dimension, the following records are created for the employee.

Exhibit 8

Emp_ID	Emp_No	Emp_Name	Location	Position	Eff_Start_Date	Eff_End_Date
100	1	XYZ	California	Software Engineer	1/1/2010	3/31/2013
200	1	XYZ	California	Senior Software Engineer	4/1/2013	4/30/2013
300	1	XYZ	Texas	Senior Software Engineer	5/1/2013	

On 7/1/2013, we are informed by the user that this employee had relocated to Texas on 3/1/2013. In this scenario, we cannot go ahead and update the effective dates of each record.

Assumption: There is a source table 'src_Employee_location' that maintains start and end date of each employee and his corresponding location.

Emp_No	Location	Start Date	End Date	Update Date
1	Texas	3/1/2013		7/1/2013
1	California	1/1/2010	2/28/2013	7/1/2013

- Design a typical Type 2 dimension for Employee. The image of Dim_Employee after this step will be as seen in exhibit 3.
- Identify all records that have changed src_Employee_location (based on Update_Date).
- Join the changes in src_Employee_location to DIM_Employee based on the following condition
  Emp_No=Emp_No and Start Date between eff_start_date and eff_end_date or eff _st_date between Start Date and End Date

After this join operation, the output looks as follows:

Dim_Employee							src_Employee				
Emp_ID	**Emp_No**	**Emp_Name**	**Location**	**Position**	**Eff_Start_Date**	**Eff_End_Date**	**Emp_No**	**Location**	**Start Date**	**End Date**	**Update Date**
100	1	XYZ	California	Software Engineer	1/1/2010	3/31/2013	1	Texas	3/1/2013		7/1/2013
100	1	XYZ	California	Software Engineer	1/1/2010	3/31/2013	1	California	1/1/2010	2/28/2013	7/1/2013
200	1	XYZ	California	Senior Software Engineer	4/1/2013	4/30/2013	1	Texas	3/1/2013		7/1/2013
300	1	XYZ	Texas	Senior Software Engineer	5/1/2013		1	Texas	3/1/2013		7/1/2013

Scenario 1

Start date (source) < eff_st_date (target) and end date(source) > eff_end_date (target)

Description- The date range of the target record is within the date range of source record.

Resolution- Update existing record with source location.

Scenario 2

Start date (source) between eff_st_date (target) and eff_end_date (target) and End Date(source) between eff_st_date(target) and eff_end_date(target)

Description- The date range of the source record within the date range of target record.

Resolution- Insert new record that retrieves location from source record and remaining data from target record, record start date= start date (source), record end date = end date (source)

Scenario 3

End Date (source) >= eff_end_date(target) and Start Date (source) > eff_start_date (target)

Description- Overlap of source and target records in which source start date in between target start and end dates

Resolution- Insert new record that retrieves location from source record and remaining data from target record, record start date= start date(source), record end date = eff_end_date (target)

Scenario 4

End Date (source) < eff_end_date(target) and Start Date (source) <= eff_start_date (target)

Description- Overlap of source and target records in which source end date in between target start and end dates

Resolution- Update existing record that retrieves all data from target record, record start date remains same, record end date = end date (source)

The data in Dim_Employee after the above pseudo code has been applied looks as follows:

Dim_Employee								
Emp_ ID	**Emp_ No**	**Emp_Na me**	**Locatio n**	**Position**	**Eff_Start_D ate**	**Eff_End_D ate**	**Scenar io**	**Type**
100	1	XYZ	Califor nia	Software Engineer	1/1/2010	2/28/2013	4	Upda te
400	1	XYZ	Texas	Software Engineer	3/1/2013	3/31/2013	3	Insert
200	1	XYZ	Texas	Senior Software Engineer	4/1/2013	4/30/2013	1	Upda te
300	1	XYZ	Texas	Senior Software Engineer	5/1/2013		1	Upda te

About the Author

Anil specializes in designing Business Intelligence solutions and has experience working on various technologies including Informatica, MicroStrategy and DataStage. He has worked with large enterprise clients such as AT&T, Walmart and Gilbane. Anil received his MS in Management Information Systems from the Texas A&M University, Mays Business School.

About iOLAP

iOLAP, Inc. (www.iolap.com) is a Dallas-based strategic data consultancy specializing in Data Warehousing and Business Intelligence strategy and solutions. iOLAP is completely focused on the Data Warehousing (DW) and Business Intelligence (BI) markets and brings a client-centric and business-focused perspective to all of its engagements. iOLAP has been in business since 1999 with in-depth expertise across all Data Warehousing and Business Intelligence technology areas, including the newest leading-edge Business Analytics and Big Data architectures. iOLAP serves some of the world's largest companies across all major industries. All sales and support comes from our centralized offices in Texas and California.

For More Information

For more information about iOLAP services, call (214) 618-5000 or email us at info@iolap.com. Please visit us at http://www.iolap.com.

Business Intelligence + Big Data