Psuedo-deltas: An Approach Customized for WFM/iTime Data Extraction

Applies to:

SAP BI 7.x.

Summary

This article illustrates a data flow technique for reliably and efficiently extracting data to BW from WFM/iTime 3.1, an SAP application. Challenges were faced due to data deletion in the source system, along with lack of reliable change markers in all relevant tables. A discussion of possible techniques is presented, along with an overview of the selected approach.

Author: Luther Blake

Company: Daugherty Business Solutions

Created on: 26 August 2010

Author Bio

Luther Blake is currently a Consultant in the Business Intelligence / Data Management Line of Service at Daugherty Business Solutions in Saint Louis, MO. He holds a Master of Science in Computer Science from Arizona State University, and is focused technically on the Microsoft SQL Server and SAP Business Warehouse platforms. He is currently working on MM-Purchasing, and has just survived a HCM and FI BW implementation.

Table of Contents

Introduction

The concept of Deltas for data extraction to SAP Business Warehouse (BW) is central to efficient data loads, and there are numerous technical articles covering the different types of Delta extraction from R/3 datasources.

The concept of a Pseudo-delta comes into play when there is no Delta capability provided by the datasource itself. For instance, many standard content datasources do not have built-in delta capability. There are well known methodologies for coping with most data loads from SAP R/3, as SAP Business Content along with an army of well-trained consultants have typically solved the problem before we encounter it. A typical solution is to utilize some date field along with an InfoPackage selection routine, to reload an 'overlap' of data and let a standard Data Store Object (DSO) with a properly designed key sort it out.

In a perfect world, we always have reliable change markers available to us. We also hope that records are marked as deleted, rather than physically removed from the relational data layer. However, consider the situation where a source system directly deletes out of its tables. If we are loading to a standard DSO, we have no way to detect that a row is no longer relevant. As a new value of the key will not be coming in via the InfoPackage, the old value will remain in the DSO, and will effectively be a ghost in the system. This certainly impacts data quality, and can wreak havoc with reporting.

This article will provide an overview of one solution to implement windowed loads for a datasource which directly deletes. While the solution is specific to a scenario encountered with one particular source system, the approach reinforces the utility of MultiProviders with respect to modeling decisions, and components of the solution may be applicable to simpler scenarios.

Background

The original implementation of this approach was constructed to handle data extraction from SAP Workforce Management (WFM)/iTime 3.1 during product Ramp-up. The WFM/iTime application has a portal-based interface to the end users, and its operational data is stored directly in SQL Server. It is not a Netweaver-based application. For this reason, there is no standard content available for the raw timeclock data. Hence, the business desired to utilize BW to deliver operational reporting requirements as part of Go-Live scope.

To provide for the reporting requirements, BW datasources were implemented using DBConnect. These datasources were initially utilized as a full load, and fed into a layer of standard DSOs. A subset of these DSOs then fed to a set of InfoCubes for higher-level analysis.

As timeclock data can be rather granular, it was apparent soon after Go-Live that the full load solution was unacceptable for long-term support. However, there was the situation of the deleted rows! In order to cope, the existing solution was to delete the entire data contents of each DSO prior to each load.

This particular data source also has the characteristic that data becomes more stable over time. Data for the prior complete week and the current week may be updated. However, at some point during the week, the prior week becomes locked and will no longer be subject to modification or deletion on the source system. This is dependent upon the business, and how well the weekly payroll processing is going.

In summary, we have a data source with the following characteristics:

- Data stabilization over time (Past week + WTD potentially volatile)
- No reliable change markers
- Changed Date/Time not present in all tables
    - Create Date is generally available
    - Source data can be physically deleted out of database for a period of time

Candidate Solutions

The list below provides a basic enumeration of the approaches which were considered.

1. Traditional full load
2. Traditional pseudo-delta
    a. Traditional pseudo-delta based on transaction date e.g. sydatum (extract only current day's data)
    b. Traditional pseudo-delta based on extracting a particular time window (e.g. 2 weeks)
3. Pseudo-delta plus logically partitioned DSOs
    - Current DSO and Historical DSO
    - Transformation to periodically copy current data to historical DSO
4. Pseudo-delta plus logically partitioned DSOs – Revised approach
    1. Current DSO and Historical DSO
    2. Initialization and daily batch jobs load both DSOs from source
    3. No transformation from one DSO to the other
5. Pseudo-delta with ABAP to delete overlapping selections in DSO layer

A traditional full load (Option 1) sufficed at the time of implementation, but was clearly not a long-term solution.

A traditional pseudo-delta based on InfoPackage selection of the current day's transaction date (Option 2 (a)) was considered, and would be appropriate in many cases. However, this particular implementation suffered from the data stabilization issue, and coping with changes where the data was changed in the past complete week or the current week. Option 2(b), which extends 2(a) to provide an overlapping window for extraction, would suffice in most cases. This particular datasource requires a full deletion of the overlapping time window. While there are built-in methods which will provide this capability with InfoCubes (Delete Overlapping Requests from InfoCube process type in the Process Chain), this becomes trickier when we desire to load into a DSO.

As we worked this problem, an Option 3 began to take shape. We are all familiar with logically partitioning our InfoCubes by time, so why not reuse the concept and use two DSOs? In this case, we would maintain one DSO which would contain the history, and another DSO which would contain only the most recent two weeks of data. We could load the Historical DSO from the Current DSO at the end of the week, as data rolled over into the new week. These DSOs are then unioned under a MultiProvider to meet the operational reporting requirements.

Option 3 has met most of the requirements, allowing only the most recent data to be loaded in the nightly batch window. Historical data can be maintained. However, there is a management headache – when do we copy the data from the Current DSO to the Historical DSO? Remember that the data in the Current DSO, as it is still volatile, must be completely deleted before loading each and every time. The obvious solution is to schedule a new process chain, which will copy the past complete week to the Historical DSO on Thursday or Friday, before the week rolls over to the new week. But now, the data exists in both DSOs until the week boundary is rolled over. With the desired approach of joining with a Multiprovider, we would have to jump through some additional hoops to make this work properly, either scheduling the job directly prior to midnight on the day that our date boundary rolls over, or developing more sophisticated logic. We chose to move on and look for other options rather than work through the possible timing issues with this approach.

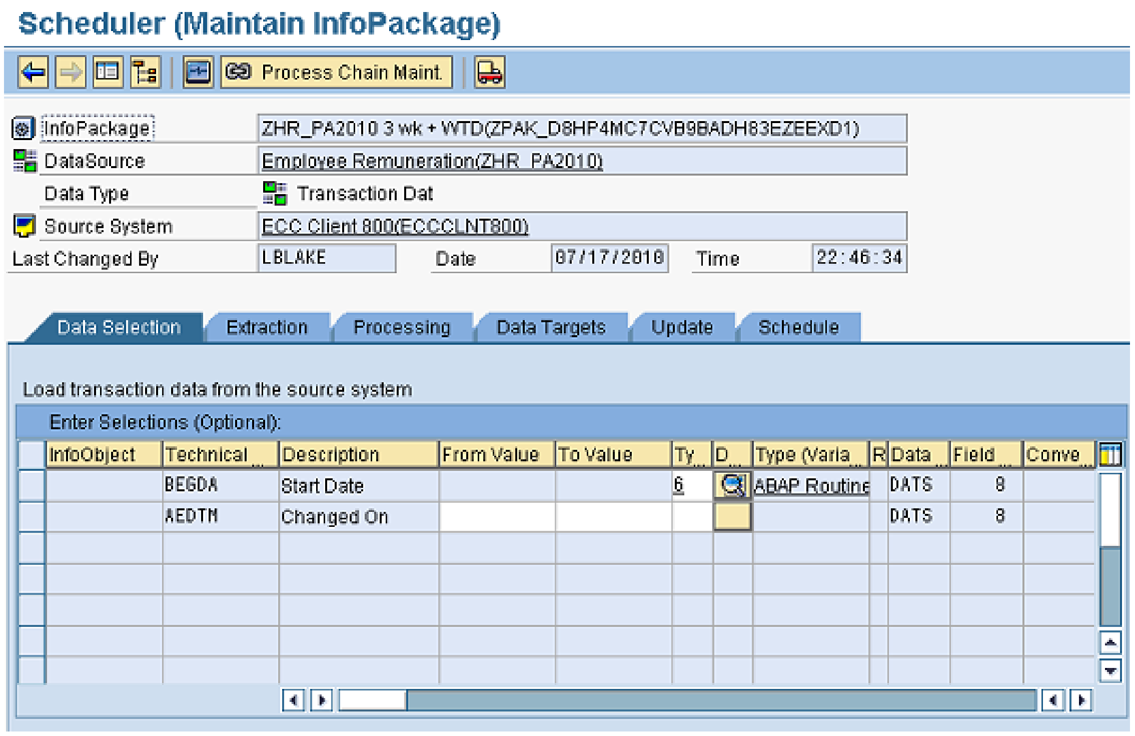
Option 4 was born out of the implementation problems with Option 3. We use the same overall concept of logically partitioning between two DSOs. However, we map each DSO to the source, and utilize ABAP routines to populate the selection criteria and route the data appropriately. The approach is unique, as data is always written to both DSOs with each load. We purposely load 3 weeks plus week to date from source, and write back the oldest week to the Historical DSO for the entire week. This is our safety interval.

Option 5 is also a candidate. As an alternative to Option 4, which seeks to address the problem by directing incoming data to DSOs utilizing disjoint partitioning criteria, the data could first be deleted by the implementation of custom ABAP code to determine the overlapping period of data and deleting the selection. This was briefly considered, and may be appropriate in some cases. This may be appropriate in some circumstances, such as on an existing implementation with many queries developed directly against the InfoProvider without the usage of a MultiProvider, where changing the data model directly may be cost-prohibitive. One must keep in mind, however, that deleting overlapping selections directly within the DSO active table will not remove entries from the change log. Option 5 would clearly need more investigation and effort to be a viable, reliable solution in this instance. Option 4 is more straightforward to implement, sustain, and transition to new support resources. Without significant evidence in favor of one approach over another, simpler always wins.

Data Flow

Below is an overview of the data flow of a sample implementation of this approach. For the purposes of illustration, we have reimplemented the original scenario using simplified naming. We have also used an R/3 data source for this example, for ease of implementation in our lab environment.

This data source is implemented as a generic extractor on InfoType 2010 in the HCM system. The PSA is loaded via two full load InfoPackages: One full load with no selection criteria for initialization and full load (if needed due to operational issue), and one InfoPackage which uses an ABAP selection routine to load three complete weeks of data plus the current week to date. In this example, we rely on the Start Date (field BEGDA) from the source, as we are still maintaining the scenario where the Changed On (field AEDTM) is unreliable in the source data.


Scheduler (Maintain InfoPackage)

InfoPackage	ZHR_PA2010 3 wk + WTD(ZPAK_D8HP4MC7CVB9BADH83EZEEXD1)
DataSource	Employee Remuneration(ZHR_PA2010)
Data Type	Transaction Dat
Source System	ECC Client 800(ECCCLNT800)
Last Changed By	LBLAKE
Date	07/17/2010
Time	22:46:34

Load transaction data from the source system

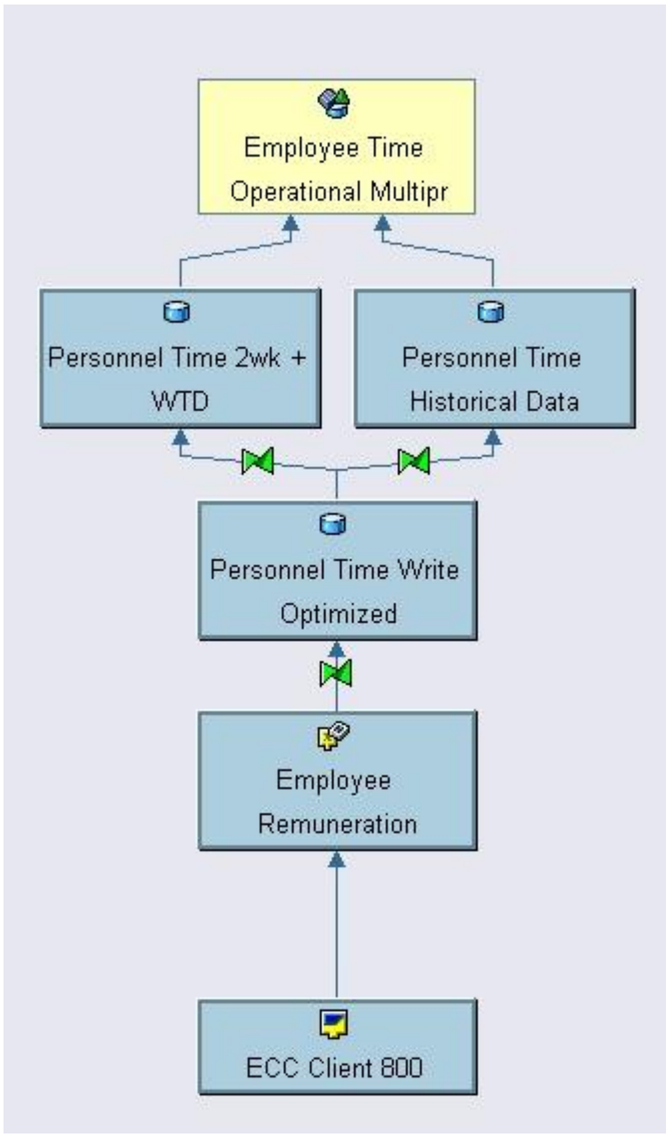
Enter Selections (Optional):

InfoObject	Technical ...	Description	From Value	To Value	Ty...	D...	Type (Varia...	R	Data ...	Field ...	Conve...
	BEGDA	Start Date			6		ABAP Routine		DATS	8	
	AEDTM	Changed On							DATS	8	

Note: Selection on transaction date, rather than change marker date

The data is further loaded into a write-optimized DSO. The insertion of a write-optimized DSO in this data flow is optional, and is a subject of data modeling topics outside of the scope of this article.

From the write-optimized DSO, the data is propagated to the Current and Historical standard DSOs. Data selection from the write-optimized DSO is achieved once again with ABAP selection routines in the filter of the DTP. We select the oldest, most stable week and load it into the historical DSO. Remember that this week is no longer subject to deletions on the data source, as the data which was locked after a period of time. This data will be copied to the Historical DSO every day until the next week when the date boundary rolls over, giving us a long window to catch any failures or errors with the overall process. This is in comparison with Option 3, where we would have had at most two days, depending on the ultimate solution which would have been implemented. The rest of the data is funneled into the Current DSO, which is truncated before every load.


Employee Time Operational Multipr
Personnel Time 2wk + WTD
Personnel Time Historical Data
Personnel Time Write Optimized
Employee Remuneration
ECC Client 800

Note: Sample data flow for Pseudo-delta example.

This approach gives us many advantages. First, both DSOs are loaded nightly in the standard batch process, avoiding the introduction of an additional weekly batch job to archive data. This additional batch job would have had to be timed and monitored to ensure the data is loaded into the Historical DSO before the week-end rollover. As the Historical DSO loads the stable week, there is no worry about deletions in the source data, and the standard DSO will operate as desired. The ABAP routines in the selection filters manipulate the dates automatically, so we have no flags to set or additional logic to manage from an Operations point of view. The utility of MultiProviders is also reinforced, allowing us to maintain a consistent reporting layer.

Below is an example which provides the basic template for our ABAP routine selections.

```
data: l_idx like sy-tabix,
      lc_startdate like sy-datum,
      lc_enddate like sy-datum,
      lv_week TYPE /BI0/OICALWEEK.

      read table l_t_range with key
        fieldname = 'DATE0'.
        l_idx = sy-tabix.

    CALL FUNCTION 'DATE_GET_WEEK'
    EXPORTING
       DATE = sy-datum
    IMPORTING
      WEEK = lv_week
      EXCEPTIONS
        DATE_INVALID = 1.


    CALL FUNCTION 'WEEK_GET_FIRST_DAY'
    EXPORTING
      week = lv_week
    IMPORTING
      DATE = lc_startdate
    EXCEPTIONS
      WEEK_INVALID = 1.

*Move two weeks back from beginning of week to get two completed weeks
    lc_startdate = lc_startdate - 14.

    l_t_range-FIELDNAME = 'DATE0'.
    l_t_range-LOW = lc_startdate.
    l_t_range-HIGH = '99991230'.
    l_t_range-SIGN = 'I'.
    l_t_range-OPTION = 'BT'.

    if l_idx <> 0.
       modify l_t_range index l_idx.
    else.
       append l_t_range.
    endif.
    p_subrc = 0.
```

Related Content

- http://www.sdn.sap.com/irj/scn/weblogs?blog=/pub/wlg/15145
- http://wiki.sdn.sap.com/wiki/display/NWTech/Processing+Delta+Records+in+Accounts+Payable
- http://www.sdn.sap.com/irj/scn/index?rid=/library/uuid/a0b3f0e2-832d-2c10-1ca9-d909ca40b54e

Disclaimer and Liability Notice